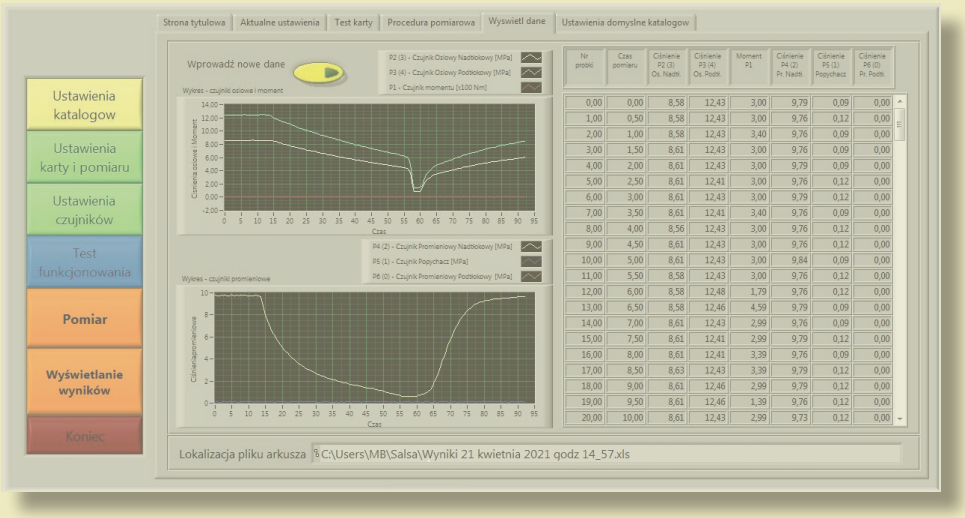
Politechnika Lubelska

PODRĘCZNIKI

Marcin Buczaj, Andrzej Sumorek

Podstawy monitorowania i diagnostyki układów mechatronicznych



Lublin 2023

Podstawy monitorowania i diagnostyki układów mechatronicznych

Podręczniki – Politechnika Lubelska

POLITECHNIKA
LUBELSKA
WYDZIAŁ
MECHANICZNY

POLITECHNIKA
LUBELSKA
WYDZIAŁ BUDOWNICTWA
I ARCHITEKTURY

POLITECHNIKA
LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI
I INFORMATYKI

Marcin Buczaj, Andrzej Sumorek

Podstawy monitorowania i diagnostyki układów mechatronicznych

POLITECHNIKA
LUBELSKA
WYDAWNICTWO

Lublin 2023

Recenzenci:
dr hab. inż. Krzysztof Okarma, prof. uczelni, Zachodniopomorski Uniwersytet Technologiczny w Szczecinie
dr inż. Wioletta Nowak, Politechnika Wrocławska

Publikacja wydana za zgodą Rektora Politechniki Lubelskiej

ISBN: 978-83-7947-564-3

Wydawca: Wydawnictwo Politechniki Lubelskiej
www.wpl.pollub.pl
ul. Nadbystrzycka 36C, 20-618 Lublin
tel. (81) 538-46-59
Druk: Drukarnia Akapit sp. z o. o.
drukarniaakapit.pl

Nakład: 50 egz.

Spis treści

Podstawy monitorowania i diagnostyki układów mechatronicznych

Streszczenie

Wspólny punkt zespolenia systemu mechanicznego, elektronicznego i informatycznego wymaga platformy, która będzie komunikowała się tak z urządzeniami fizycznymi, jak i z systemami przetwarzania danych. Taką platformą wydaje się być środowisko programowania graficznego LabVIEW (prod. National Instruments). W opracowaniu przedstawiono specyfikę pracy w środowisku LabVIEW. Główny nacisk położono na charakterystykę dostępnych obiektów oraz stosowanie typowych struktur programowania, takich jak np. pętle czy rejestry. Zakres przedstawianych zagadnień jest na tyle szeroki, że w podręczniku można odnaleźć opis podstawowych typów danych, zalecenia w stosunku do projektowania paneli czołowych oprogramowania, objaśnienia typowych architektur aplikacji. Struktura rozdziałów jest dwuczęściowa: opiera się na podziale na część zawierającą wstępne informacje teoretyczne, po których następuje część praktyczna. Struktura podręcznika została tak ułożona, że może on być wykorzystywany do wspomagania prowadzenia zajęć praktycznych. Możliwości opisywanej platformy programowania docenią tak inżynierowie mechanicy, chcący rozbudować urządzenia o funkcje automatyki, jak i programiści, chcący zaimplementować swoje algorytmy do urządzeń mechanicznych i elektrycznych.

Słowa kluczowe: graficzne środowisko programowania, LabVIEW, mechatronika, aplikacja kontrolno-pomiarowa

Fundamentals of monitoring and diagnostics of mechatronic systems

Summary

The mechanical, electronic, and IT systems need a platform that connects them and communicates with physical devices and data processing systems. It seems like LabVIEW, a graphical programming environment from National Instruments. The study presents the specifics of working in the LabVIEW environment. The primary emphasis was placed on the characteristics of objects and the use of typical programming structures, such as loops and registers. The scope of the presented issues is so broad that the content includes characteristics of basic data types, recommendations for designing software front panels, and characteristics of typical application architectures. The chapter structure is two-part. It is based on a division into a part containing introductory theoretical information, followed by a practical part. The structure of the work is organized in such a way that it can be directly used to support practical classes. Mechanical engineers can use LabVIEW programming platform to add automation features to their devices. Programmers can use it to implement their algorithms into mechanical and electrical devices.

Keywords: graphical programming environment, LabVIEW, mechatronics, control and measurement application

Wstęp

Źródłosłów pojęcia mechatronika łączy w sobie dwie dziedziny techniki: mechanikę i elektronikę. W praktyce oznacza to, że celem działań z zakresu mechatroniki jest budowa urządzeń mechanicznych o możliwościach poszerzonych o rozwiązania z zakresu elektroniki oraz automatyki i sterowania. Można też przyjąć interpretację, że układy mechatroniczne to elektroniczne układy automatyki i sterowania, wzbogacone o możliwość fizycznego oddziaływania na środowisko. W każdym z tych przypadków występuje połączenie w punkcie fizycznego zachowania obiektu oraz elektronicznego monitorowania lub sterowania, najczęściej razem z procesem podejmowaniem decyzji. W rezultacie obniżenie kosztów systemów komputerowych skutkuje wprowadzeniem do układów mechatronicznych systemów akwizycji danych, systemów sterowania i automatyki oraz systemów podejmowania decyzji. Wspólny punkt zespolenia systemu mechanicznego, elektronicznego i informatycznego wymaga platformy, która będzie komunikowała się tak z urządzeniami fizycznymi, jak i z systemami przetwarzania danych. Taką platformą wydaje się być środowisko programowania graficznego LabVIEW (prod. National Instruments).

Opracowanie poświęcono podstawowym możliwościom oferowanym przez środowisko LabVIEW. Przedstawiono specyfikę pracy użytkownika w tym środowisku, sposób podejścia do programów, które tutaj nazywane są przyrządami wirtualnymi. Główny nacisk położono na charakterystykę dostępnych obiektów oraz stosowanie typowych struktur programowania, takich jak np. pętle czy rejestry. Zakres przedstawianych zagadnień jest na tyle szeroki, że w treści można odnaleźć opis podstawowych typów danych, zalecenia w stosunku do projektowania paneli czołowych oprogramowania, objaśnienia typowych architektur aplikacji. Poza lokalnym dostępem do aplikacji zamieszczono także opis postępowania do danych i oprogramowania w sposób zdalny. Opracowanie zawiera również rozdziały prezentujące praktyczne zastosowanie opisywanych struktur w konkretnych rozwiązaniach wspomagających prowadzenie pomiarów, magazynowania danych i podejmowania decyzji.

Układ rozdziałów jest dwuczęściowy. Oparty jest na podziale na część zawierającą wstępne informacje teoretyczne, po których następuje część praktyczna. Część praktyczna to zestaw „problemów", które można rozwiązać, postępując zgodnie z zamieszczonymi instrukcjami. Każdy rozdział kończy zestaw pytań kontrolnych, które służą ugruntowaniu uprzednio pozyskanej wiedzy i umiejętności. Struktura pracy sprzyja wykorzystywaniu tego podręcznika do wspomagania prowadzenia zajęć praktycznych. Pomimo dołożenia wszelkich starań, autorzy zdają sobie sprawę, że przy realizacji zadań praktycznych mogą wystąpić problemy. Mogą one wynikać z różnic w wersji środowiska programistycznego i konfiguracji systemu operacyjnego stosowanego przez autorów i czytelników.

Środowisko LabVIEW dedykowane jest do graficznego opracowywania aplikacji współpracujących z szeroko pojętą aparaturą kontrolno-pomiarową. Wydaje się być interesującym rozwiązaniem dla użytkowników, którzy chcą scalić w jedno rozwiązanie urządzenia mechaniczne z systemami informatycznymi. Jego możliwości docenią zarówno inżynierowie mechanicy chcący rozbudować urządzenia o funkcje automatyki, jak i programiści chcący zaimplementować swoje algorytmy do urządzeń mechanicznych i elektrycznych.

1. Zajęcia wstępne

1.1. Cel zajęć

Ćwiczenie pierwsze ma charakter wprowadzający. Podczas wprowadzenia zostaną określone warunki wykonywania zajęć laboratoryjnych, przedstawiony zostanie zakres realizowanego materiału, zasady związane z wykonywaniem ćwiczeń oraz kryteria i warunki zaliczenia końcowego.

Do celów tych zajęć zaliczymy w szczególności poznanie:

- zasad współpracy podczas realizacji programu zajęć i reguł związanych z wykonywaniem ćwiczeń,
- szczegółowych zasad BHP oraz przyswojenie procedur bezpieczeństwa w laboratorium komputerowym,
- zakresu ćwiczeń realizowanych podczas zajęć laboratoryjnych,
- specyfiki obsługi laboratoryjnego stanowiska komputerowego i indywidualnych kont użytkowników,
- przeznaczonych do ćwiczeń treści internetowych oraz dostępności do zasobów sieciowych i funkcjonowania programu LabVIEW.

1.2. Procedury wstępne w laboratorium

1.2.1. Indywidualny profil użytkownika

W celu skonfigurowania prywatnej przestrzeni pracy zostaną przeprowadzone poniższe działania:

- utworzenie własnego profilu użytkownika na komputerze w laboratorium,
- ustanowienie hasła dla własnego profilu użytkownika,
- utworzenie na pulpicie katalogu przeznaczonego do obsługi programów wykorzystywanych podczas realizacji kolejnych ćwiczeń,
- skopiowanie do własnego katalogu plików przygotowanych przez prowadzącego.

1.2.2. Test funkcjonowania środowiska LabVIEW

Przed przystąpieniem do opracowywania własnych aplikacji konieczne jest:

- uruchomienie środowiska LabVIEW w zadeklarowanej przez prowadzącego zajęcia wersji programu i sprawdzenie poprawności jego działania;
- wykonaniu instrukcji testowych wskazanych przez prowadzącego,
- utworzeniu na pulpicie skrótu do programu LabVIEW.

2. Budowa przyrządu wirtualnego. Edycja elementów panelu czołowego i schematu blokowego

2.1. Cel zajęć

Ćwiczenie drugie dotyczy podstaw związanych z obsługą środowiska programistycznego LabVIEW i tworzeniem w nim programów użytkowych. Środowisko LabVIEW jest wiodącym programem do budowy przyrządów wirtualnych, stosowanych powszechnie przez inżynierów. Potencjał programu LabVIEW jest szczególne widoczny przy realizacji projektów z zakresu diagnostyki, monitorowania i symulacji procesów technologicznych.

Do celów tych zajęć zaliczymy w szczególności poznanie:

- obsługi środowiska LabVIEW i zarządzanie procesem tworzenia oprogramowania,
- budowy przyrządów wirtualnych,
- reguł korzystania z gotowych szablonów dostępnych w programie LabVIEW,
- edycji elementów panelu czołowego i schematu blokowego,
- zasad zarządzania narzędziami obsługi,
- sposobów diagnostyki funkcjonowania programów.

2.2. Wstęp

2.2.1. Środowisko programistyczne LabVIEW

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) to środowisko programistyczne firmy National Instruments (NI), służace do tworzenia aplikacji obsługujących komputerowe systemy pomiarowe, kontrolno-pomiarowe, diagnostyczno-pomiarowe oraz do prowadzenia symulacji [1, 2, 20].

Do tworzenia aplikacji wykorzystywany jest graficzny język programowania (programowanie w języku G). Idea tworzenia kodu źródłowego opiera się na graficznym przepływie danych od węzła do węzła. Zasada tworzenia oprogramowania przypomina język FBD, służący do programowania sterowników PLC i PAC.

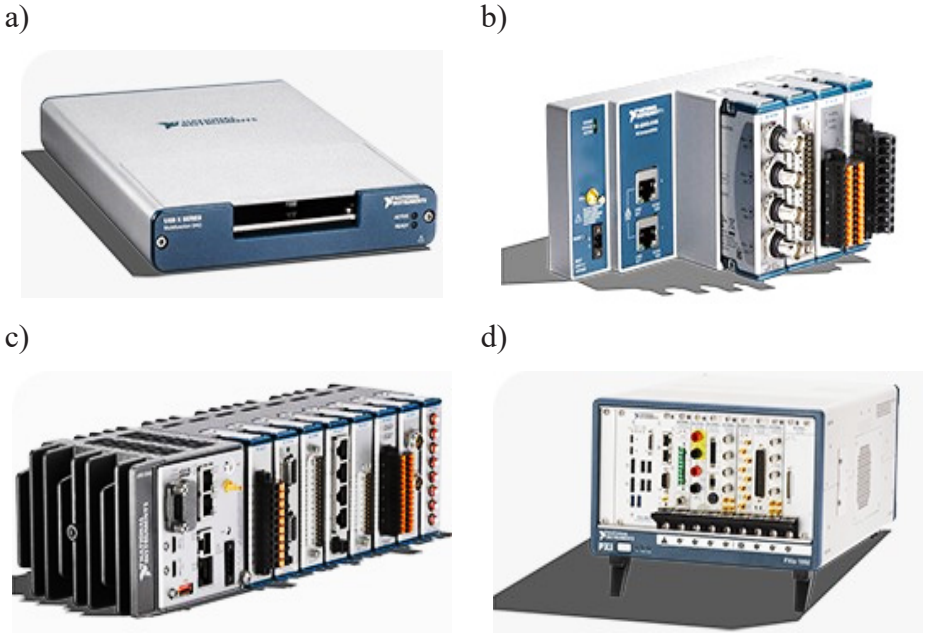
Środowisko LabVIEW jako pakiet programistyczny umożliwia budowę całych systemów kontrolno-pomiarowych oraz diagnostycznych. Poszczególne komponenty pozwalają na [1, 2, 14, 20]:

- realizację procesu pomiarowego i procesu sterowania (karty pomiarowe DAQ – moduły DIO i AIO),
- obsługę akwizycji i rejestracji danych,
- przetwarzanie danych pomiarowych,
- zarządzanie procesem przetwarzania i analizy danych,
- budowę symulatorów procesów fizycznych i matematycznych,
- symulację doświadczeń inżynierskich,

- budowę aplikacji i interfejsów użytkownika,
- porównywanie danych pomiarowych z wynikami symulacji,
- obsługę plików danych w różnych formatach,
- współpracę z innymi programami i sprzętem.

W zależności od konfiguracji, LabVIEW umożliwia wykorzystanie pakietów programowych i rozszerzeń, dających możliwość dostosowania zaimplementowanych charakterystycznych narzędzi i modułów programowych. Podstawowe z nich to: Analog DIAdem, NI-DAQmx, Datasocket, FPGA Compile, Industrial Controller, Robotics Analog i Digital Waveform Editor, OPC Servers, Real Time Trace Viewer, Data Communication, Signal Express, Sound and Vibration, Test Stand, Connectivity, Electrical Power, Instruments I/O, Arduino oraz Control & Simulation.

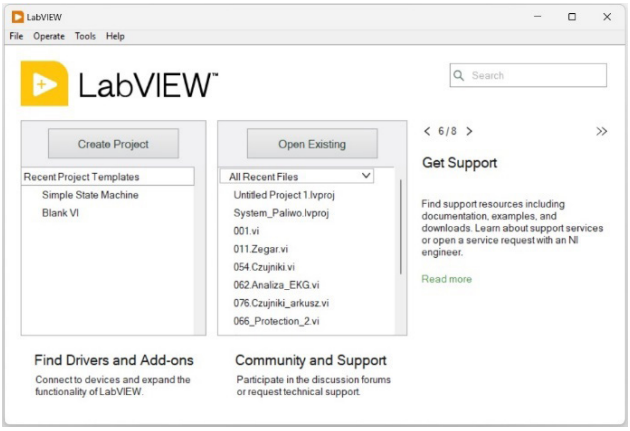
Środowisko programistyczne LabVIEW to nie tylko zaawansowany program do tworzenia aplikacji, ale również duża rodzina sprzętu umożliwiającego realizację procesów diagnostycznych, kontrolno-pomiarowych, symulacji, monitorowania i analizy danych pomiarowych. Wśród sprzętu rozróżnić można kilka kluczowych gam produktów (rys. 2.1) takich jak: karty pomiarowe podłączane do komputerów (PC-Based Systems), platformy pomiarowe pracujące w układzie z komputerem lub stanowiące autonomiczne układy (CompactDAQ, CompactRIO, PXI) [21].



Rys. 2.1. Reprezentacja różnych serii i typów produktów firmy National Instrument: a) PC-Based, Systems, b) CompactDAQ, c) CompactRIO, d) PXI [21]

2.2.2. Charakterystyka i obsługa programu LabVIEW

Środowisko LabVIEW jest programem umożliwiającym budowę aplikacji z zakresu różnych dziedzin inżynierii. Budowa i tworzenie aplikacji w dużej mierze opiera się na korzystaniu przez programistę z modułów funkcyjnych, dostosowanych do charakterystycznych działań związanych z przetwarzaniem danych [1, 4, 20]. Okno startowe programu LabVIEW zostało przedstawione na rysunku 2.2.



Rys. 2.2. Okno startowe programu LabVIEW w wersji 2022

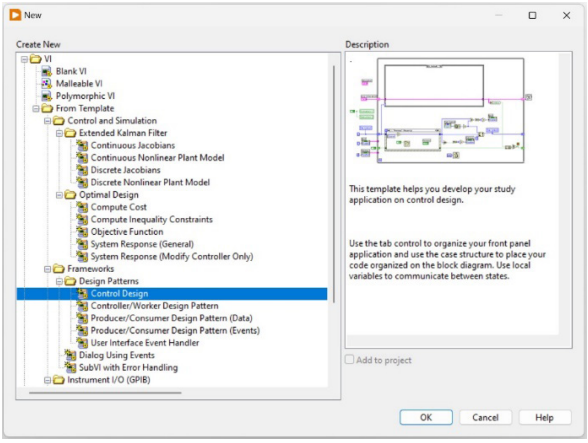
Do podstawowych cech programu LabVIEW zalicza się:

- możliwość programowania w języku graficznym (podobnym do języka FBD sterowników PLC),
- tworzenie programu w oknie front panelu (interfejs użytkownika) i oknie schematu blokowego (kod źródłowy),
- intuicyjne tworzenie kodu źródłowego za pomocą kolorowych nitek (połączenia automatycznie definiują rodzaj i typ przetwarzanych danych),
- możliwość tworzenia aplikacji z plikami wykonywalnymi (budowa programów do funkcjonowania niezależnie od zainstalowanego pakietu LabVIEW),
- możliwość tworzenia plików instalacyjnych (dopasowanie zainstalowanej wersji programu do konkretnej konfiguracji sprzętowej użytkownika),
- wybór wielu platform i systemów operacyjnych (Windows, Linux).

Podstawowym rodzajem programu tworzonego w środowisku LabVIEW jest instrument wirtualny VI (virtual instrument). Każdy z VI może być zbudowany ze zdefiniowanych w środowisku LabVIEW modułów funkcyjnych, narzędzi programowych oraz z innych VI. Program VI może działać i realizować określone zadania, ale tylko na komputerach z zainstalowanym środowiskiem LabVIEW. Jednostką organizacyjną wyższego rzędu jest projekt (project). Jest to zestaw danych zawierający konkretną konfigurację, dokumentację, definicję typów, zarządzanie zależnościami oraz specyfi-

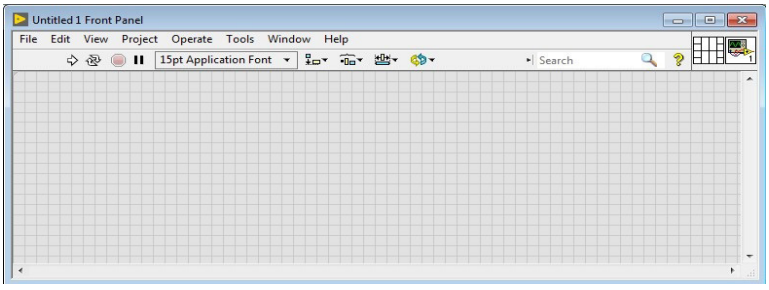
kację programu. Na podstawie projektu można budować aplikacje z plikiem wykonywalnym lub pliki instalacyjne. W tym przypadku tworzone są programy funkcjonujące poza środowiskiem LabVIEW, bez konieczności jego instalowania u końcowego użytkownika.

Proces realizacji każdego programu (VI) może odbywać się poprzez tworzenie go od początku (blank VI) lub korzystanie z dostępnych szablonów. W przypadku użycia dostępnego szablonu należy posłużyć się gotowym programem, który wystarczy jedynie dostosować do własnych oczekiwań. Dostępnych jest wiele plików szablonów (rys. 2.3), można także stworzyć i zapisać swój własny. Jest to korzystne w przypadku opracowywania aplikacji bazujących na tym samym kodzie programu lub zawierających określoną szatę front panelu.

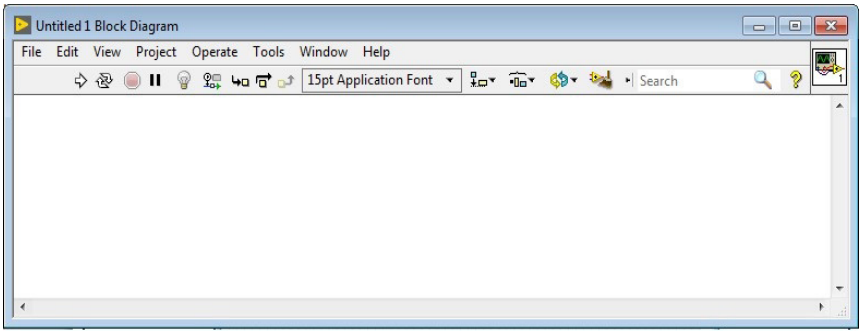


Rys. 2.3. Okno wyboru szablonu programu

W przypadku wyboru opcji tworzenia programu od początku (Blank VI), program wygeneruje dwa puste okna. Jedno z nich to okno front panelu, na którym buduje się interfejs programu (rys. 2.4). Drugie to okno schematu blokowego (rys. 2.5), na którym tworzy się kod źródłowy programu.



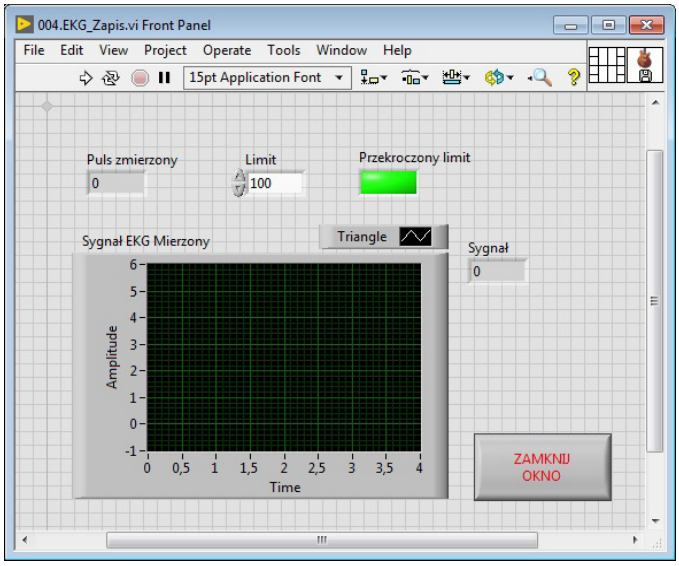
Rys. 2.4. Puste okno interfejsu (Front Panel) w programie LabVIEW



Rys. 2.5. Puste okno schematu blokowego (Block Diagram) w programie LabVIEW

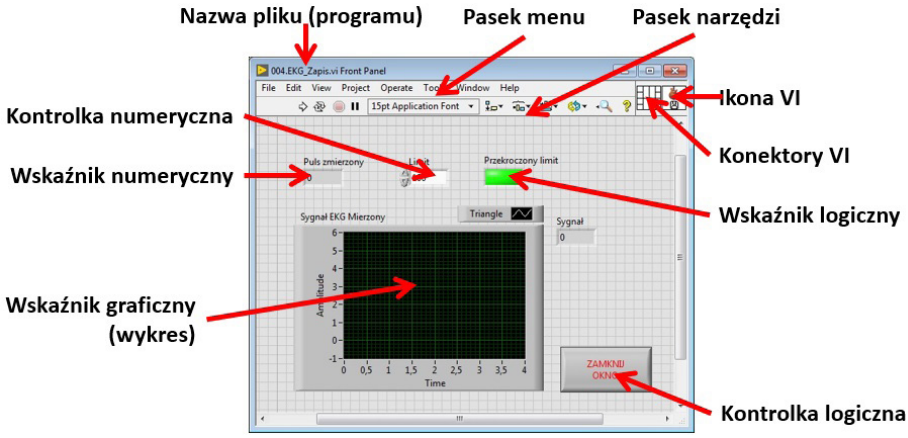
Każdy program opracowany w środowisku LabVIEW składa się z dwóch okien:

- okna interfejsu użytkownika (Front Panelu, panelu czołowego), w którym zamieszcza się elementy interfejsu użytkownika w postaci kontrolek i wskaźników oraz elementów graficznych i opisów (rys. 2.6),
- okna z algorytmem funkcjonowania programu (Block Diagramu), w którym zamieszcza się graficzny kod źródłowy programu w postaci węzłów programowych, odnośników do kontrolek i wskaźników oraz połączeń (nitek powiązań) (rys. 2.8).

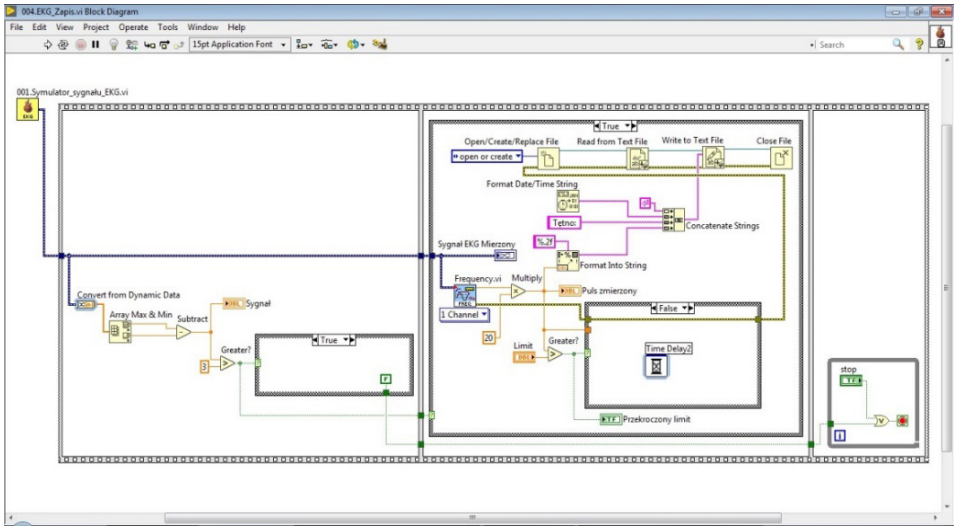


Rys. 2.6. Okno interfejsu użytkownika z wprowadzonymi elementami zarządzającymi funkcjonowaniem programu

Na każdym oknie front panelu można wyróżnić charakterystyczne elementy służące do obsługi programu oraz do zarządzania funkcjonowaniem tworzonej aplikacji. Zostały one przedstawione na rysunku 2.7.

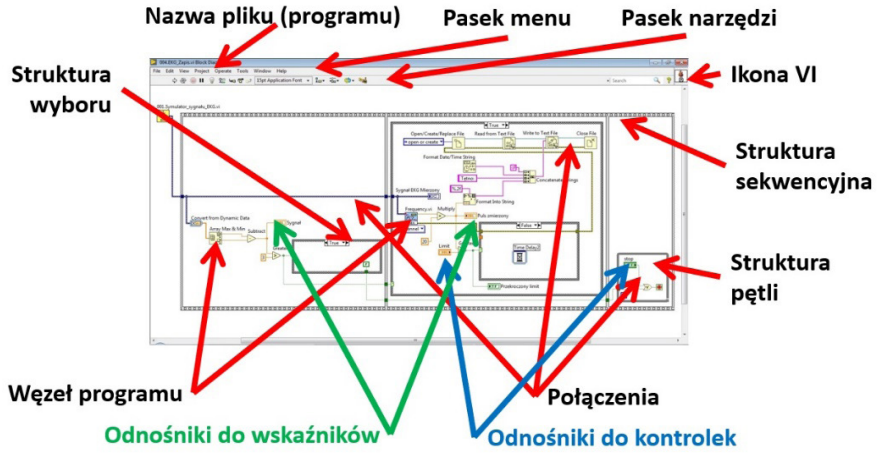


Rys. 2.7. Okno interfejsu użytkownika z zaznaczonymi charakterystycznymi elementami związanymi z edycją, funkcjonowaniem i zarządzaniem programem



Rys. 2.8. Okno schematu blokowego z umieszczonymi elementami kodu źródłowego

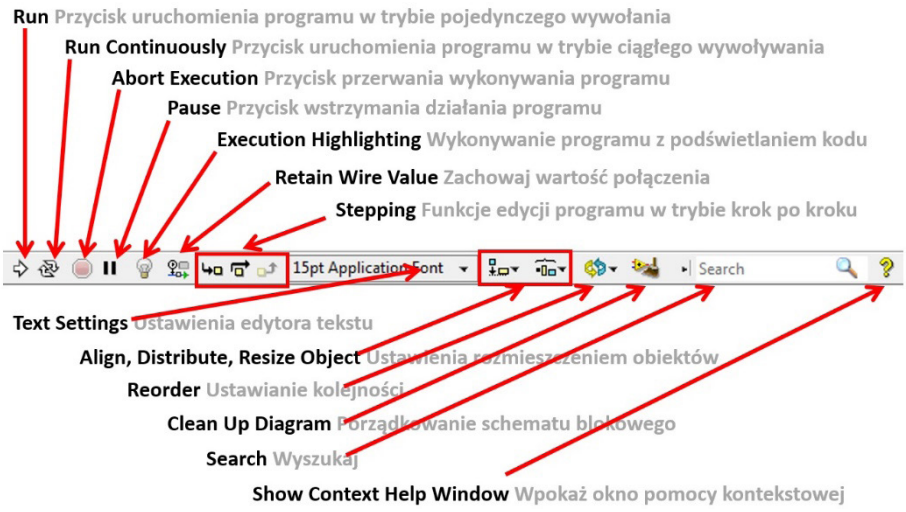
Podobnie, wyświetlając okno schematu blokowego, można w nim wyróżnić charakterystyczne elementy umożliwiające zarządzanie programem, funkcjonowaniem i obsługą aplikacji oraz obiekty stanowiące kod źródłowy programu. Opis charakterystycznych elementów okna schematu blokowego został przedstawiony na rysunku 2.9.



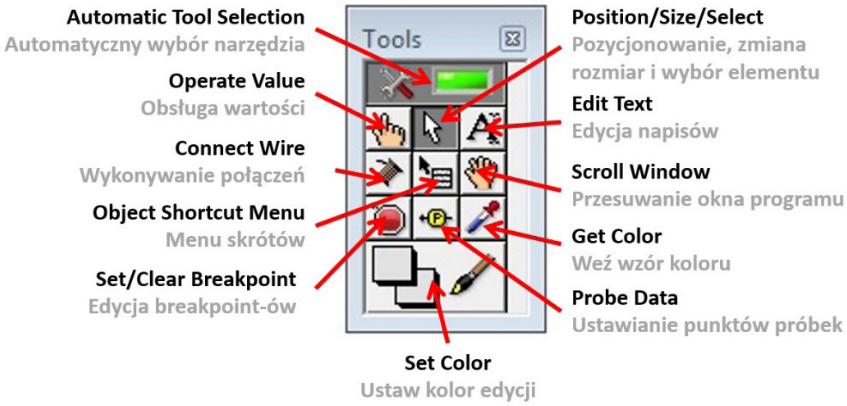
Rys. 2.9. Elementy charakterystyczne w oknie schematu blokowego programu

2.2.3. Zarządzanie i tworzenie aplikacji VI w środowisku LabVIEW

Zarządzanie procesem tworzenia aplikacji, wstawiania elementów i wykonywania określonych czynności związane jest z użyciem dostępnych narzędzi i funkcji. Najważniejsze funkcje sterujące znajdują się w pasku narzędzi (rys. 2.10) oraz w palecie narzędzi (rys. 2.11). Pasek narzędzi i paleta narzędzi dostępne są w obu oknach programu: (na panelu czołowym i na schemacie blokowym). Za pomocą określonych narzędzi można realizować proces edycji i modyfikacji programu oraz zarządzać ustawieniami elementów front panelu i schematu blokowego [1, 5, 17, 19, 20].

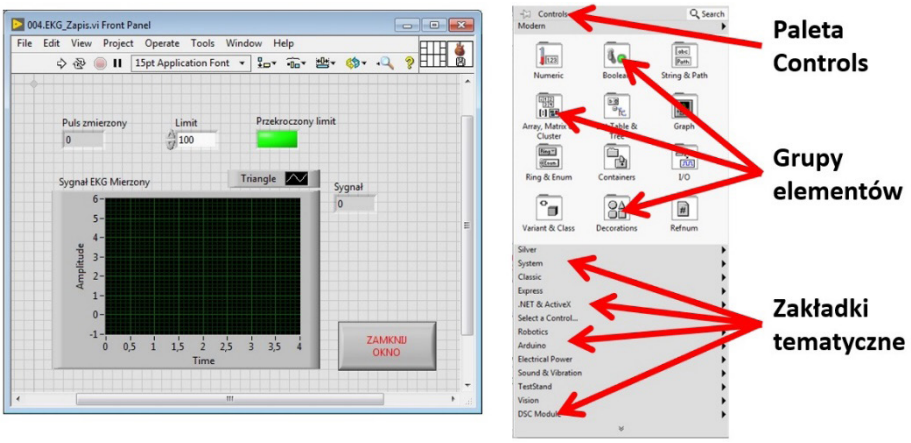


Rys. 2.10. Pasek narzędzi w środowisku LabVIEW



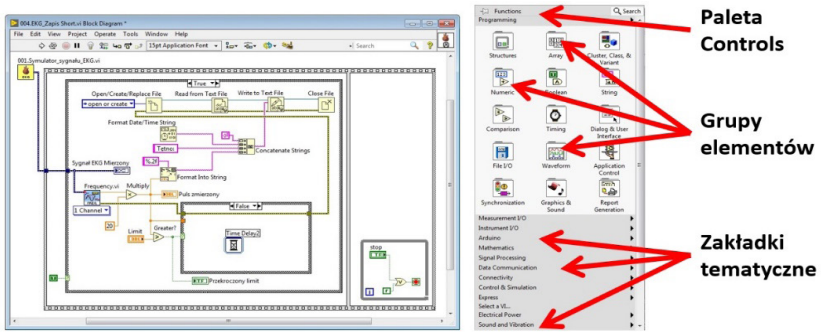
Rys. 2.11. Paleta Tool środowiska LabVIEW

W oknie front panelu umieszcza się elementy z palety Controls (rys. 2.12), które służą do zarządzania programem podczas jego działania. W zależności od potrzeb są to elementy, takie jak kontrolka i wskaźnik, które mogą reprezentować różne typy danych, np. liczbowe, logiczne, tekstowe [1, 5, 19, 20].



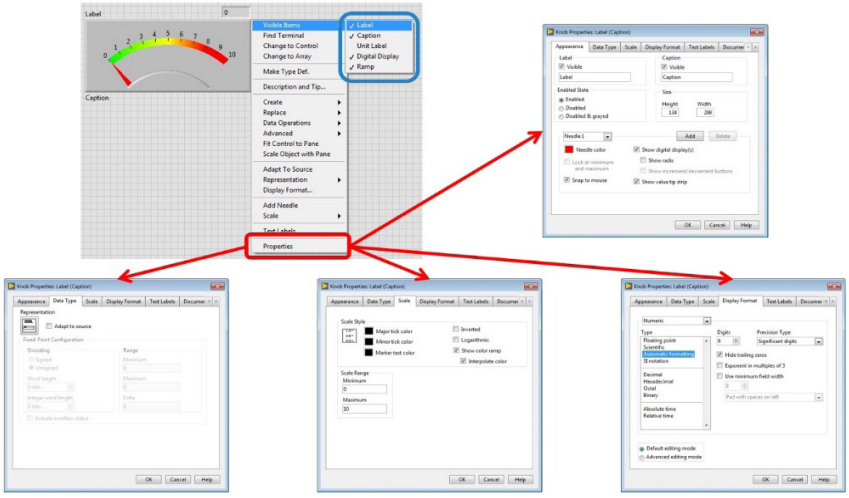
Rys. 2.12. Front panel i paleta Controls z wyszczególnionymi zakładkami tematycznymi i grupami elementów

W oknie schematu blokowego umieszcza się elementy z palety Functions (rys. 2.13), które służą do przetwarzania danych. Moduły funkcyjne mogą pobierać informacje różnego typu. Elementy na schemacie blokowym (węzły programowe) mają wejścia i wyjścia. Na schemacie blokowym oprócz węzłów umieszcza się również nitki pokazujące przepływ danych. Kolor nitek odpowiada typowi danych. Łączenie modułów funkcyjnych odbywa się za pomocą „nitek" programu. Nitki bez węzła mogą się rozgałęziać, ale nigdy łączyć.



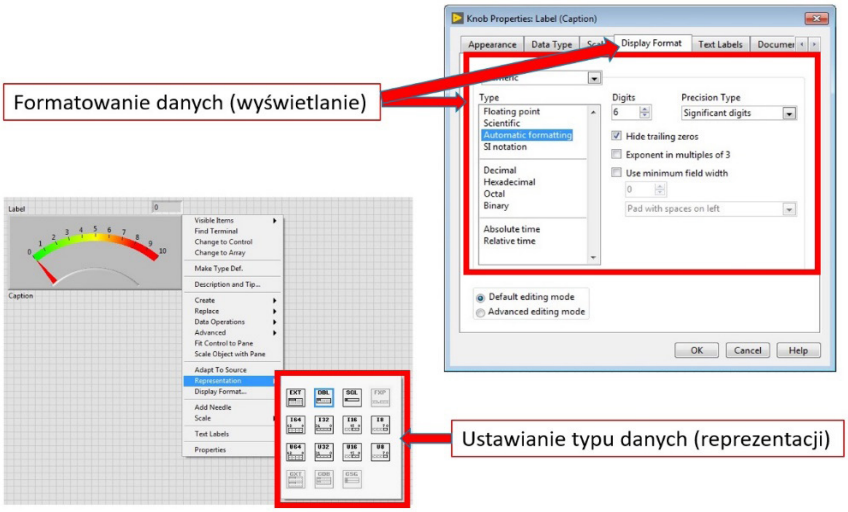
Rys. 2.13. Schemat blokowy i paleta Functions z wyszczególnionymi zakładkami tematycznymi i grupami elementów

Ważnym elementem tworzenia odpowiedniej szaty graficznej poszczególnych obiektów oraz dostosowania sposobu wprowadzania i wyświetlania danych do potrzeb użytkownika, jest stosowne do sytuacji zarządzanie właściwościami. W celu dostosowania wybranych właściwości, należy używać funkcji Properties dostępnej w oknie menu kontekstowego. Sposób wywołania funkcji Properties oraz przykładowe okna dialogowe służące do konfigurowania wskaźnika liczbowego typu Meter zostały przedstawione na rysunku 2.14.



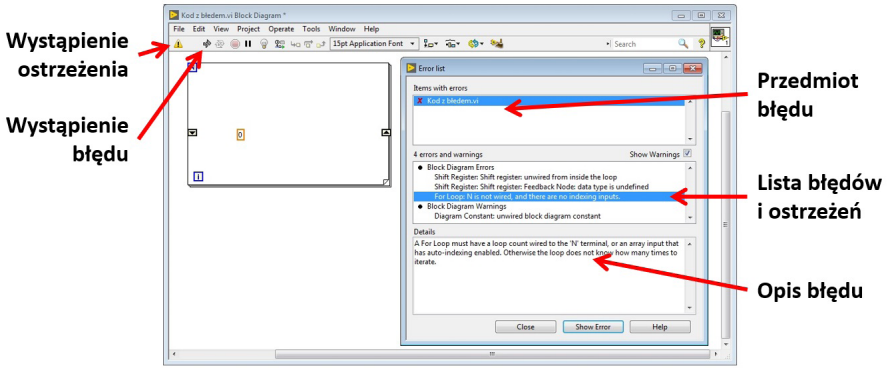
Rys. 2.14. Zarządzanie właściwościami obiektów

Równie ważnym działaniem, zmierzającym do odpowiedniej prezentacji danych, jest określenie i ustawienie typu danych oraz sposobu ich formatowania (rys. 2.15). Dzięki właściwemu wyborowi typu danych i sposobu wyświetlania można ograniczyć rozmiar koniecznej pamięci do obsługi danych.



Rys. 2.15. Ustawienia i funkcje związane z wybranym elementem dostępne poprzez menu kontekstowe

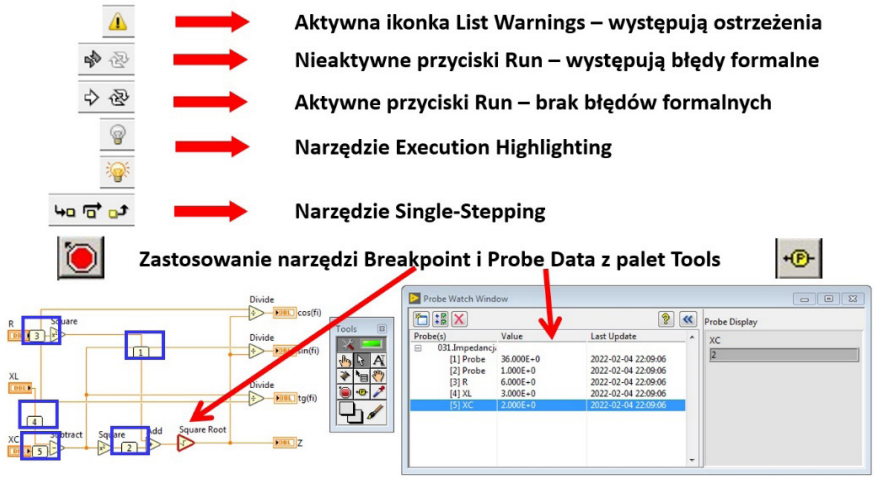
Do sprawdzania poprawności funkcjonowania opracowywanej aplikacji można użyć jednego z kilku dostępnych narzędzi. Najważniejszym z nich jest Error list. Dzięki niemu można zweryfikować, czy w programie nie występują tzw. błędy formalne. Po odpowiednim skonfigurowaniu Error list możliwe jest również wyświetlanie ostrzeżeń. W przypadku wystąpienia w programie tylko ostrzeżenia (warning) program może być uruchomiony, zgłosi on jedynie uwagi co do ewentualnych nieprawidłowości. Natomiast w przypadku wystąpienia w programie błędu (error) program nie będzie mógł się uruchomić do momentu usunięcia wszystkich zgłaszanych błędów. Okno dialogowe Error list oraz przykładowe informacje generowane w przypadku wystąpienia błędu przedstawiono na rysunku 2.16. Kliknięcie w konkretny błąd naprowadza na miejsce jego wystąpienia w określonym miejscu schematu blokowego.



Rys. 2.16. Okno Error list i jego konstrukcja w środowisku LabVIEW

Inne narzędzia przydatne w procesie kontroli funkcjonowania programu (patrz rys. 2.17) to:

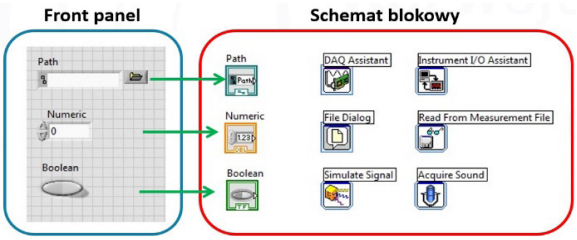
- narzędzie Execution Hightlighting,
- narzędzie Single-Stepping,
- narzędzie Breakpoint,
- narzędzie Probe Data.



Rys. 2.17. Narzędzia do testowania i śledzenia funkcjonowania programu VI

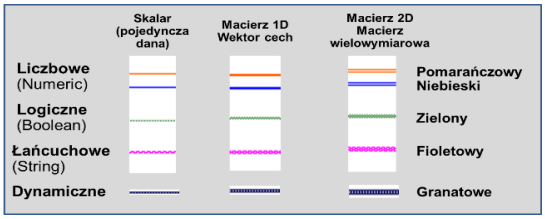
2.2.4. Zasady tworzenia wirtualnych narzędzi

Budowa wirtualnych narzędzi (programów VI) to proces składający się z dwóch części: tworzenia front panelu i tworzenia schematu blokowego programu. Budowa front panelu to uzupełnienie okna niezbędnymi elementami i ich skonfigurowanie według potrzeb użytkownika. Za pomocą elementów umieszczonych na front panelu, użytkownik będzie wprowadzał dane do programu (kontrolki) i otrzymywał wyniki (wskaźniki). Każdy wprowadzony na front panel element ma swój indywidualny terminal na schemacie blokowym. Wszystkie określone i zdefiniowane w środowisku LabVIEW typy kontrolek i wskaźników dostępne są w palecie Controls. Drugi etap to tworzenie schematu blokowego programu. Schemat blokowy programu stanowi kod źródłowy aplikacji przedstawiony w postaci graficznej. W oknie schematu blokowego do terminali kontrolek i wskaźników wstawia się moduły funkcyjne służące do przetwarzania dostarczonych danych. Na rysunku 2.18 przedstawione zostały okna front panelu oraz schematu blokowego z umieszczonymi elementami. Na front panelu są to: kontrolka ścieżki pliku (Patch Control), kontrolka numeryczna (Numeric Control) i kontrolka stanu (Boolean Control). Na schemacie blokowym widać terminale tych elementów oraz ikony sześciu modułów funkcyjnych.



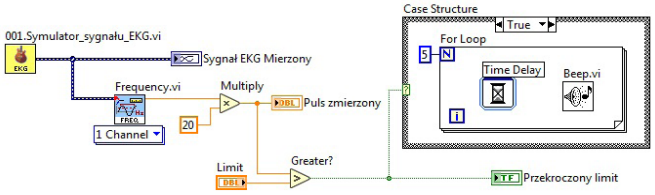
Rys. 2.18. Widok okna front panelu i schematu blokowego z zamieszczonymi przykładowymi elementami

Zamieszczane moduły funkcyjne służą do realizacji zadań związanych z przetworzeniem danych wejściowych i otrzymaniem wyników. Elementy przetwarzające dane w środowisku LabVIEW to węzły. Uzupełnieniem schematu blokowego są struktury programistyczne (np. pętle, struktury czasowe, struktury wyboru, struktury sekwencyjne). Struktury programistyczne w środowisku LabVIEW to zestawy instrukcji sterujących pracą programu i realizacją kodu źródłowego w VI. Są one dostępne w palecie Functions w grupie Structures. Zarządzanie przepływem danych odbywa się w środowisku LabVIEW poprzez tworzenie połączeń między terminalami kontrolek i wskaźników a modułami funkcyjnymi wstawionymi do okna schematu blokowego programu. Połączenia te („nitki", „druty") mają określone kolory przyporządkowane zdefiniowanym typom danych. Podstawowe typy danych i reprezentujące je nitki (połączenia) przedstawia rysunek 2.19.



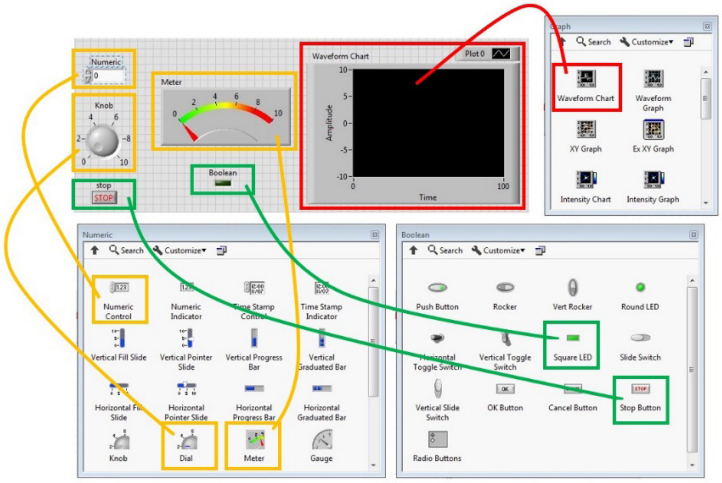
Rys. 2.19. Reprezentacja graficzna różnych typów danych na schemacie blokowym

Na rysunku 2.20 przedstawiono przykładowy schemat blokowy programu z zastosowaniem różnego typu węzłów, struktur programistycznych oraz danych.

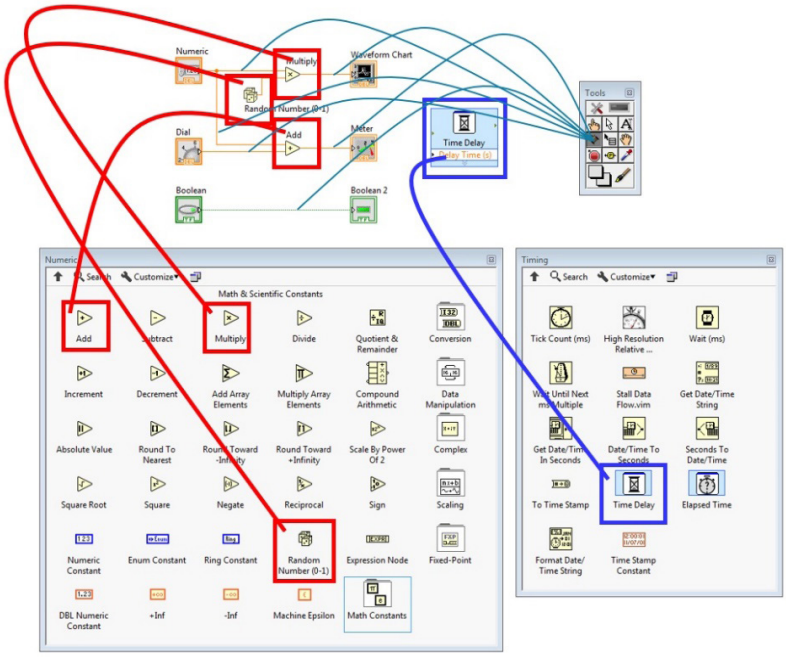


Rys. 2.20. Schemat blokowy przykładowego programu w środowisku LabVIEW

Uzupełnianie okna front panelu przez wybrane typy wskaźników i kontrolek zostało przestawione na rysunku 2.21. Uzupełnianie okna schematu blokowego przez wybrane moduły funkcyjne zostało przestawione na rysunku 2.22.



Rys. 2.21. Lokalizacja wybranych typów wskaźników i kontrolek



Rys. 2.22. Lokalizacja wybranych typów modułów funkcyjnych

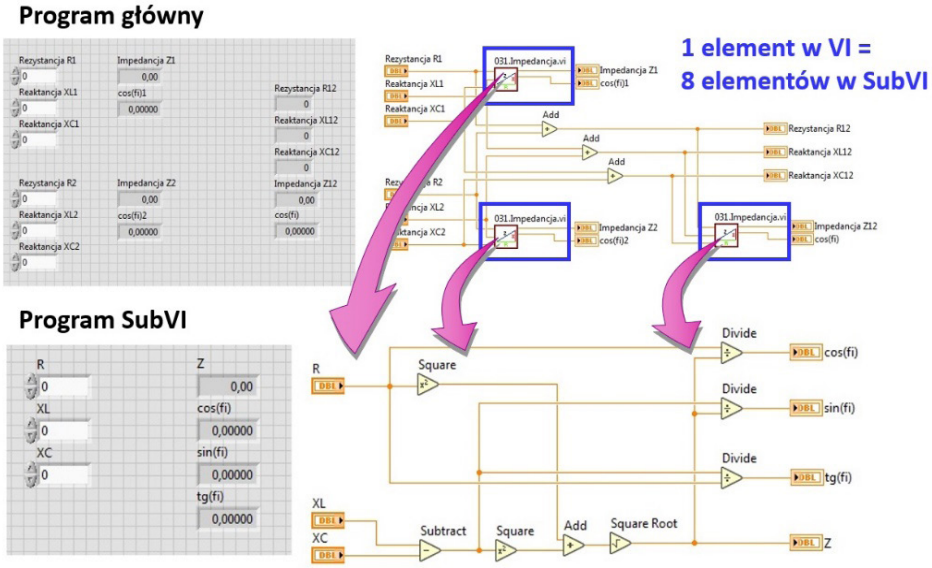
2.2.5. Tworzenie i wykorzystanie SubVI

Programowanie w środowisku LabVIEW z założenia ma być przejrzyste i intuicyjne. Na obie te cechy niewątpliwie ma wpływ liczba elementów i połączeń umieszczonych w kodzie źródłowym na schemacie blokowym. W przypadku rozległych programów, wymagających użycia wielu elementów, procedur i węzłów, część kodu można zapisać w postaci SubVI (podprogramu). Podobnie, wykonanie takiej czynność usprawiedliwia kod źródłowy, składający się z wielu powtórzeń tego samego kodu w obrębie jednego programu. Zastosowanie SubVI umożliwia również podział większego projektu na mniejsze moduły i podział na członków zespołu [1, 2, 20].

Wykonanie programu SubVI umożliwia także późniejszą jego implementację w kolejnych projektach, bez konieczności żmudnego „wyciągania" kodu programu z innych aplikacji i wpasowywania go do danego programu.

Ważnym aspektem wykorzystania SubVI jest fakt, że po wstawieniu go do programu głównego, SubVI ma wielkość standardowego węzła funkcyjnego.

Funkcjonowanie SubVI jako węzła funkcyjnego w programie przedstawiono na rysunku 2.23. Na zaprezentowanym schemacie widać, że w programie głównym użyto trzykrotnie tę samą część kodu (procedurę). Dzięki temu uzyskano większą przejrzystość programu. W przypadku wielkokrotnego zastosowania danej procedury w postaci SubVI, możliwa jest szybka identyfikacja całej procedury, a wykonanie poprawek odnosi się do wszystkich miejsc jej użycia. Standardowe podejście wymagałoby identyfikacji wszystkich miejsc, gdzie konieczne jest wstawienie poprawek.

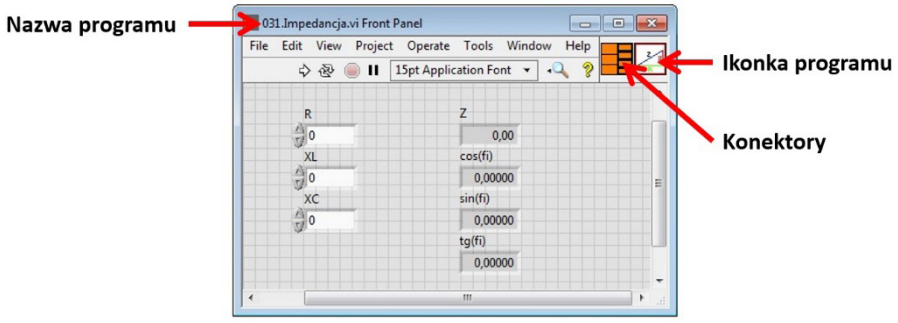


Rys. 2.23. Modularyzacja programu głównego w środowisku LabVIEW

Każdy program tworzony w środowisku LabVIEW, przewidziany do użycia jako procedura programowa w innym programie (SubVI), charakteryzuje się tym, że posiada swoją:

- nazwę pliku (najlepiej niepowtarzalną),
- ikonkę (najlepiej indywidualną),
- terminal wejść i wyjść (konektory).

Są one widoczne w oknach front panelu i schematu blokowego każdego programu (patrz rys. 2.24).



Rys. 2.24. Oznaczenia indywidualne programu w środowisku LabVIEW

Proces tworzenia pliku SubVI polega na:

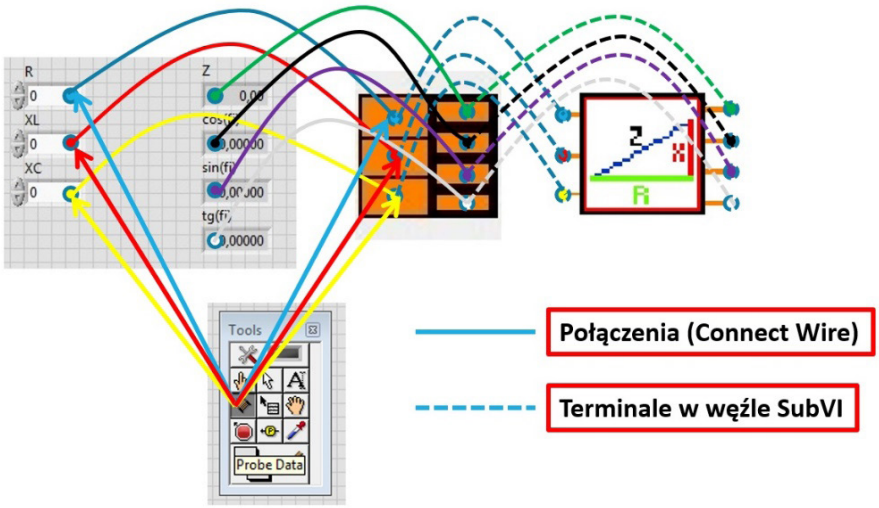
- wykonaniu realizującego założone cele programu. Ważne jest, aby program nie zawierał błędów formalnych, bo wtedy jako SubVI będzie dysfunkcyjny,
- zapisaniu programu w pliku VI na dysku,
- zdefiniowaniu terminali wejścia i wyjścia,
- wyedytowaniu indywidualnej ikony dla SubVI.

Dwie pierwsze czynności są standardowe w przypadku tworzenia każdego programu. Dwie pozostałe odnoszą się do wymagań i idei tworzenia kodu źródłowego danego programu VI. Konkretny element wstawiany do schematu blokowego ma swoją indywidualną reprezentację graficzną. Dzięki temu jest łatwo rozpoznawalny i kojarzony ze ściśle określoną czynnością. Każdy element służący do przetwarzania danych ma również zdefiniowane i określone poszczególne terminale, do których należy podłączyć sygnały (nitki) w schemacie blokowym. Dzięki temu możliwe jest użycie SubVI jako węzła przetwarzającego określone dane.

Proces definiowania konektorów (terminali wejść i wyjść) polega na:

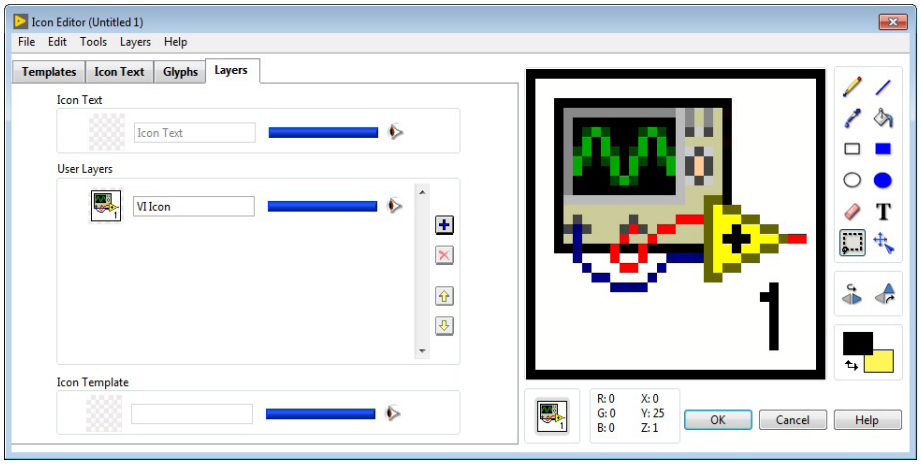
- wybraniu odpowiedniego wzoru (Pattern),
- wybraniu w palecie Tools narzędzia do tworzenia połączeń (Connect Wire),
- wykonania połączeń między kontrolkami i wskaźnikami a polami terminala.

Prawidłowe wykonanie tych czynności (patrz rys. 2.25) zmienia puste pola w terminalu wejść i wyjść na pola kolorowe, których kolory odpowiadają typom danych.

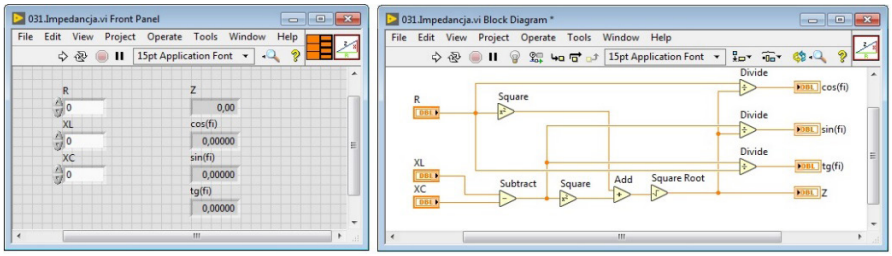


Rys. 2.25. Proces tworzenia połączeń terminali wejść i wyjść

Edycja indywidualnej ikony dla SubVI daje duże możliwości rozpoznawalności programu oraz szybkiej identyfikacji SubVI w kodzie źródłowym programu. Edycję ikony realizuje się w oknie Icon Editor w prawym górnym rogu okien front panelu i schematu blokowego. Okno edytora ikon (rys. 2.26) daje możliwość wstawienia własnej ikony według indywiduwalnego pomysłu (rys. 2.27) (proste narzędzia edycyjne w oknie edytora ikon), wstawienia przez importowanie grafiki (ikona przygotowana w innym programie), użycia wzorców w jednej z zakładek: szablon (Templates), grafik i rysunków (Glyphs) lub warstw (Layers).



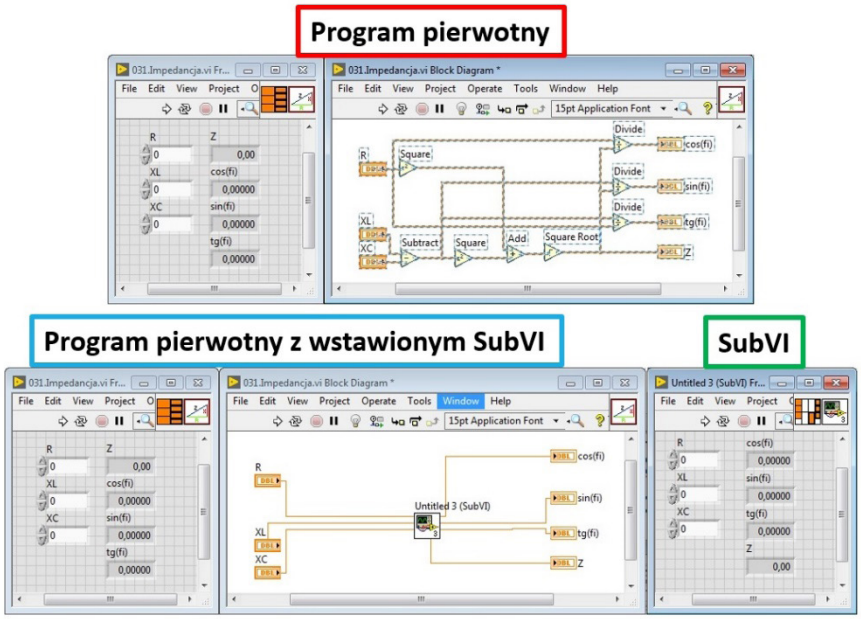
Rys. 2.26. Okno edytora ikon programu w środowisku LabVIEW



Rys. 2.27. W pełni przygotowany SubVI, posiadający swoją nazwę, ikonę oraz zdefiniowane terminale wejścia i wyjścia

Możliwe jest również przygotowanie SubVI z całości kodu bieżącego programu w sposób automatyczny (rys. 2.28). W tym celu należy:

- opracować program, który ma być wyodrębniony jako kod programu i zadeklarowany jako plik SubVI,
- zadeklarować nazwy elementów wejścia i wyjścia zgodnie z oczekiwanymi nazwami docelowymi,
- zaznaczyć cały kod źródłowy programu w oknie schematu blokowego,
- użyć polecenia Create SubVI z zakładki Edit.



Rys. 2.28. Proces tworzenia SubVI bezpośrednio z kodu programu

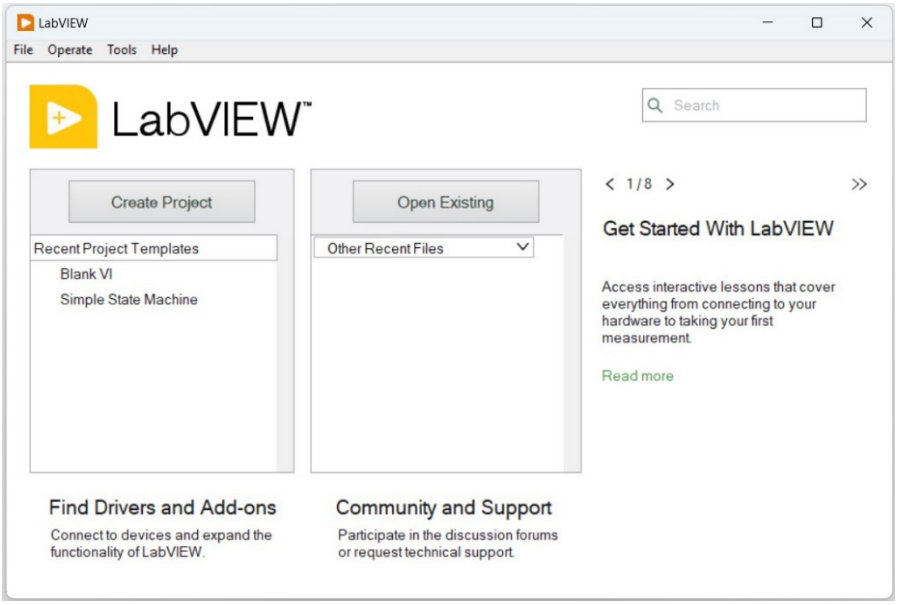
W analogiczny sposób można tworzyć SubVI zawierające tylko część programu głównego. W tym przypadku należy zaznaczyć wybraną do scalenia część kodu źródłowego programu.

2.3. Zadania

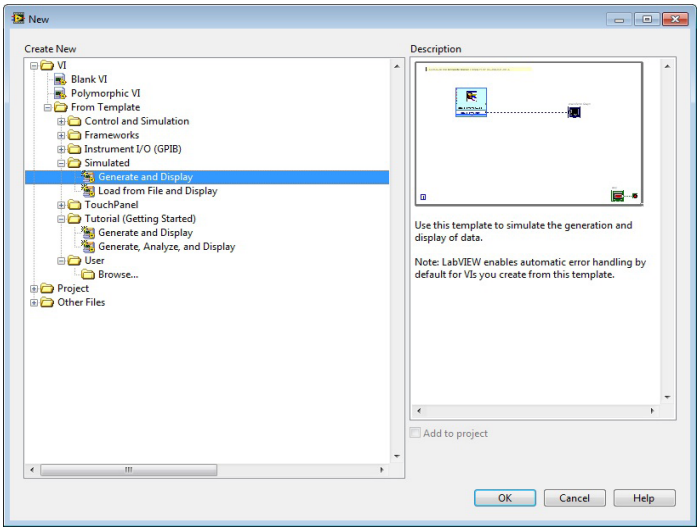
2.3.1. Przyrząd wirtualny do akwizycji danych. Użycie gotowych szablonów

Cel zadania:	zapoznanie ze środowiskiem LabVIEW przez zbudowanie (na podstawie gotowego szablonu) przyrządu wirtualnego (VI), który generuje sygnał i pokazuje go na panelu czołowym.
Zakres zadania:	wykorzystanie gotowego wzorca programu i utworzenie na jego bazie nowego programu, którego zadaniem będzie prezentacja wykresów o zadeklarowanych na wyświetlaczu graficznym parametrach.

1. **Uruchomić środowisko LabVIEW.** W oknie startowym w pasku menu z zakładki File wybrać opcję [New...].



2. W wyświetlonym oknie dialogowym New należy odszukać szablon programu o nazwie Generate and Display. Jego przypuszczalna lokalizacja jest zgodna z lokalizacją przedstawioną na rysunku. Należy zwrócić uwagę, że w oknie [Description] przedstawione są informacje dotyczące wybranego szablonu przyrządu.



3. Wybór szablonu przyrządu zatwierdzić, wybierając przycisk [OK] lub dwukrotnie klikając na nazwę wybranego przyrządu w oknie Create new. Po zatwierdzeniu szablonu ukaże się interfejs przedstawiający panel czołowy.

4. **Zapoznanie z panelem czołowym (okno Front Panel).** Wyświetlony interfejs użytkownika powinien zawierać standardowe okno wykresu Weveform Graph i przycisk [STOP]. Jeżeli okno panelu czołowego nie jest widoczne, należy z menu Window wybrać polecenie Show Front Panel.

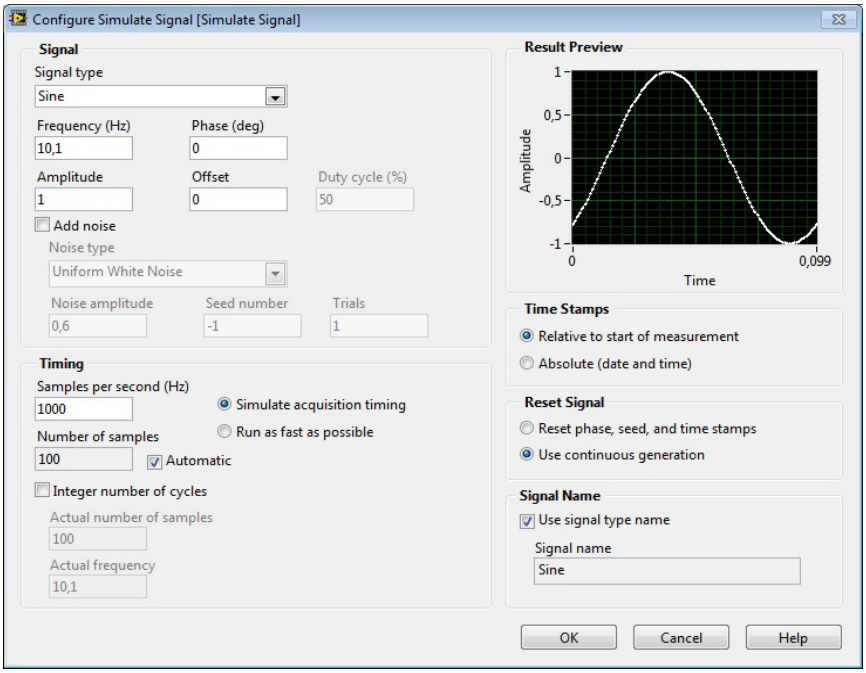
5. **Zapoznanie ze schematem blokowym (okno Block Diagram).** Wyświetlony na białym tle schemat blokowy przedstawia strukturę połączeń i zależności między obiektami umieszczonymi na panelu. Jeżeli okno schematu blokowego nie jest widoczne, należy z menu Window wybrać polecenie Show Block Diagram.
6. Symulację działania programu można sprawdzić, włączając funkcję [Run] z menu [Operate], wciskając kombinację klawiszy [Ctrl] + [R] lub wybierając przycisk  . Wyłączenie działania symulacji można wykonać za pomocą polecenia [Stop] z menu [Operate], kombinacji klawiszy [Ctrl] + [.], przycisku  lub umieszczonego na panelu przycisku 

.

Porada: przełączanie między panelem czołowym (front panel) a schematem blokowym (block diagram) wykonuje się za pomocą polecenia [Show Front Panel] i [Show Block Diagram] znajdujących się w menu Window. Podobnie za pomocą poleceń [Show Controls Palette] i [Show Functions Palette] wyświetla się w panelu czołowym (front panel) paletę kontrolek / regulatorów, a w schemacie blokowym paletę funkcji.

7. **Dodawanie wskaźników (Indicators) i kontrolek (Controls) do panelu czołowego.** Wskaźniki umieszczane na panelu umożliwiają symulowanie wejściowych wielkości fizycznych i dostarczanie danych wejściowych do przyrządów wirtualnych. Program daje możliwość implementowania wskaźników o różnych kształtach, wielkościach oraz o nieograniczonym zakresie wielkości wejściowych.
8. Sprawdzić, czy paleta regulatorów (Control) jest widoczna na panelu czołowym. Jeżeli nie jest, uaktywnić ją. Wyświetlenie palety Controls realizuje się poprzez włączenie funkcji Controls Palette z menu View lub ustawiając kursor w oknie panelu czołowego i klikając prawym klawiszem myszy.
9. W palecie regulatorów (Control Palette) odszukać opcję [Numeric Controls] (regulatory liczbowe) i zatwierdzić wybór, klikając na ich ikonę.
10. Wybrać obiekt Knob (gałka), wskazując kursorem myszy ikonę opcji. Jeżeli tło nagłówka z nazwą opcji podświetli się na granatowo, oznacza to, że dana opcja została zatwierdzona i jest aktywna. Możliwe jest wtedy wprowadzenie danego regulatora/wskaźnika na panel czołowy. Umieszczanie wskaźnika odbywa się za pomocą myszki. Należy przesunąć kursor myszy na miejsce w polu panelu, gdzie ma się znajdować wstawiany wskaźnik i kliknąć przyciskiem myszy. W dalszej części ćwiczenia gałka będzie używana do regulacji amplitudy sygnału.
11. Usuwanie umieszczonego wcześniej wskaźnika odbywa się po uprzednim zaznaczeniu danego elementu za pomocą polecenia [Cut] (wytnij) z menu [Edit], kombinacji klawiszy [Ctrl] + [x] lub za pomocą klawisza [Delete].
12. Zapisać wynik pracy pod nazwą 021.AkwizycjaSygnalu.vi w lokalizacji wskazanej przez prowadzącego.

13. **Zmiana typu sygnału.** W celu dostosowania ustawień typu sygnału do tworzonego projektu należy uaktywnić okno schematu blokowego. W tym oknie znajdują się informacje dotyczące wewnętrznych ustawień i połączeń między elementami widocznymi na panelu czołowym.
14. Sprawdzić, czy widoczne jest okno schematu blokowego – jeżeli nie, to należy je uaktywnić.
15. W celu uaktywnienia okna dialogowego, umożliwiającego dokonywanie zmiany ustawień sygnału, należy wybrać wskaźnikiem myszy pole symulatora sygnału (Simulate Signal) i kliknąć dwukrotnie jego obszar lub, klikając prawym przyciskiem myszy, otworzyć menu kontekstowe i wybrać Właściwości (Properties). Wykonanie tej czynności powinno spowodować wyświetlenie okna konfiguracji symulowanego sygnału (Configure Simulate Signal) tożsamego z przedstawionym na rysunku.



16. W polu Signal type (typ sygnału) konfigurowany jest kształt sygnału. Dostępne są następujące typy: Sine (sinusoidalny), Square (prostokątny), Triangle (trójkątny), Sawtooth (piłokształtny) i DC (stały). Należy zadeklarować opcję sygnału piłokształtnego. Wprowadzoną zmianę zatwierdzić przyciskiem [OK].
17. W celu poszerzenia liczby konektorów obiektu Simulate Signal należy umieścić wskaźnik myszy nad dolną krawędzią obiektu. Wskaźnik myszy zostanie zastąpiony symbolem dwóch strzałek skierowanych do góry i dołu. Kiedy wskaźnik znajdzie się we wskazanym miejscu, należy wcisnąć przycisk myszy i przesunąć ją w dół. W celu ograniczenia liczby konektorów sygnałów wejściowych i wyjściowych, należy wykonać te czynności, kierując wskaźnik myszy do góry. Poszerzyć ikonę obiektu Simulate Signal tak, aby widoczne było wejście Amplitude.
18. Nieopisane typy sygnałów wejściowych po zwinięciu menu obiektu są przedstawiane na obszarze obiektu (np. przyrządu wirtualnego) w postaci kolorowych strzałek. Kolor strzałki określa typ sygnału, jaki może być przyłączony do danego wejścia lub typ sygnału, jaki jest dostępny z danego wyjścia.

19. **Tworzenie ścieżek połączeń (wiring) między obiektami schematu blokowego.** Postępowanie według poniższych wskazówek pozwoli na połączenie regulatora gałki potencjometru (Knob) z wejściem Amplitude (amplituda) w obiekcie Simulate Signal. Dzięki temu możliwe będą zmiany wartości amplitudy generowanego sygnału na wykresie.

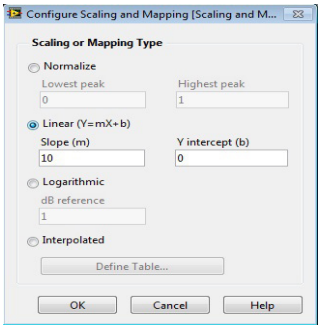
Na wstępie należy się upewnić, czy aktywna jest funkcja Automatic Tool Selection w oknie narzędziowym Tools. Okno narzędziowe Tools wywołuje się z menu View poleceniem Tools Palette.

20. Umieścić kursor myszy w obszarze ikonki gałki (Knob). Kursor myszy zmieni kształt na narzędzie do przesuwania (pozycjonowania).
21. Przenieść ikonę gałki do wewnątrz pętli (szary prostokąt z zaokrąglonymi rogami).
22. Usunąć zaznaczenie gałki przez kliknięcie na wolnym obszarze schematu blokowego.
23. Umieścić kursor myszy nad terminalem wyjściowym ikony gałki (Knob). Kursor myszy zmieni kształt na narzędzie do tworzenia połączeń (wiring tool).
24. Przyciskając lewy przycisk myszy wykonać połączenie terminala wyjściowego ikony gałki (Knob), z terminalem wejściowym obiektu symulatora sygnału (Simulate Signal) (za przemieszczającym się po ekranie kursorem myszy będzie rysowała się przerywana linia). Należy, wskaźnikiem myszy, wybrać na docelowe miejsce zakończenia połączenia i wcisnąć ponownie lewy klawisz myszy. Jeżeli połączenie wykonane będzie prawidłowo, to przerywana linia zamieni się w linię ciągłą oznaczoną kolorem danego typu sygnału. W przeciwnym razie linia będzie dalej przerywana oraz wyświetlony zostanie znak nieudanego połączenia. Usuwanie zbędnego lub nieprawidłowego połączenia wykonuje się za pomocą klawisza [Delete]. Po prawidłowym wykonaniu procedury wykonywania połączenia linia powinna mieć kolor pomarańczowy.
25. Zapisać w bieżącym pliku zmiany wykonane w programie (polecenie Save z menu File).

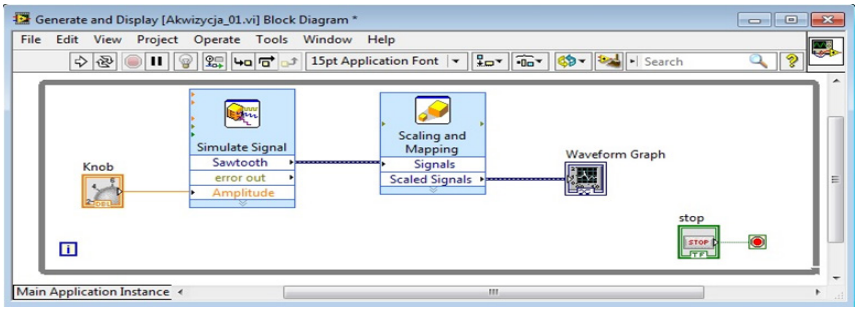
26. **Uruchamianie programu.** Postępowanie według wskazówek umożliwia uruchomienie i wyłączenie stworzonego programu oraz dokonywanie zmiany wielkości amplitudy generowanej na wyświetlaczu sygnału.
27. Upewnić się, czy na ekranie monitora widoczny jest panel czołowy programu, jeżeli nie, uaktywnić go.
28. Uruchomić program dowolną z metod tj. za pomocą polecenia Run z menu Operate, wciskając kombinację klawiszy [Ctrl] + [R] lub wybierając przycisk  .
29. Wskazać kursorem myszy obiekt potencjometru (Knob).
30. Zmiany ustawienia gałki potencjometru uzyskuje się podtrzymując wciśnięty lewy przycisk myszy i wykonując ruchy okrężne wewnątrz obszaru gałki. Dokonując zmian ustawień położenia gałki potencjometru (Knob), należy zwrócić uwagę na zmiany zachodzące na wyświetlanym wykresie generowanego sygnału.
31. Wyłączyć program dowolną z metod tj. za pomocą polecenia Stop z menu Operate, kombinacją klawiszy [Ctrl] + [.] lub umieszczonym na panelu przyciskiem 

.

32. **Modyfikowanie sygnału.** W wielu przypadkach zachodzi konieczność zmiany lub modyfikacji różnych parametrów. W tym celu używane są kolejne bloki funkcyjne umożliwiające dokonanie koniecznych poprawek pozwalających na dostosowanie parametrów sygnałów do potrzeb. Postępowanie według podanych wskazówek spowoduje wprowadzenie do programu przeskalowanego przebiegu i wyświetlenie go na wykresie.
33. W oknie schematu blokowego wybrać (zaznaczyć) połączenie obiektu symulatora sygnału (Simulate Signal) z wykresem Waveform Graph.
34. Kiedy całe połączenie zostanie zaznaczone przerywaną linią, należy wybrać klawisz [Delete].
35. Jeśli nie jest widoczna paleta funkcji (Functions), wybrać z menu View polecenie Functions Palette.
36. Wstawić do wnętrza pętli (między symulator sygnału (Simulate Signal) a wykres Waveform Graph) obiekt Scaling and Mapping (skalowanie i odwzorowywanie), znajdujący się w palecie Functions w zakładce Express w grupie Arithmetic and Comparasion. Po wstawieniu obiektu automatycznie powinno otworzyć się okno dialogowe umożliwiające zmianę jego ustawień.

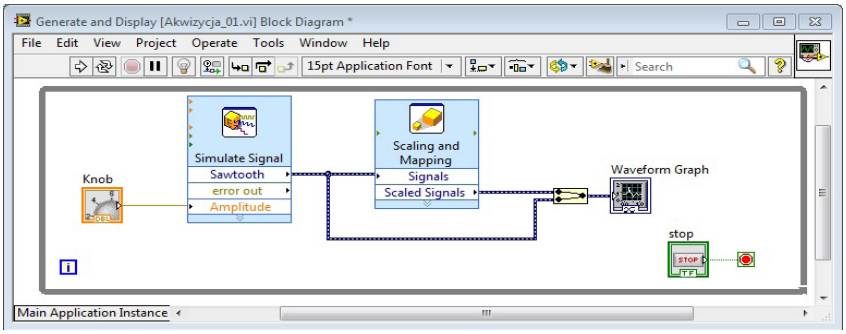


37. Jeżeli okno konfiguracji obiektu Scaling and Mapping (patrz rysunek) nie zostanie automatycznie otworzone, należy je uaktywnić (np. poleceniem Properties z menu kontekstowego). Jeśli nie jest aktywne, to zaznaczyć opcję [Linear (Y = mX + b)] włączającą liniową modyfikację sygnału wyjściowego. Następnie ustawić parametr [Slope (m)] (nachylenie), wpisując w oknie wartość 10. Zatwierdzić zmiany, wciskając przycisk [OK].
38. Wykonać niezbędne połączenia między obiektami Scaling and Mapping (skalowanie i odwzorowywanie), Simulate Signal i Weveform Graph, tak jak zostało to przedstawione na rysunku. W tym celu wstawić połączenie wyjścia Sawtooth (piłokształtny) obiektu symulatora sygnału (Simulate Signal) z wejściem Signals (sygnał) w obiekcie Scaling and Mapping oraz połączenie wyjścia Scaled Signal (sygnał przeskalowany) obiektu Scaling and Mapping z obiektem wykresu Waveform Graph.



39. Zapisać wykonane w programie zmiany w pliku pod bieżącą nazwą. Funkcja Save z zakładki File w pasku menu.

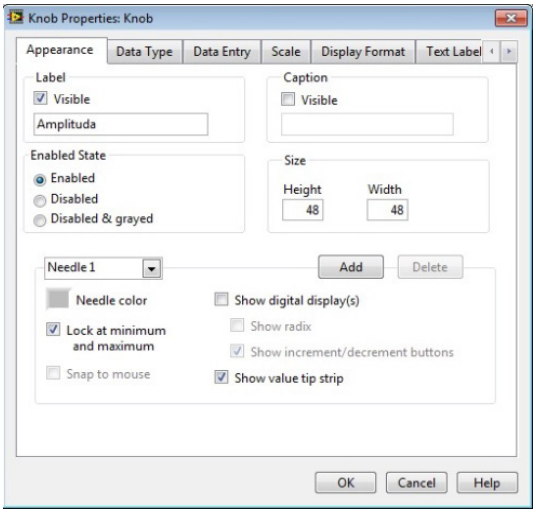
40. **Prezentacja dwóch sygnałów na wykresie.** Niekiedy występuje konieczność połączenia dwóch (lub więcej) sygnałów, np. w celu równoczesnego wyświetlenia czy obróbki. Do realizacji tego zadania jest wykorzystywana funkcja łączenia sygnałów (Merge Signals function). Postępowanie według podanych wskazówek (krok po kroku) umożliwia wygenerowanie na wyświetlaczu graficznym dwóch sygnałów. Jeden generowany sygnał przedstawia przeskalowany, a drugi nieprzeskalowany przez obiekt Scaling and Mapping (skalowanie i odwzorowywanie), sygnał wychodzący z obiektu Simulate Signal.
41. Upewnić się, czy na ekranie monitora widoczne jest okno schematu blokowego. Jeżeli nie jest widoczne, uaktywnić je.
42. Wykonać połączenie między wyjściem Sawtooth (piłokształtny) obiektu Simulate Signal (symulator sygnału) a wykresem Waveform Graph, tak jak na rysunku. Po wprowadzeniu tego połączenia w programie wygenerowana zostanie funkcja łączenia sygnałów (Merge Signals), umożliwiająca wyświetlanie dwóch (lub więcej) sygnałów na jednym wyświetlaczu graficznym.



43. Zapisać wykonane w programie zmiany w pliku pod bieżącą nazwą (polecenie Save z menu File).

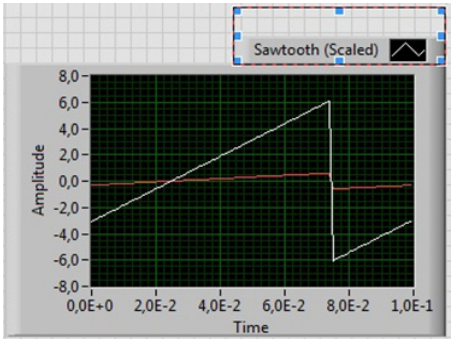
44. Uaktywnić okno panelu czołowego i uruchomić proces symulacji działania programu. Jeżeli wszystkie dotychczasowe instrukcje zostały wykonane prawidłowo, na wykresie wyświetlacza graficznego powinny ukazać się dwa przebiegi (biały i czerwony).
45. Zakończyć proces symulacji (przycisk Stop).

46. **Dostosowywanie obiektu Knob.** Obiekt Knob (gałka potencjometru) reguluje zmiany amplitudy generowanego sygnału. Postępowanie według wskazówek umożliwia przystosowanie obiektu Knob (gałka potencjometru) na panelu czołowym do potrzeb użytkownika.
47. Wybrać obiekt gałka potencjometru prawym przyciskiem myszy. Z wyświetlonej listy wybrać polecenie Properties (Właściwości).
48. Na karcie Appearance (wygląd) zmienić nazwę etykietki obiektu (Label) z Knob na Amplituda (patrz rysunek).
49. Następnie na karcie Scale za pomocą funkcji Scale Range (zakres skali), znajdującej się w karcie Scale, ustaw maksimum zakres regulacji na wartość 5,0.
50. Wprowadzone zmiany zatwierdzić przyciskiem [OK].



51. Po dokonaniu tych zmian zapisać aktualną wersję pliku programu na dysku pod bieżącą nazwą.
52. W celu poznania funkcji dostępnych w oknie Properties obiektu Knob należy dokonać próbnych zmian w wyglądzie obiektu. W tym celu należy użyć dostępnych funkcji, jednocześnie obserwując wprowadzane zmiany na ekranie.
53. Po dokonaniu testowych zmian należy zamknąć okno właściwości, wybierając przycisk [Cancel] (Anuluj), co uniemożliwi wprowadzenie zmian w opracowywanym programie.

54. **Modyfikowanie wykresu Waveform Graph.** Wykres Waveform Graph wyświetla aktualne dane o dwóch wielkościach. W opcjach właściwości możliwe jest dostosowanie wyglądu poszczególnych elementów wykresu, jak i całego obiektu, do potrzeb użytkownika. Postępowanie według podanych wskazówek pozwala na wyświetlenie informacji o obu wyświetlanych sygnałach oraz dokonanie zmiany niektórych własności wyświetlanych sygnałów.
55. Dostosowanie ustawień obiektu Waveform Graph wykonuje się w oknie panelu czołowego. Przesunąć wskaźnik myszki nad obszar legendy obiektu. Kiedy kursor myszki będzie nad tym obszarem, uaktywni się procedura obramowania obszaru. Należy wskazać kursorem na środek górnej krawędzi legendy obiektu (pojawi się wtedy strzałka dwustronna).
56. Nacisnąć lewy przycisk myszki i przesunąć wskaźnik, aż pojawi się informacja o drugim przebiegu. Wykonanie tej instrukcji przedstawiono na rysunku.



57. Wybrać obiekt Waveform Graph prawym przyciskiem myszy. Z wyświetlonej listy wybrać Properties (Właściwości). W ten sposób uaktywnione i wyświetlone na ekranie zostanie okno właściwości obiektu.
58. W karcie Plots znajduje się rozwijane menu z nazwami poszczególnych przebiegów. W celu zmiany nazwy należy wybrać nazwę Sawtooth (scaled) i uaktywnić ją, wciskając lewy przycisk myszy. Następnie uaktywnić opcję Do not use waveform names for plot names. W celu zmiany wyświetlanej etykietki wybranego sygnału należy wpisać własną nazwę sygnału, np. „Skalowana Piła”. Wprowadzone zmiany zatwierdzić, wciskając przycisk [OK], co dodatkowo spowoduje zamknięcie okna Properties.
59. W celu lepszego poznania funkcji własności obiektu należy sprawdzić, jakie zmiany zajdą po wyłączeniu funkcji Autoscale (skalowanie automatyczne). Funkcja ta znajduje się w zakładce Scales (skale). Po dokonaniu przykładowych zmian należy zamknąć okno właściwości, naciskając przycisk [Cancel] (Anuluj), co uniemożliwi wprowadzenie zmian w projekcie.
60. Po dokonaniu tych zmian zapisać plik pod nazwą 021_GenerateDisplay.vi, a następnie zamknąć go.

2.3.2. Symulator generatora sygnału diagnostycznego. Przyrząd wirtualny redukujący liczbę próbek

Cel zadania: tworzenie nowego programu od podstaw. Dodawanie elementów kodu źródłowego umożliwiających budowę wirtualnego przyrządu symulującego sygnał o zadanych parametrach.

Zakres zadania: wykorzystanie podstawowych narzędzi programistycznych, związanych z obsługą i przetwarzaniem danych, do budowy przyrządu wirtualnego, umożliwiającego symulowanie wybranego typu sygnału diagnostycznego oraz jego dostosowanie do zadania pomiarowego.

1. **Tworzenie nowego programu od postaw.** W przypadku kiedy niedostępny jest żaden szablon dla opracowywanego programu, można wykonać nowy czysty projekt i umieścić w nim dostępne przyrządy (Express VI), które spełniają wymagania tworzonego programu. Postępowanie według podanych wskazówek umożliwia utworzenie nowego projektu przyrządu wirtualnego, prezentuje zastosowanie gotowego przyrządu wirtualnego (Express VI) i dodanie go do schematu blokowego.
2. W oknie dialogowym wyświetlonym bezpośrednio po uruchomieniu LabVIEW należy wybrać opcję [New...] (Nowy...). Następnie w kolejnym oknie dialogowym wybrać opcję [Blank VI] (czysty program). Po zaznaczeniu opcji [Blank VI] zatwierdzić decyzję, wciskając przycisk [OK].
3. Przejrzeć zawartość (chwilowo pustych) okien panelu czołowego i schematu blokowego. Sprawdzić, czy mogą być wyświetlane palety Controls (front panel), Functions (block diagram) oraz paleta Tools (narzędzi).
4. Jeśli niewidoczne jest okno Context Help (pomoc kontekstowa), trzeba je uaktywnić. W tym celu należy z menu Help wybrać polecenie Show Context Help lub wcisnąć kombinację klawiszy [Ctrl] + [H]. W oknie będą pojawiać się informacje dotyczące aktualnie zaznaczanych obiektów.

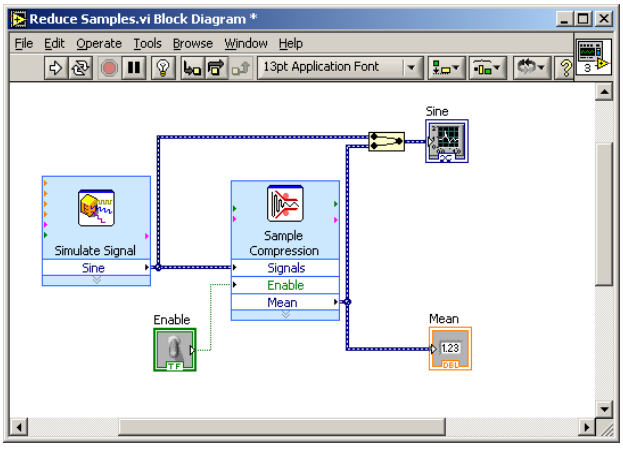
5. **Opracowanie kodu programu w oknie schematu blokowego.** W oknie schematu blokowego (block diagram) z palety funkcji (Functions Palette), z zakładki Express należy zaznaczyć opcję Input (wejście). W oknie pomocy kontekstowej (Context Help) ukaże się informacja dotycząca użycia wejściowych przyrządów wirtualnych.
6. Wybrać z palety Functions, z zakładki Express i z grupy Input, obiekt Simulate Signal (symulator sygnału) i wstawić go do panelu schematu blokowego. Automatycznie zostanie wyświetlony arkusz Właściwości (Properties) obiektu (jeżeli nie, należy go wyświetlić – menu kontekstowe). W oknie dialogowym określić rodzaj sygnału na sinusoidalny (sine), ustawić wartości częstotliwości sygnału na 10,7 Hz i amplitudę na 2,0 V.

7. Wynik dokonanych zmian ustawień można obserwować w oknie Results Preview. Po wykonaniu tych instrukcji należy zaakceptować wprowadzone zmiany w oknie Configure Simulate Signal, wybierając przycisk [OK].
8. Przesuwając wskaźnik myszy nad obiekt Simulate Signal Express VI, zapoznać się z informacjami wyświetlanymi w oknie Context Help.
9. Wynik pracy zapisać w pliku 022_RedukcjaProbek.vi w lokalizacji wskazanej przez prowadzącego.

10. **Modyfikowanie parametrów generowanego sygnału.** Postępowanie według wskazówek umożliwia znalezienie, za pomocą Systemu pomocy (Help) środowiska LabVIEW, gotowego przyrządu pomiarowego typu Express VI redukującego liczbę punktów w sygnale.
11. Wybrać polecenie [LabVIEW Help...] z menu Help. Polecenie może być uruchomione przez kombinację klawiszy [Ctrl]+[?].
12. Na karcie Wyszukaj, w polu „Wpisz poszukiwane słowo" [Type In Word(s) to search] należy wpisać tekst "sample compression".
13. Poszukiwanie inicjuje klawisz [Enter] lub przycisk Lista tematów [List Topics].
14. W wyszukanej liście wybrać temat: Sample Compression Express VI. Klikając dwukrotnie lewym klawiszem myszki, można zatwierdzić wybór nazwy wybranego tematu. Po zatwierdzeniu tej instrukcji w oknie informacyjnym pojawią się dane dotyczące Sample Compression Express VI.
15. Po przeczytaniu informacji, wybrać przycisk [Add on the block diagram] (dodaj do schematu blokowego). Nastąpi wtedy uaktywnienie procedury umożliwiającej wstawienie wybranego elementu do tworzonego projektu. Rezultatem prawidłowo wykonanej instrukcji jest uaktywnienie schematu blokowego (block diagram) i ukazanie się na nim wyszukanego elementu.
16. Obiekt Sample Compression można również wstawić ręcznie, wybierając go z palety Functions, z zakładki Express i z grupy Signal Manipulation.
17. W obszarze schematu blokowego programu należy przesunąć Sample Compression Express VI na prawo od Simulate Signal Express VI.
18. Zmodyfikować ustawienia Sample Compression Express VI (w części Reduction Specifications) tak, aby wartość Reduction factor (współczynnik redukcji) wynosiła 25. Wprowadzone zmiany zatwierdzić, klikając [OK].
19. Połączyć wyjście Sine obiektu Simulate Signal Express VI z wejściem obiektu Sample Compression Express VI.

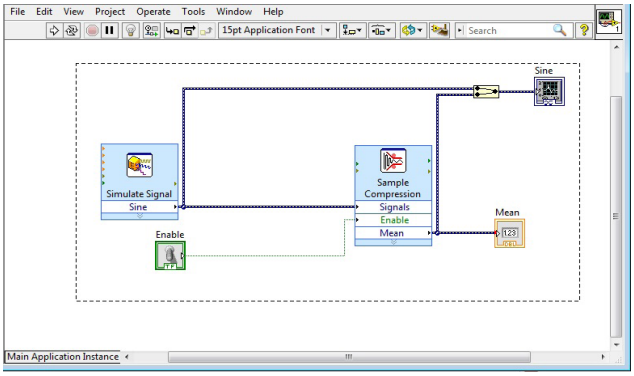
20. **Modyfikowanie panelu czołowego.** Wskaźniki umieszczane na front panelu programu umożliwiają symulowanie wejściowych wielkości fizycznych i dostarczanie danych wejściowych do przyrządów wirtualnych. Postępowanie według wskazówek prowadzi do wstawienia w tworzonym projekcie niezbędnych do sterowania jego pracą kontrolek (controls) i wskaźników (indicators).

21. W oknie schematu blokowego za pomocą prawego przycisku myszy wybrać wyjście Mean (średnia) obiektu Sample Compression Express VI, zaś z menu Create wybrać polecenie Numeric Indicator (wskaźnik numeryczny). Na panelu czołowym umieszczona zostanie ikona reprezentująca stworzony obiekt. Automatycznie wykonane zostanie połączenie między tym wskaźnikiem a wyjściem Mean obiektu Sample Compression Express VI.
22. Za pomocą prawego przycisku myszy wybrać wyjście Mean (średnia) obiektu Sample Compression Express VI, zaś z menu Select Input/Output wybrać polecenie Enable (Aktywuj).
23. Za pomocą prawego przycisku myszy wybrać wejście Enable (Aktywuj) obiektu Sample Compression VI, z menu Create wybrać polecenie Control (kontrolka/regulator). Podobnie jak w przypadku tworzenia wskaźnika Mean, zostanie wyświetlona ikona reprezentująca wstawiony obiekt.
24. Wybrać prawym klawiszem myszy połączenie wyjścia Sine obiektu Simulate Signal i z menu Create wybrać polecenie Graph Indicator. Po wykonaniu tej instrukcji na obszarze schematu blokowego wyświetlony zostanie wygenerowany obiekt.
25. Wykonać połączenie między wyjściem Mean obiektu Sample Compression VI a wyświetlaczem graficznym Sine. Pojawi się obiekt Merge Signals, umożliwiający wyświetlenie większej liczby przebiegów wykresów na jednym wyświetlaczu.
26. Uporządkować elementy oraz wykonane połączenie tak, aby zbudowany na ekranie schemat blokowy był przejrzysty i zbliżony do przestawionego na rysunku.



27. Przejść do okna panelu czołowego. Należy zwrócić uwagę na elementy, które zostały umieszczone automatycznie po wprowadzeniu ich do schematu blokowego.
28. Zapisać program na bieżącym etapie pracy pod wprowadzoną uprzednio nazwą (użyć np. skrótu klawiszowego [Ctrl] + [S]).

29. **Konfiguracja przyrządu wirtualnego do pracy ciągłej, zarządzanej przez użytkownika.** W aktualnym stanie program generuje tylko jeden sygnał i zatrzymuje się. Działanie przyrządu wirtualnego (VI), aż do spełnienia pewnego warunku, realizowane jest poprzez zastosowanie w programie pętli While. Postępowanie według podanych wskazówek (krok po kroku) umożliwia dodanie do projektu pętli While, co spowoduje działanie programu do momentu zatrzymania go przez użytkownika.
30. W oknie panelu sterowania uruchomić program. Program wykona się jednokrotnie i samoczynnie zakończy pracę.
31. Wyświetlić okno schematu blokowego, a następnie z palety Functions, z zakładki Express i z grupy Execution Control, wybrać obiekt pętli While Loop.
32. Wskaźnik myszy zmieni rodzaj wyświetlanego znacznika. Należy otoczyć wskaźnikiem myszy wszystkie dotychczasowe obiekty od lewego górnego rogu do prawego dolnego. Proces „otaczania" pętlą While przedstawiony został na rysunku.

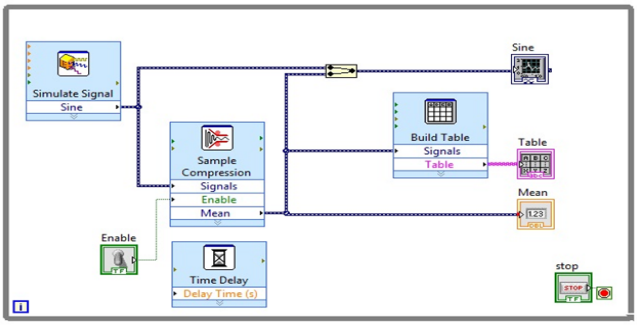


33. Przejść do okna panelu czołowego. Uruchomić program przyciskiem Run i obserwować zmiany zachodzące na wykresie, wprowadzone po zastosowaniu pętli While.
34. Zapisać zmiany w pliku pod bieżącą nazwą.

35. **Kontrola prędkości działania programu.** Liczbę generowanych punktów na wyświetlaczu graficznym można zmniejszyć, dodając instrument Time Delay Express VI (opóźnienie) w oknie schematu blokowego. Postępowanie według podanych dalej wskazówek pozwala na wprowadzenie kontroli prędkości działania przyrządu wirtualnego (VI).
36. W oknie schematu blokowego z palety Functions, z zakładki Express, z grupy Execution Control należy wybrać obiekt Time Delay Express VI (opóźnienie), a następnie umieścić go wewnątrz obszaru pętli While.
37. Zadeklarować czas opóźnienia (Time Delay) na 0,250 (sekundy).
38. Zatwierdzić zmiany przyciskiem [OK].

39. Wyświetlić okno panelu czołowego (front panel) i uruchomić program.
40. Obserwować zmiany zachodzące na wykresie. Sprawdzić, jaką rolę w tworzonym projekcie spełnia przełącznik Enable.
41. Zatrzymać program przyciskiem [STOP] z panelu czołowego.
42. Zapisać zmiany w programie.

43. **Prezentacja danych generatora sygnału diagnostycznego w postaci tabeli.** Postępowanie według podanych wskazówek pozwoli na wyświetlenie gromadzonych danych w postaci dostosowanej do potrzeb procesu diagnostycznego. Dane będą prezentowane jako wartości średnich kroczących generowanego sygnału podstawowego i pokazywane w tabeli na panelu czołowym.
44. Wyświetlić okno panelu czołowego i z palety Controls, z zakładki Express, z grupy Text Indicators (wskaźniki tekstowe), wybrać obiekt Express Table. Wstawić go do okna panelu czołowego, obok ikony wykresu.
45. Uaktywnić okno schematu blokowego. Należy zwrócić uwagę, że terminal tabeli Table został automatycznie połączony z Build Table Express VI.
46. Sprawdzić, czy obiekty Build Table VI i Table znajdują się wewnątrz obszaru pętli While. Jeżeli nie, to należy przenieść je tam razem poprzez wygenerowane przez program połączenie.
47. Ulokować Build Table VI i Table z prawej strony terminala Mean.
48. Wykonać połączenie między terminalem Mean instrumentu Sample Compresion Express VI a wejściem Signals obiektu Build Table Express VI. Przeprowadzenie powyższych instrukcji powinno zaowocować stworzeniem schematu blokowego wraz z połączeniami podobnego do przedstawionego na rysunku.



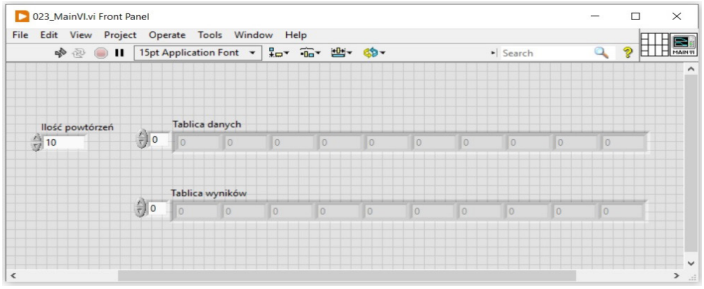
49. Wyświetlić okno panelu czołowego i uruchomić program.
50. Sprawdzić, jaką rolę pełni teraz przełącznik Enable, w przypadkach, w których przełącznik jest w pozycji On i Off.
51. Zatrzymać działanie programu.
52. Zapisać zmiany w pliku pod bieżącą nazwą i zamknąć okno programu.

2.3.3. Identyfikacja i usuwanie błędów w programie

Cel zadania: poznanie technik wykrywania i usuwania błędów formalnych programu. Postępowanie według podanych wskazówek umożliwia zlokalizowanie i usunięcie błędów formalnych w pliku oraz sprawdzenia funkcjonowania przyrządu wirtualnego (VI).

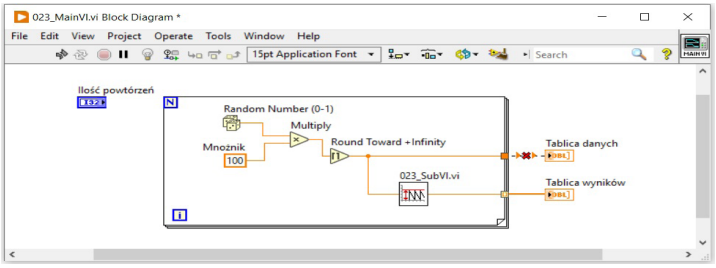
Zakres zadania: wykorzystanie podstawowych narzędzi edytorskich, związanych ze wspomaganiem pracy programisty, w zakresie sprawdzania poprawności wykonania kodu źródłowego programu pod względem formalnym. Obsługa funkcji Error list, narzędzi Probe Data, Breakpoint i Highlight Execusion oraz funkcji Step Into, Step Over oraz Step Out.

1. **Lokalizacja błędów formalnych w plikach przyrządów wirtualnych.** Z menu File wybrać polecenie Open. Z lokalizacji wskazanej przez prowadzącego wybrać plik 023_MainVI.vi. Wykonanie powyższych instrukcji umożliwia otworzenie gotowego programu, którego panel czołowy przedstawiony został na rysunku.



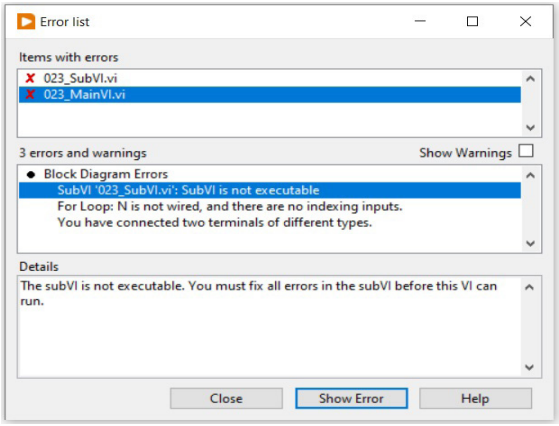
Należy zwrócić uwagę, że standardowy przycisk Run paska narzędzi ma aktualnie postać złamanej strzałki. Widoczny przycisk informuje użytkownika, że w otworzonym programie lub w jego podprogramach są błędy formalne, uniemożliwiające prawidłową pracę programu. Do momentu wyeliminowania przedstawionych błędów nie możliwe będzie uruchomienie programu.

2. Wyświetlić okno schematu blokowego (jego wygląd został przedstawiony na rysunku).



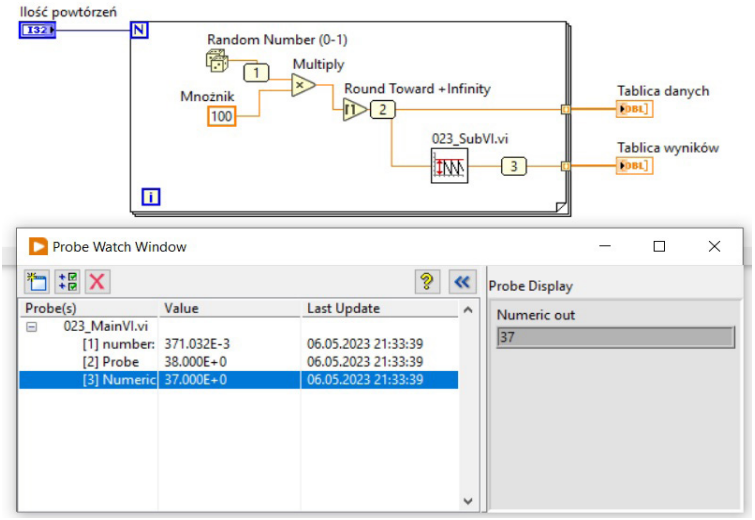
3. W celu znalezienia i usunięcia błędów popełnionych przy projektowaniu należy uruchomić za pomocą przycisku funkcję Error list, wyświetlającą okno listy błędów (patrz rysunek). W oknie informacyjnym przedstawione zostały wykryte błędy w programie głównym i podprogramie:

- podprogram 023_SubVI.vi jest niewykonywalny,
- brak sygnału definiującego liczbę powtórzeń pętli For,
- nieprawidłowe podłączenie sygnału do elementu Tablica danych.



4. W celu wyeliminowania pierwszego błędu, należy przejść do okna schematu blokowego podprogramu 023_SubVI.vi, poprzez dwukrotne kliknięcie lewym klawiszem myszki w obszar tego obiektu. Zostanie uaktywnione okno panelu czołowego tego obiektu, wtedy należy uruchomić okno jego schematu blokowego i wcisnąć przycisk funkcji List Errors. Wyświetlona informacja opisuje, gdzie leży problem powodujący nieprawidłowe działanie programu. Tym błędem jest brak połączenia między modułem dzielenia (Multiply) i modułem dekrementacji (Decrement). Należy wykonać brakujące połączenie i zapisać zmiany w pliku.
5. Wyeliminowanie drugiego błędu polega na stworzeniu połączenia między ikoną kontrolki Ilość powtórzeń a terminalem N pętli For. Należy wykonać brakujące połączenie.
6. Eliminacja trzeciego błędu polega na zmianie trybu związanego z prezentacją danych w jednym z terminali wyjściowych pętli For. Obecnie ustawiony jest tryb Last Value, a powinien być ustawiony tryb Indexing. Zmiana trybu odbywa się po wywołaniu menu kontekstowego tego tunelu i zmianie wybranej opcji z zakładce Tunnel Mode.
7. Wprowadzone poprawki umożliwiły wyeliminowanie błędów, co spowodowało uaktywnienie przycisku Run.
8. Zapisać plik poprawionego programu na dysku pod bieżącą nazwą i uruchomić go, sprawdzając jego działanie.

9. **Kontrola funkcjonowania programu.**
   W aktywnym oknie schematu blokowego uruchomić program przyciskiem Run.
10. Uruchomić program w trybie ciągłym. Użyć w tym celu przycisku Run Continuously. Obserwować różnice pomiędzy działaniem programu w trybie pojedynczego wywołania (Run) a trybie ciągłego wywołania (Run Continuously).
11. Prześledzić działanie programu za pomocą Highlight Execution (przycisk paska narzędzi).
12. Prześledzić działanie programu metodą krokową. Sprawdzić funkcjonowanie przycisków Step Into, Step Over oraz Step Out.
13. Przy analizie podprogramu skorzystać również ze Step Into, a do wyjścia z podprogramu ze Step Out.
14. Sprawdzić funkcjonowanie narzędzia Probe Data (próbówka) znajdującego się w palecie Tools. Aby prawidłowo użyć to narzędzie, należy umieścić je na dowolnym połączeniu schematu blokowego. Po wykonaniu podłączenia narzędzia, na ekranie wyświetlone zostaje jego okno informacyjne (patrz rysunek).



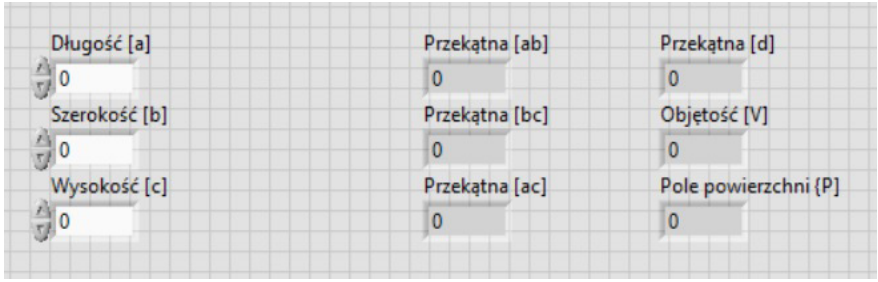
15. Kilkukrotnie uruchomić program, obserwując wskazania narzędzia Probe.
16. Usunąć punkty kontrolne.
17. Na dowolnie wybranym węźle, połączeniu zamieścić narzędzie Breakpoint. Uruchomić program i skontrolować, czy przerwanie działania programu nastąpiło w oczekiwanym punkcie.
18. Powtórzyć operację w kilku wybranych punktach.
19. Usunąć punkty Breakpoint.
20. Zapisać program w bieżącej postaci.
21. Zamknąć okno programu i okno środowiska LabVIEW.

2.3.4. Prostopadłościan opracowywanie SubVI

Cel zadania: utworzenie kompletnego przyrządu wirtualnego do wyznaczania, na podstawie danych o długościach boków a, b i c, podstawowych wielkości charakterystycznych dla prostopadłościanu. Dla programu zostaną utworzone: ikona identyfikacyjna oraz terminal wejść i wyjść.

Zakres zadania: postępowanie według podanych instrukcji umożliwia zbudowanie przyrządu wirtualnego jako procedury programowej, pozwalającej na wyznaczanie przekątnych, objętości i pola powierzchni prostopadłościanu oraz przystosowanie programu do zastosowania go jako podprogramu (SubVI) w innych programach.

1. **Tworzenie front panelu programu.** Uruchomić program LabVIEW i otworzyć nowy, czysty plik programu (Blank VI).
2. Po wyświetleniu nowego, czystego pliku programu, należy z menu Window wybrać polecenie Tile Left and Right. Daje ono możliwość jednoczesnego wyświetlenia obok siebie dwóch podstawowych okiem projektu (okna panelu czołowego i okna schematu blokowego). Umożliwia to sprawne przemieszczanie się między tymi oknami.
3. Umieścić na panelu czołowym 3 kontrolki numeryczne. W tym celu wybrać trzykrotnie z palety Controls obiekt Numeric Control znajdujący się w grupie Numeric Controls. Wygenerowanym kontrolkom numerycznym nadać nazwy „Długość [a]", „Szeokość [b]" i „Wysokość [c]" (jeśli etykieta obiektu jest niewidoczna, wybrać Properties z menu kontekstowego i wpisać nazwę obiektu w pole Label zaznaczając opcję Visible).
4. W podobny sposób umieścić na panelu czołowym 6 wskaźników numerycznych (Numeric Indicator). Wskaźniki te są dostępne w palecie Controls grupie Numeric. Wstawionym wskaźnikom numerycznym nadaj nazwy „Przekątna [ab]", „Przekątna [bc]", „Przekątna [ac]" „Przekątna [d]" „Objętość [V]" oraz „Pole powierzchni [P]".
5. Rozmieścić wstawione do okna front panelu elementy w sposób zbliżony do przestawionego na rysunku.



6. **Tworzenie schematu blokowego programu.** Przejść do okna schematu blokowego i uaktywnić paletę Function. Następnie uaktywnić grupę obiektów Express Numeric znajdującą się w zakładce Express → Arithmetic & Comparison.
7. Umieścić w obszarze okna schematu blokowego następujące obiekty:

- trzy moduły dodawania (Add),
- trzy moduły potęgowania (Square),
- cztery moduły pierwiastkowania (Square Root),
- sześć modułów funkcji Compound Arithmetic. Czterem modułom zadeklarować funkcję mnożenia (Multiply), a dwóm funkcję dodawania (Add),
- stałą liczbową DBL Numeric Constant o wartości 2.

8. Po zamieszczeniu wymaganych obiektów można przystąpić do tworzenia połączeń. Połączenia mają być tak wykonane, aby umożliwiały realizację następujących funkcji:

$$ab = \sqrt{a^2 + b^2},$$

$$bc = \sqrt{b^2 + c^2},$$

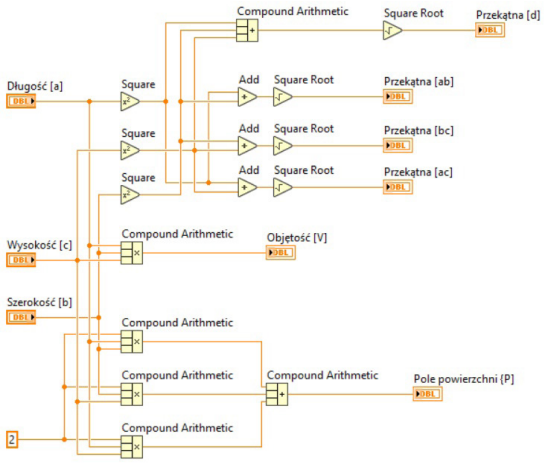
$$ac = \sqrt{a^2 + c^2},$$

$$d = \sqrt{a^2 + b^2 + c^2},$$

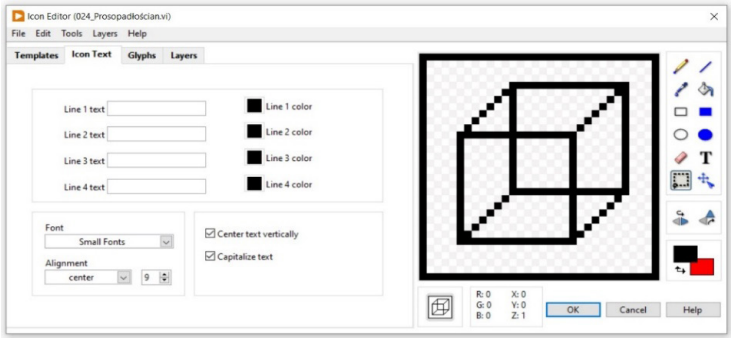
$$V = abc\ ,$$

$$P = 2ab + 2bc + 2ac.$$

9. Po zrealizowaniu powyższych instrukcji należy przeprowadzić porządkowanie obiektów w oknie schematu blokowego tak, żeby wszystkie połączenia były widoczne i prezentowały się w przejrzysty sposób. Na rysunku przedstawione zostało okno schematu blokowego po uporządkowaniu znajdujących się na nim obiektów i poprowadzeniu niezbędnych połączeń.



10. **Uruchomienie programu.** Uaktywnić okno panelu czołowego.
11. Wprowadzić wartości liczbowe w pola kontrolek/regulatorów liczbowych.
12. W celu sprawdzenia poprawności działania stworzonego projektu uruchomić go, wykorzystując funkcję Run Continuously (ciągłe działanie).
13. Zapisać opracowaną aplikację na dysku pod nazwą 024_Prostopadłościan.vi.
14. **Tworzenie ikony i panelu terminali we/wy dla programu VI.** W celu zmiany standardowej ikony w ikonę charakterystyczną dla danego programu (procedury), należy uruchomić edytor ikon (Icon Editor). Realizacja tego procesu polega na wskazaniu kursorem myszy na prawy górny róg okna panelu czołowego lub okna schematu blokowego i dwukrotnie kliknąć lewym klawiszem myszy lub po rozwinięciu menu skrótów, za pomocą prawego klawisza myszy, wybrać funkcję Edit Icon. Po wykonaniu tych czynności na ekranie monitora wyświetlone zostanie okno edytora ikon.



Uwaga: edytor ikon umożliwia stworzenie własnych symboli identyfikacyjnych dla nowopowstających przyrządów wirtualnych. W edytorze ikon za pomocą funkcji Import Glyph from File jest możliwy import rysunków z plików graficznych.

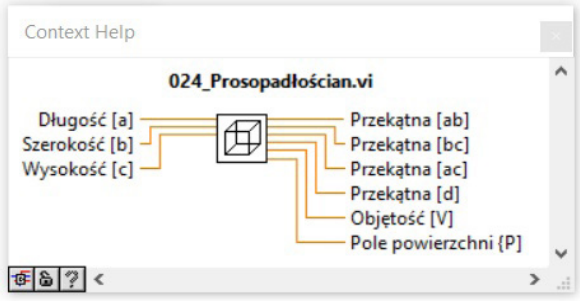
15. Do edycji projektowanego obszaru ikony służą narzędzia znajdujące się w belce po prawej stronie okna edytora. Za ich pomocą można stworzyć dowolny obraz ikony. Po zakończeniu edycji zmianę wyglądu ikony zatwierdza się, wciskając przycisk [OK].

Uwaga: edytor ikon umożliwia stosowanie różnego rodzaju czcionek. Do tego celu służy narzędzie Text Tool uruchamiane poprzez dwukrotne kliknięcie prawego przycisku myszy nad polem obiektu Text.

16. W celu wykonania układu połączeń (wejść i wyjść dla tworzonego przyrządu wirtualnego), za pomocą których możliwe będzie podłączenie go jako obiektu w innym projekcie, należy z menu uruchomić funkcję Connector (w oknie panelu

czołowego pole sąsiadujące z lewej strony okna edytora ikonki). Automatycznie przy tworzeniu nowego programu generowany jest szablon posiadający 12 terminali wejścia/wyjścia.

17. W celu dopasowania szablonu do wymagań tworzonej aplikacji, należy wybrać odpowiadający potrzebom użytkownika wzorzec (funkcja Patterns w menu skrótów edytora połączeń). W przypadku bieżącego programu należy użyć wzorca z 6 wejściami i 6 wyjściami. Jeżeli tło szablonu nie jest białe oznacza to, że wcześniej zostały już przypisane typy sygnałów wejściowych lub wyjściowych. Przed przystąpieniem do tworzenia nowego układu połączeń wejścia i wyjścia dla przyrządu wirtualnego, należy w menu kontekstowym uruchomić funkcję Disconnect All Terminals.
18. W tworzonym przyrządzie wirtualnym obiektami wejściowymi są kontrolki „Długość [a]", „Szerokość [b]" i „Wysokość [c]" (kontrolki numeryczne), a obiektami wyjściowymi, wskaźniki „Przekątna [ab]", „Przekątna [bc]", „Przekątna [ac]" „Przekątna [d]" „Objętość [V]" oraz „Pole powierzchni [P]" (wskaźniki numeryczne). Aby dokonać stosownych połączeń i zdefiniować terminale I/O dla tego programu, należy w palecie narzędzi uruchomić funkcję tworzenia połączeń Connect Wire i przesunąć wskaźnik myszki nad obszar obiektu „Długość [a]", potem wcisnąć lewy przycisk myszki (obiekt powinien zostać zaznaczony obwiednią), następnie skierować myszkę nad lewą część kwadratu i ponownie kliknąć lewy przycisk. Kolor pola powinien się zmienić z białego na kolor sygnału przypisanego danemu rodzajowi obiektu, z którym dokonane zostało połączenie (w tym przypadku na pomarańczowy). Analogicznie należy wykonać połączenia między pozostałymi elementami, przypisując je do konkretnych terminali.
19. Po utworzeniu panelu terminali i przejściu do okna panelu czołowego, wyświetlić okno pomocy kontekstowej (Help/Show Context Help lub CTRL + H).
20. W trakcie obserwowania okna pomocy kontekstowej, umieścić wskaźnik myszy nad obszarem konektora/ikony przyrządu wirtualnego. W oknie pomocy kontekstowej powinna zostać wyświetlona informacja w postaci analogicznej do przedstawionej na rysunku.



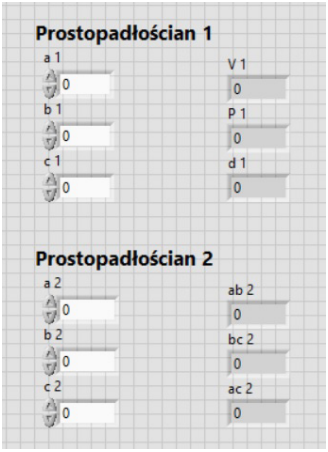
21. Zapisać dokonane zmiany w programie pod istniejącą nazwą (024_Prostopadłościan.vi).

2.3.5. Geometria – wykorzystanie SubVI

Cel zadania: budowa przyrządu wirtualnego do wyznaczania charakterystycznych parametrów prostopadłościanów. Realizacja założonych procedur obliczeniowych przez wielokrotne zastosowanie danego podprogramu. Poznanie technik wykorzystania wcześniej wykonanych programów do realizacji zadań w programach głównych.

Zakres zadania: sprawdzenie funkcjonowania wcześniej przygotowywanych programów. Dopasowanie zakresu realizowanych zadań do potrzeb aktualnego programu. Ograniczenie miejsca realizacji procedury programowej przez zamknięcie jej w jednym węźle programowym.

1. **Tworzenie front panelu programu.** Uruchomić środowisko LabVIEW i otworzyć nowy plik programu. Zbudować panel czołowy programu, przypominający przedstawiony na rysunku.

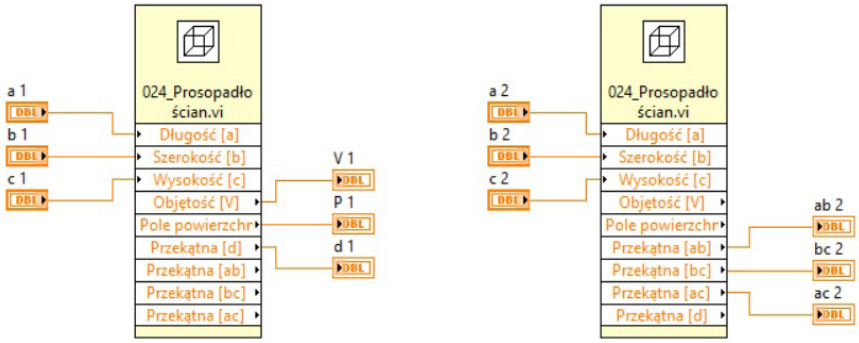


2. Na front panelu programu należy umieścić następujące elementy znajdujące się w palecie Controls w zakładce Modern → Numeric:

- 6 kontrolek numerycznych typu Numeric Control, którym trzeba nadać nazwy: „a 1”, „b 1”, „c 1” oraz „a 2”, „b 2”, „c 2”. Będą to elementy umożliwiające wprowadzenie długości boków prostopadłościanów 1 i 2,
- 6 wskaźników numerycznych typu Numeric Indicator, którym trzeba nadać nazwy: „V 1”, „P 1”, „d 1” oraz „ab 2”, „bc 2”, „ac 2”. Będą to elementy umożliwiające prezentacje wyników wybranych parametrów dla prostopadłościanów 1 i 2.

3. Wstawić dwie etykiety tekstowe i nadać im nazwy „Prostopadłościan 1” i „Prostopadłościan 2”.

4. **Tworzenie schematu blokowego programu.** Uaktywnić okno schematu blokowego i sprawdzić, czy znajdują się tam ikony reprezentujące poszczególne kontrolki i wskaźniki, umieszczone wcześniej na froncie panelu programu.
5. Uaktywnić paletę Functions. Następnie wejść do zakładki „Select a VI...". W uaktywnionym oknie Select the VI to Open odnaleźć następujący plik 024_Prostopadłościan.vi. Jest to plik z gotowym programem służącym do wyznaczania parametrów geometrycznych prostopadłościanu wykonanego w poprzednim ćwiczeniu. Prawidłowe zrealizowanie tego polecenia skutkuje wstawieniem obiektu 024_Prostopadłościan.vi do okna schematu blokowego.
6. Korzystając z powyższej instrukcji, w analogiczny sposób wstawić drugi obiekt 024_Prostopadłościan.vi.
7. Uporządkować ikony kontrolek i wskaźników. Wykonać ścieżki połączeń tak, by powstał schemat połączeń analogiczny do przedstawionego na rysunku.



8. Należy zwrócić uwagę, że program składa się tylko z ikon reprezentujących poszczególne kontrolki i wskaźniki umieszczone na front panelu oraz dwa elementy podprogramu (SubVI). Wyznaczanie wybranych parametrów odbywa się wewnątrz SubVI.
9. Należy zwrócić uwagę, że w przypadku wykorzystywania SubVI w programach głównych, zwiększa się przejrzystość programu głównego, procedury obliczeniowe zajmują mniej miejsca oraz możliwe jest wielokrotne wstawiane danego SubVI w danym programie głównym.
10. Należy zwrócić uwagę, że w przypadku zastosowania danego SubVI można użyć tylko takich typów danych, jakie dany SubVI obsługuje.
11. Zapisać plik z wykonanym programem w pliku pod nazwą 025_Geometria.vi.
12. Uaktywnić okno panelu czołowego.
13. Wprowadzić liczby w pola kontrolek/regulatorów liczbowych. Uruchomić program, wykorzystując funkcję Run Continuously (ciągłe działanie).

2.4. Pytania kontrolne

1. Jak scharakteryzować zastosowanie okien panelu czołowego i schematu blokowego wraz ze sposobem wzajemnego układu okien i sposobami poruszania się pomiędzy oknami?
2. Jakie różnice występują pomiędzy narzędziami dostępnymi w oknach Front Panel oraz Block Diagram?
3. Jak scharakteryzować podstawowe typy danych obsługiwanych w środowisku LabVIEW?
4. Jakie narzędzia dostępne są przy śledzeniu kodu i wykrywaniu błędów? Należy opisać każde z nich.
5. Jak opisać możliwy przebieg procesu tworzenia, deklarowania właściwości oraz wstawiania podprogramu (SubVI)?

3. Pętle. Prezentacja, przekazywanie i magazynowanie danych

3.1. Cel zajęć

Bieżąca część pracy łączy w sobie dwa główne zagadnienia, tj. zapewnienie ciągłości wykonywania kodu oraz obsługę takich samych lub zróżnicowanych typów danych wraz z ich prezentacją. Celem zajęć jest teoretyczne i praktyczne zapoznanie użytkownika środowiska LabVIEW ze sterowaniem wykonywaniem kodu:

- zadeklarowaną liczbę razy lub do wystąpienia błędu (pętla For, For Loop),
- do czasu spełniania warunku lub do wystąpienia błędu (pętla While, While Loop.

Użytkownik zostanie zaznajomiony z mechanizmem przechowywania danych w przypadkach:

- wykonywania pętli w komórkach rejestru przesuwnego,
- danych typu liczbowego za pomocą macierzy,
- danych jednego typu za pomocą tablic,
- danych różnych typów za pomocą klastrów.

Jednocześnie zostaną przedstawione obiekty do prezentacji danych na panelu czołowym aplikacji:

- w funkcji czasu,
- jednej zmiennej w funkcji drugiej zmiennej.

3.2. Wstęp

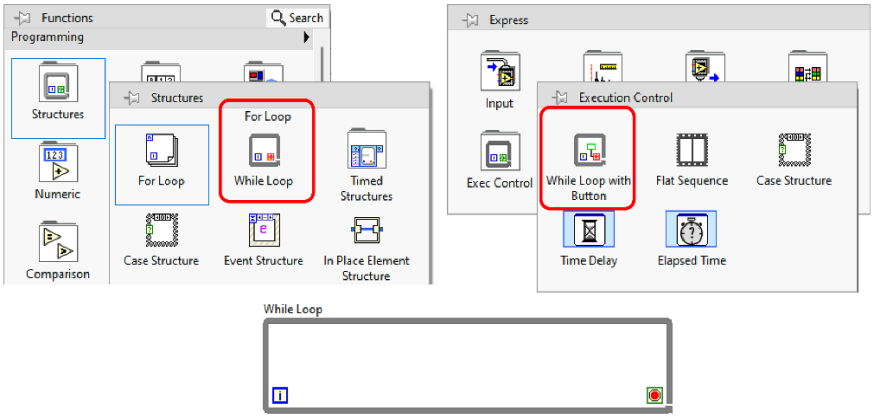
3.2.1. Pętla While (pętla warunkowa)

Pętla While stanowi odpowiednik znanej z tekstowych języków programowania pętli Do lub Repeat Until. Zgodnie z nazwą, jej funkcjonowanie polega na wykonywaniu kodu programu otoczonego pętlą programu do momentu spełnienia warunku pętli [2, 3]. Pętla While stanowi podstawową strukturę pozwalającą na ciągłą pracę programu. Dostępna jest z poziomu dwóch palet schematu blokowego (rys. 3.1):

- paleta Functions /Programming /Structures,
- paleta Functions /Express /Execution Control.

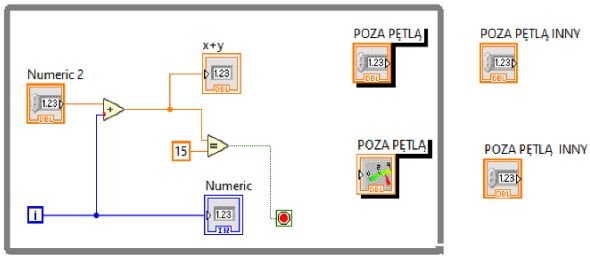
Pętla While wyposażona jest w dwa terminale:

- Terminal i (liczby iteracji) – terminal wyjściowy udostępniający liczbę iteracji pętli zliczaną od zera.
- terminal warunkowy – terminal wejściowy. Terminal sterowania pętlą, do którego przypisuje się warunek wykonywania/zakończenia działania pętli. Domyślnym terminalem warunkowym jest terminal Stop if true (czerwony), który powoduje zakończenie działania po przypisaniu mu wartości logicznej true (1). Za pomocą menu kontekstowego możliwa jest elastyczna zmiana sposobu działania terminala na warunek Continue if true (zielony).



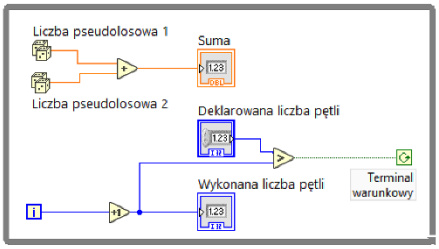
Rys. 3.1. Pętla warunkowa – widok w oknie schematu blokowego

Poprawne zamieszczanie kodu w pętli polega na otoczeniu pętlą powtarzanego kodu bezpośrednio po wybraniu ikony pętli z palety. W przypadku, kiedy takie działanie jest niemożliwe, poprawne jest przesuwanie elementów na schemacie blokowym kodu do wewnątrz pętli. Częstym problemem jest postępowanie polegające na przesuwaniu całej pętli na elementy kodu (optycznie elementy widać wewnątrz pętli z dodatkowym cieniem) (rys. 3.2). W takim przypadku elementy wykonywane są poza pętlą.



Rys. 3.2. Obsługa pętli warunkowej (While) – elementy poza pętlą

Na rysunku 3.3 zamieszczono kod przykładowego programu, wykorzystujący pętlę While. Głównym zadaniem programu jest obliczanie sumy dwóch liczb pseudolosowych z zakresu 0–1 i prezentowanie ich na panelu czołowym. Dodawanie wykonywane będzie do momentu, w którym wartość określająca liczbę pętli będzie mniejsza niż wartość zadeklarowana przez użytkownika. Należy zwrócić uwagę na trzy elementy. Terminal warunkowy został zdefiniowany do pracy w niedomyślnym trybie Continue if True. Wyświetlana na panelu czołowym wartość Wykonana liczba pętli musi zostać powiększona o jeden, ponieważ terminal iteracyjny dostarcza wartości począwszy od wartości zero. Pętla musi zostać wykonana minimum jeden raz, ponieważ warunek jej wykonania sprawdzany jest po wykonaniu kodu objętego obszarem pętli.

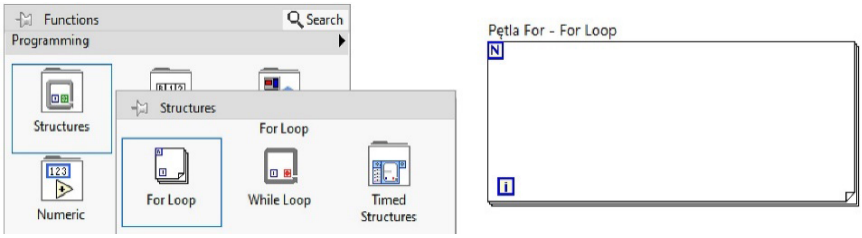


Rys. 3.3. Przykład użycia pętli warunkowej (While) przy dodawaniu wartości liczbowych

3.2.2. Pętla For

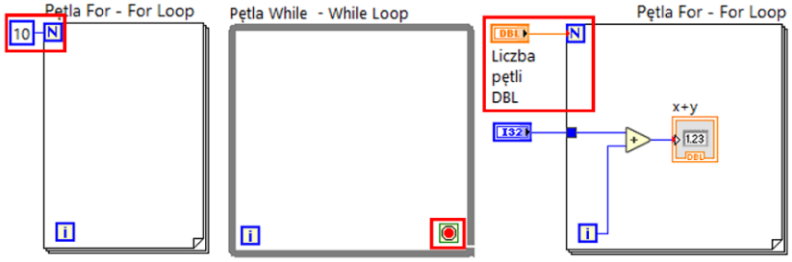
Standardowa Pętla For stanowi pętlę w z limitowaną liczbą iteracji kodu zamieszczonego wewnątrz pętli [2, 21]. Dostępna jest na schemacie blokowym poprzez paletę Functions /Programming /Structures (rys. 3.4). Typowo pętla For posiada dwa terminale:

- wejściowy N – do deklarowania liczby iteracji. Najczęściej podłącza się do niego stałą liczbową,
- wyjściowy i – udostępniający informację o liczbie iteracji zliczaną od zera.



Rys. 3.4. Widok pętli For wraz z paletą Structures

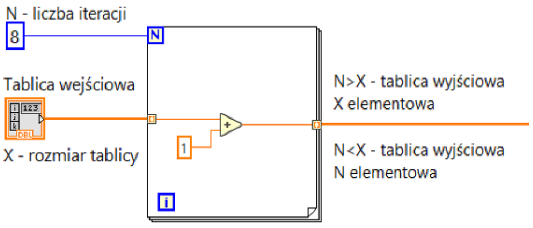
Podstawowa różnica pomiędzy pętlami For i While tkwi w czasie wykonywania pętli: pętla For wykonywana jest zadeklarowaną liczbę iteracji, a pętla While – do spełnienia warunku.



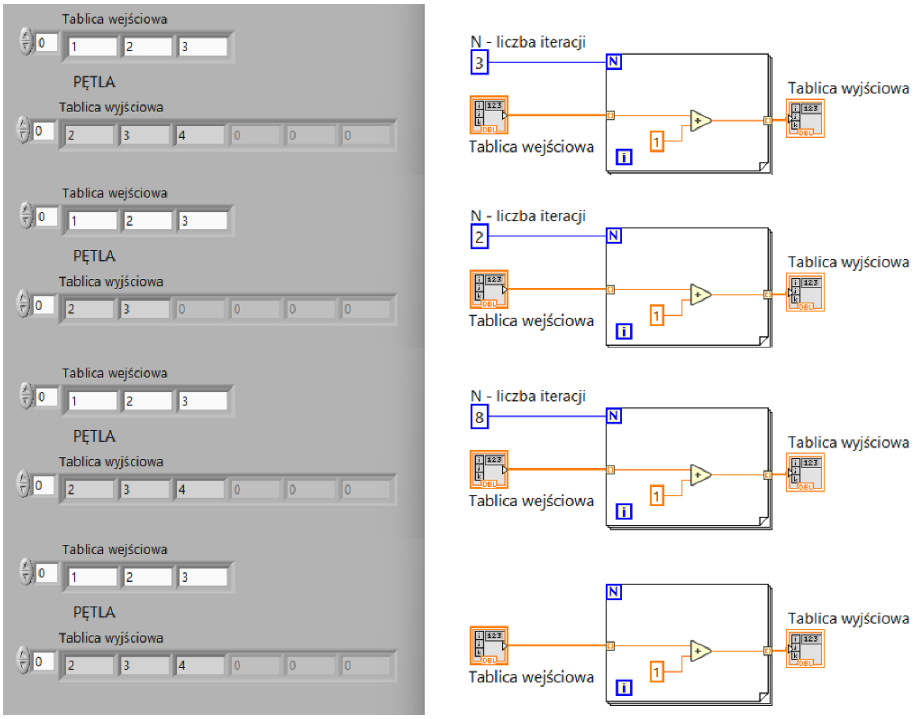
Rys. 3.5. Charakterystyczne elementy pętli For i While

Podstawowym sposobem deklarowania liczby iteracji jest przyłączenie zmiennej typu liczbowego do terminala iteracji (N). Przy zadeklarowaniu typu danych dla liczby pętli innego niż wymagany przez pętlę For (Integer 32) następuje automatyczne dopasowanie. Miejsce konwersji oznaczone jest znakiem (kropką, coercion dot) (rys. 3.5).

W przypadku operacji na zmiennych tablicowych może dojść do niezgodności liczby iteracji pętli (N) i wymiaru tablicy (X), tzn. liczba pętli może być większa lub mniejsza niż liczba elementów w tablicy (rys. 3.6). W przypadku takiej niezgodności pętla wykonywana jest tyle razy, na ile wskazuje mniejsza z wartości.



Rys. 3.6. Rozmiar tablicy wyjściowej przy niezgodności liczby iteracji i rozmiaru tablicy



Rys. 3.7. Deklaracja liczby pętli For

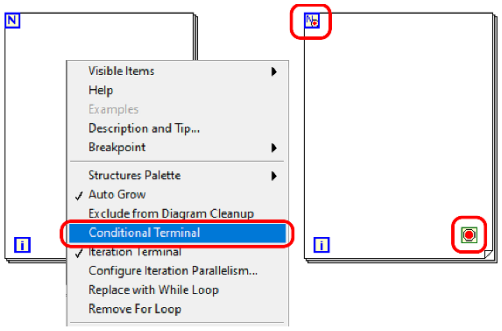
Potencjalne przypadki niezgodności liczby elementów tablicowych i liczby zadeklarowanych pętli zaprezentowano na rysunku 3.7:

- elementy tablicy We X = liczba iteracji N –> tablica wyjściowa X = N,
- elementów tablicy We X więcej niż liczba iteracji N – tablica wyjściowa N elementowa (N < X),
- elementów tablicy We X mniej niż liczba iteracji N – tablica wyjściowa X elementowa (X < N),
- brak liczby iteracji N – tablica wyjściowa X-elementowa (X – liczba elementów tablicy wejściowej).

3.2.3. Warunkowa pętla For

W LabVIEW dostępna jest hybryda pętli While i For, którą nazwano tutaj warunkową pętlą For. Warunkowa pętla For wykonuje zadeklarowaną liczbę iteracji, chyba że wcześniej zostanie spełniony warunek jej zakończenia. Taka pętla usuwa podstawowy problem pętli For w przypadkach, kiedy należy nieoczekiwanie zakończyć jej działanie jeszcze przed realizacją wszystkich zadeklarowanych pętli np. w przypadku wystąpienia błędu lub wcześniejszego osiągnięcia oczekiwanego wyniku. Eliminuje konieczność stosowania dodatkowych struktur sterujących kodem w przypadku zaistnienia błędów.

Przekształcenie pętli For w warunkową pętlę For polega na zadeklarowaniu wystąpienia terminala warunkowego za pomocą menu kontekstowego pętli i polecenia Conditional Terminal. Efektem deklaracji jest zamieszczenie terminala warunkowego w dolnym rogu pętli. To, że pętla For jest teraz warunkowa, sygnalizuje zmodyfikowany obraz terminala liczby iteracji (lewy górny róg pętli) (rys. 3.8).

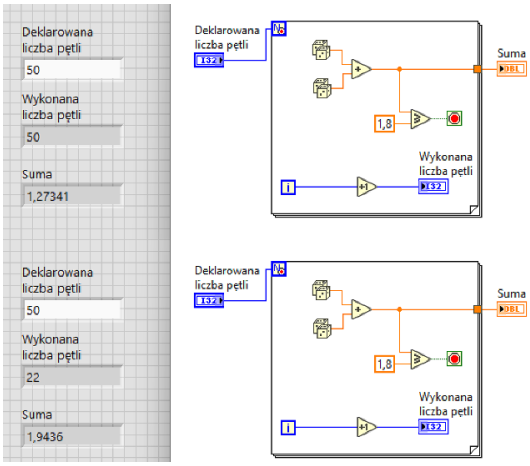


Rys. 3.8. Warunkowa pętla For – deklarowanie

Przykładowe działanie warunkowej pętli For zaprezentowano poniżej. Program zamieszczony w pętli dodaje dwie liczby pseudolosowe z zakresu 0÷1. Zadaniem programu jest dodawanie liczby określoną liczbę razy lub do osiągnięcia efektu, którym tutaj jest uzyskanie sumy większej lub równej 1,8.

Na rysunku 3.9 zamieszczono dwa efekty funkcjonowania programu:

- górny kod – działanie pętli przez całe zadeklarowane 50 iteracji, ponieważ przez 50 iteracji nie osiągnięto sumy liczb ≥1,8 (zadeklarowana liczba iteracji 50, wykonana liczba iteracji 50),
- dolny kod – działanie pętli przez 15 iteracji, ponieważ wcześniej (przy 15 wykonaniu pętli) suma liczb pseudolosowych osiągnęła wartość, która warunkowo zakończyła pętlę (zadeklarowana liczba iteracji 50, wykonana liczba iteracji 22).



Rys. 3.9. Warunkowa pętla For – przykład działania

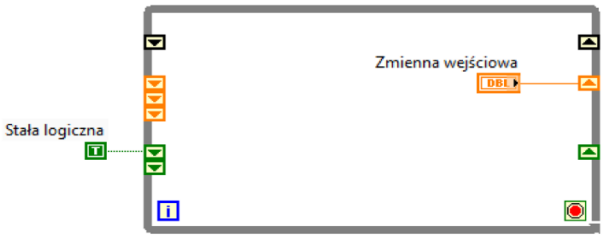
3.2.4. Rejestr przesuwny

W przypadku konieczności gromadzenia danych, wymagane jest stosowanie takich struktur, jak np. tablice czy klastry. Strukturę do tymczasowego przechowywania danych w pętlach bez wykorzystywania dodatkowych obiektów stanowi rejestr przesuwny (shift register).

Funkcjonowanie rejestru polega na przekazywaniu danych z jednej iteracji pętli (zapis do prawego terminala rejestru) do kolejnej komórki rejestru (lewy terminal rejestru). Z tej komórki rejestru wartość może być pobierana do momentu kolejnej aktualizacji komórki rejestru (do wykonania kolejnej pętli). Rejestr przesuwny jest dostępny dla obu typów pętli While i For (rys. 3.10). Takie tymczasowe magazynowanie danych jest szczególnie użyteczne przy uśrednianiu danych (upraszcza strukturę kodu przez niewprowadzanie dodatkowych obiektów).

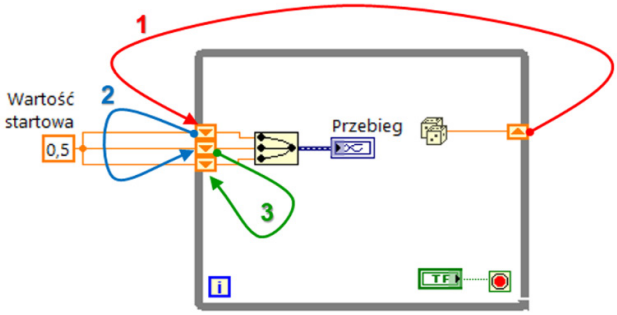
Typowe operacje związane z rejestrem:

- wprowadzanie rejestru przesuwnego – wybór pionowej krawędzi pętli klawiszem myszy / Add Shift Register,
- dodawanie kolejnych komórek rejestru – Add element z menu kontekstowego rejestru (lub zmiana rozmiaru lewego rejestru za pomocą myszy).



Rys. 3.10. Komórki rejestru przesuwnego zamieszczone w pętli While

Zasadę funkcjonowania rejestru przedstawiono na rysunku 3.11. Polega ona na przekazywaniu danych z jednej iteracji pętli (zapis do prawego terminala rejestru) do kolejnej komórki rejestru (lewy terminal rejestru). Kolejna wartość zapisana do terminala rejestru wypycha aktualną wartość. Chcąc poszerzyć historię danych należy wyposażyć rejestr w więcej komórek (elementów), na przykład za pomocą polecenia kontekstowego Add element.



Rys. 3.11. Zasada działania rejestru przesuwnego

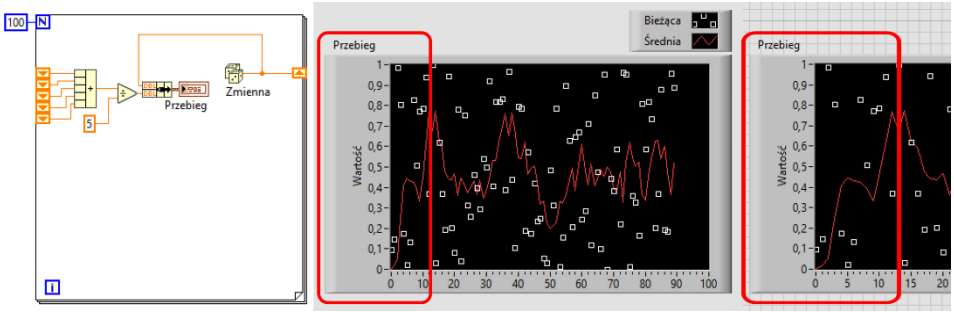
Cechy charakterystyczne rejestru przesuwnego:

- rejestr przyjmuje barwę zgodną z typem danych. Czarny rejestr oznacza brak przyłączonych danych – brak inicjacji. Do zadeklarowania typu danych wystarczy połączenie ze zmienną/stałą określonego typu dowolnego elementu rejestru z zewnątrz lub wewnątrz pętli,
- nieużywane (bez przypisanej bezpośrednio wartości) komórki rejestru (jeśli nie zostały im przypisane konkretne wartości) są inicjowane wartością 0 (0 – liczbowe, false – logiczne, empty string – dla zmiennych łańcuchowych).

Rejestr przesuwny można „idealnie” wykorzystywać przy chwilowym uśrednianiu wartości w powtarzalnych operacjach realizowanych w pętlach. Należy sobie zdawać sprawę z ograniczeń rejestru. Przy uśrednieniu problematyczne są wtedy początkowe (zerowe) wartości rejestru, które powodują zniekształcenie wyniku [16]. Pierwsze wartości uśredniane są razem ze „sztucznymi”, początkowymi wartościami zerowymi.

Przykładowy program z problemem błędnego początkowego uśredniania zilustrowano w oparciu o program zaprezentowany na rysunku 3.12. W trakcie jego pracy:

- rejestr inicjowany jest domyślną wartością zero,
- generowany jest przebieg czasowy zmiennej pseudolosowej (100 punktów) z oczekiwaną wartością średnią równą 0,5,
- na wykresie wyświetlane są zarówno punkty zmiennej losowej (kółka), jak i wartość średnia liczona w oparciu o 5 ostatnich wartości.



Rys. 3.12. Rejestr przesuwny – przykład zastosowania w uśrednianiu

Na prawej części rysunku 3.12 przedstawiono przebieg czasowy zmiennej pseudolosowej z powiększoną średnią za pierwsze 20 punktów. Początkowe wartości średniej zaniżone są przez inicjację rejestru przesuwnego domyślną wartością zero.

Problem zanika po wypełnieniu wszystkich komórek rejestru wartościami obliczonymi (zmierzonymi). Opcjonalnie można sztucznie korygować początkowe wartości średniej, inicjując rejestr wartościami oczekiwanymi (wprowadzamy przekłamanie wyników). Innym rozwiązaniem może być analiza i obserwacja danych pochodzących od punktu, w którym rejestr jest zapełniony wartościami rzeczywistymi (tutaj od punktu 5).

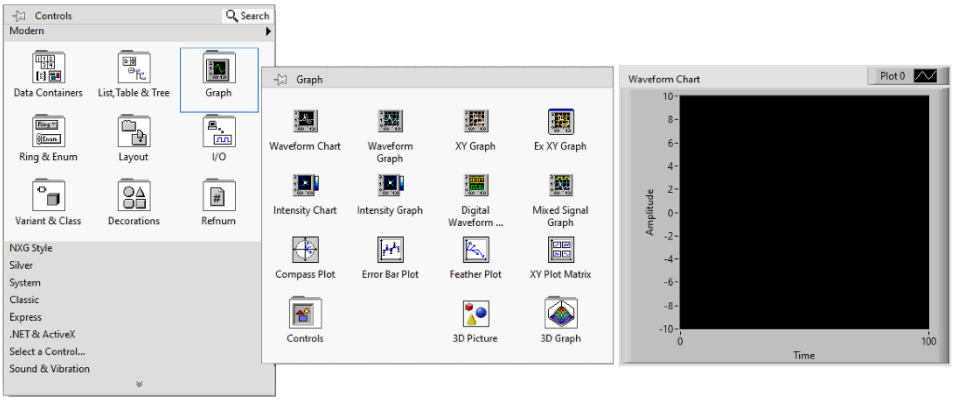
3.2.5. Wykres waveform chart

W wielu przypadkach wygodniejsze jest prezentowanie wyników w wersji graficznej (termometry, diody, zbiorniki, suwaki). Obiekt graficzny przedstawia jednocześnie całą serię danych, dzięki czemu możliwe jest łatwe zaobserwowanie trendów zmian analizowanych lub zbieranych wielkości. Najprostszym typem wykresu jest obiekt wykresu Waveform chart, który faktycznie jest wskaźnikiem liczbowym (typ numeric) (rys. 3.13).

Cechy charakterystyczne Waveform chart:

- kojarzy (łączy) otrzymywane wartości z kolejnością ich otrzymywania (uwzględnia różnice interwałów czasowych),
- umożliwia wyświetlanie jednej lub więcej zmiennej,
- posiada szerokie możliwości modyfikowania (formatowania):

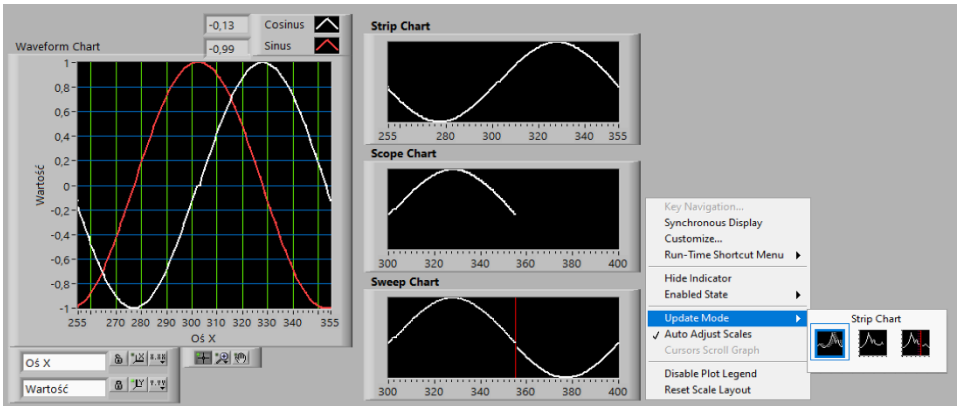
- wyglądu zewnętrznego,
- wyświetlania dodatkowych elementów,
- skalowania osi,
- formatu liczbowego osi,
- sposobu skalowania (zakres, krok).



Rys. 3.13. Paleta wskaźników graficznych

Prosta modyfikacja wyglądu wykresu opiera się o wyświetlanie i wygaszanie jego elementów (składników). Dostęp do modyfikowanych elementów uzyskuje się przez menu kontekstowe poszczególnych elementów wykresu oraz arkusz Właściwości.

Domyślnie przypisana obiektowi wykresu pamięć pozwala na przechowywanie 1024 punkty wykresu. Możliwa jest zmiana „długości historii".



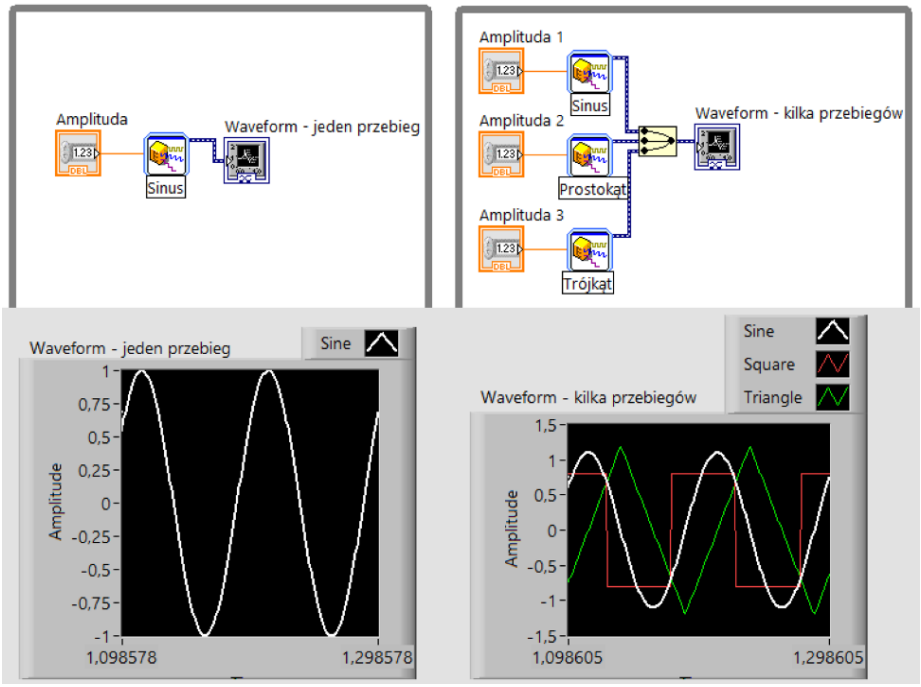
Rys. 3.14. Zmodyfikowany wygląd wskaźnika Waveform chart (z lewej). Deklaracja trybów wyświetlania przebiegów na wykresie (z prawej)

Waveform chart może wyświetlać przebiegi w trzech trybach (rys. 3.14). Wybór powinien uwzględniać wygodę pracy lub odzwierciedlać sposób wyświetlania rzeczywistego instrumentu odzwierciedlanego przez Waveform chart. Tryby wyświetlania można konfigurować przez menu kontekstowe (Advanced /Update Mode) lub arkusz właściwości. Tryby wyświetlania noszą nazwy:

- Strip chart – tryb domyślny, wyświetlanie danych bieżących od lewej do prawej z pozostawieniem historii,
- Scope chart – wyświetlanie okresowe danych z lewej do prawej (historia dotyczy wybranego przebiegu ze skali x) i czyszczeniem ekranu,
- Sweep chart – wyświetlanie okresowe danych z lewej do prawej, nowsze dane „czyszczą" starsze.

Jak wspomniano, Waveform chart może wyświetlać jeden lub więcej przebiegów (serii danych). Dla wyświetlania pojedynczej serii wartości liczbowych nie ma konieczności prowadzenia dodatkowej konfiguracji. Waveform chart akceptuje serie wartości skalarnych oraz zmienne dynamiczne (np. z Express VI).

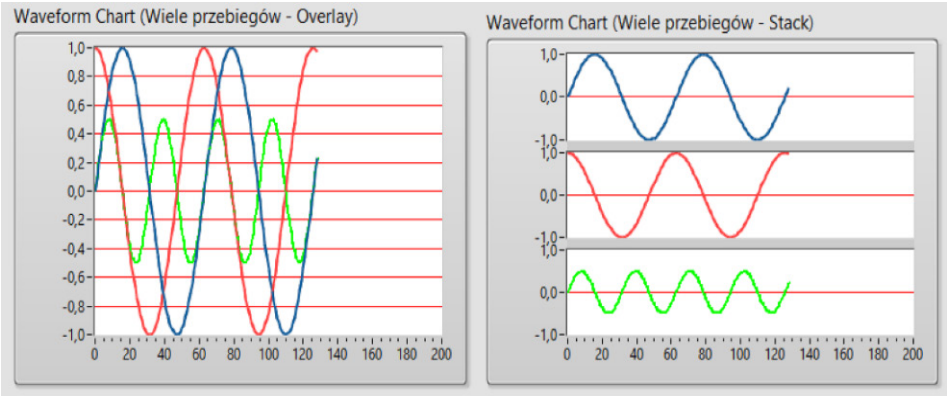
Wyświetlanie kilku połączonych wartości (przebiegów) wymaga połączenia grupy zmiennych w jeden „sygnał" np. za pomocą funkcji bundle, merge (rys. 3.15). Taki pojedynczy „sygnał" jest akceptowany przez wskaźnik Waveform chart.



Rys. 3.15. Obsługa Waveform chart przy wyświetlaniu jednego i grupy przebiegów

Przy prezentacji kilku przebiegów zmiennych pole wykresu można skonfigurować (np. przez menu kontekstowe wykresu (/Stack, /Overlay)) do pracy w trybie wyświetlania (rys. 3.16):

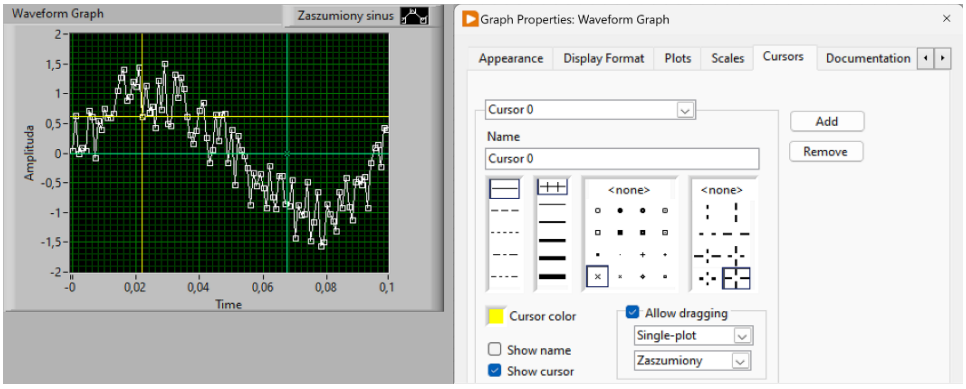
- Overlay – przebiegi nakładają się na siebie na jednym ekranie,
- Stack – przebiegi wyświetlane są indywidualnie na pojedynczych ekranach ułożonych jeden nad drugim (stos).



Rys. 3.16. Wykres typu Waveform chart pracujący w trybie Overlay i Stack

3.2.6. Wykres waveform graph

Wskaźnik Waveform graph optycznie przypomina Waveform chart. Wyświetla pojedyncze wartości jako funkcję y = f(x). Przy braku dodatkowych danych punkty x rozmieszczone są w jednakowych odległościach na osi x. Mimo, że graficznie podobny do waveform chart, ale może wyświetlić wiele wartości danych (y) w tej samej chwili.



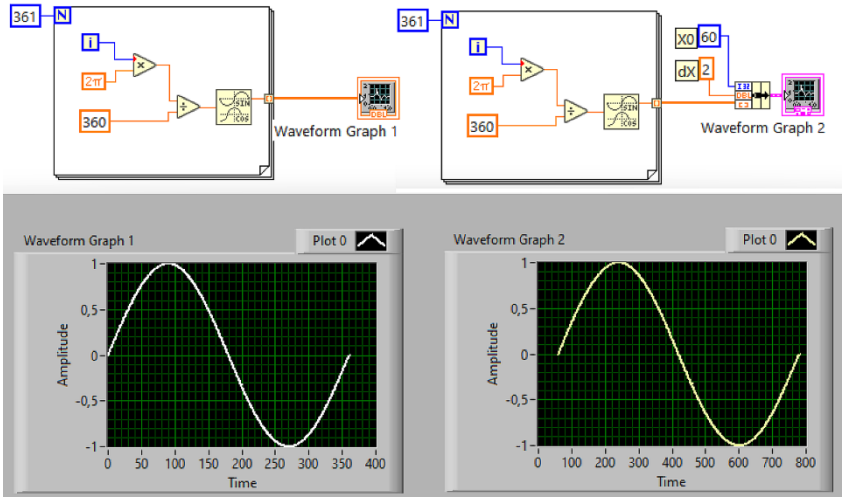
Rys. 3.17. Wykres waveform graph z częścią arkusza właściwości

Cechy charakterystyczne dla Waveform graph:

- bardziej „zasobożerny” niż chart,
- akceptuje wartości w postaci tablicowej lub dynamicznej (dane, czas z Express VI),
- podlega podobnemu formatowaniu jak Waveform chart (brak niektórych funkcji, takich jak np. Update Mode, History Length),
- obsługuje kursory (wskaźniki) na obszarze wykresu. Możliwe jest odczytywanie położenia kursorów na wykresie (przez użytkownika lub programowo), co pozwala na prowadzenie analizy wartości wyświetlanych przebiegów.

Przebieg w postaci pojedynczej serii danych zostanie wyświetlony jako przebieg z domyślnym zerowym punktem początkowym i domyślnym skokiem równym jeden (zmienna wejściowa zadeklarowana jako tablica 1D) (rys. 3.18).

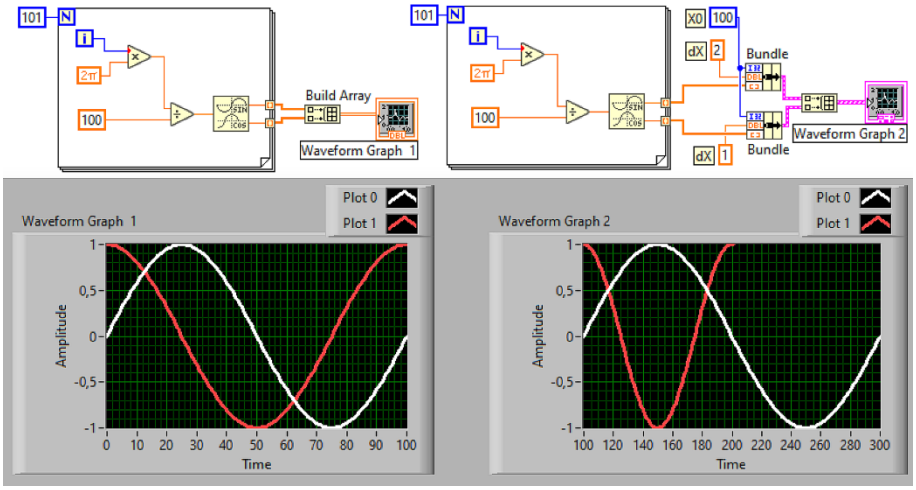
Waveform graph daje możliwość czasowego odwzorowania parametrów przebiegu. Do zadeklarowania parametrów wykorzystuje się klaster z jednowymiarową tablicą danych (funkcja Bundle). Kolejność danych w klastrze powinna odpowiadać punktowi początkowemu, interwałowi prezentacji i wartościom zmiennej wejściowej (rys. 3.18).



Rys. 3.18. Obsługa Waveform graph przy kształtowaniu danych wejściowych przebiegów

Przy prezentacji kilku przebiegów, należy połączyć ją w jedną strukturę tablicy. Najprostsze wydaje się połączenie danych tablicowych za pomocą funkcji tablicowej Build Array (rys. 3.19).

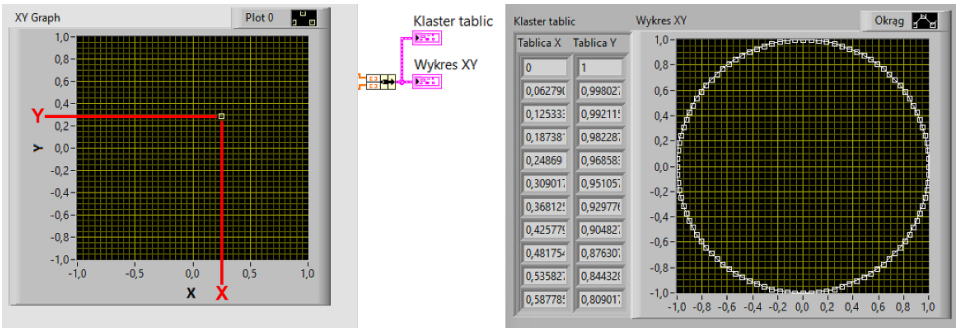
W przypadku kilku przebiegów, każdy z nich może mieć zróżnicowane parametry czasowe (punkt początkowy, interwał pomiaru). W takim przypadku do konfiguracji parametrów czasowych używa się funkcji Bundle, dalej to połączenie danych klastrowych za pomocą Build Array (rys. 3.19).



Rys. 3.19. Obsługa Waveform graph przy prezentacji kilku przebiegów

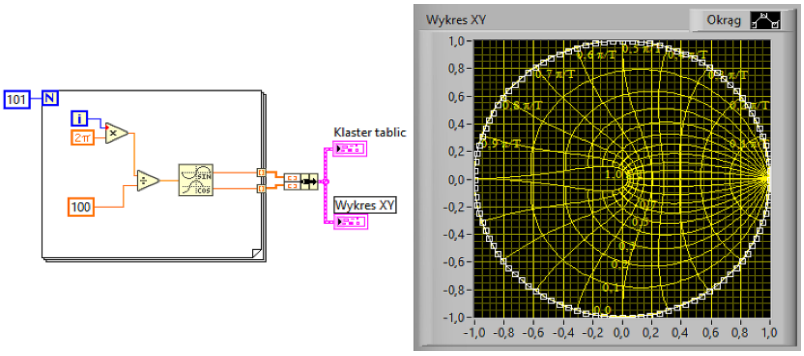
3.2.7. Wykres XY (XY graph)

Wykres XY domyślnie, w układzie kartezjańskim, wyświetla punkt o koordynatach x i y przypisanych jego wejściu (rys. 3.20). Brak wyboru znacznika dla punktu lub charakteru krzywej przebiegu skutkuje brakiem przebiegu na wykresie, mimo przesyłania danych do kontrolki wykresu. Możliwe jest skonfigurowanie wykresu do pracy na płaszczyznach Nyquista (np. do określania stabilności układu), płaszczyznach S i Z (rys. 3.21).



Rys. 3.20. XY graph – zasada prezentacji przebiegu

Przykładowy wykres kołowy w układzie kartezjańskim z zaznaczonymi liniami płaszczyzny Nyquista.



Rys. 3.21. Przykładowy wykres XY w z zaznaczonymi liniami płaszczyzny Nyquista

Przy wyświetlaniu przebiegów wykres XY akceptuje dane w postaci:

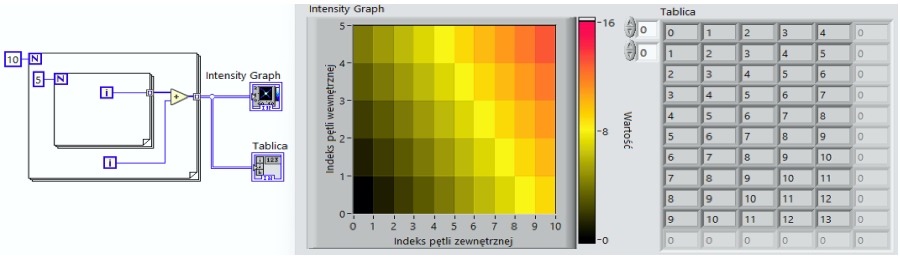
- klastra tablic zmiennych x i y,
- tablicy punktów, w których punktem jest klaster zawierający zmienne x i y,
- tablicy zmiennych zespolonych, z których część rzeczywista przypisana jest osi x, zaś urojona osi y.

3.2.8. Wykresy natężenia (intensity graph, chart)

W odniesieniu do wykresu typu intensity graph zasada wyświetlania danych polega na tym, że dwie zmienne (indeksy w tablicy danych) definiują położenie punktu, zaś trzecia wartość reprezentowana jest za pomocą jednego z 256 kolorów.

Cechy charakterystyczne dla wykresów natężenia:

- zastosowanie – wykresy ukształtowania terenu, natężenia pola, rozkładu temperatury, …,
- łatwa konfiguracja znaczników i barw odpowiadających wartościom poprzez menu kontekstowe,
- w przypadku błędnej interpretacji danych (wiersze traktowane jako kolumny) tablicowych wymagane jest ich przekształcenie funkcją Transpose Array,
- pomocnym narzędziem przy formatowaniu danych wykresów jest Express VI – Build XY Graph.

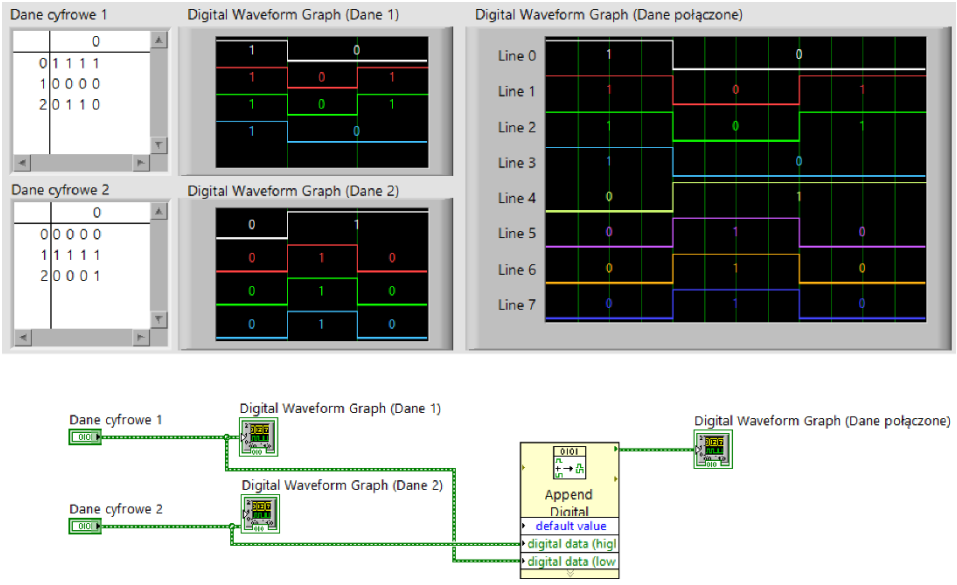


Rys. 3.22. Wykresy typu intensity graph

Współrzędne x i y opierają się na indeksach tablicy (rys. 3.22). Wartość odzwierciedlana kolorem konfigurowana jest za pomocą belki barw.

3.2.9. Digital Graphs

Wykres typu Digital Waveform Graph prezentuje wejściowe dane binarne w postaci logicznej, waveform (przebieg czasowy) lub tablicowej. Domyślnie wykres wyświetla dane binarne w postaci linii magistralnych (rys. 3.23). W tym przypadku ważne jest uwzględnienie, które z bitów są mniej (bardziej) znaczące. W przypadku formatowania danych cyfrowych do postaci przebiegu czasowego zmiennych cyfrowych przydatny jest węzeł funkcji Build Waveform. W takim przypadku wymagane jest dodatkowe zdefiniowanie początku przebiegu (start time) i interwału pomiędzy kolejnymi wartościami (dt).



Rys. 3.23. Przykład łączenia danych cyfrowych (Append Digital Signals.vi) i prezentacji na wykresie Digital Waveform Graph

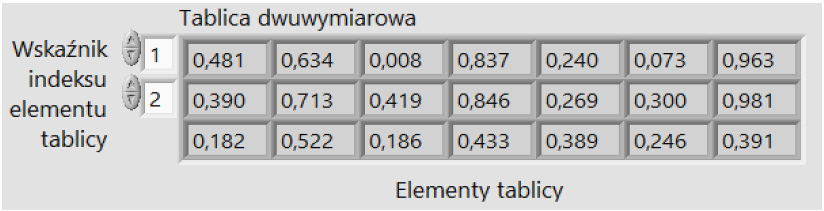
3.2.10. Tablice. Funkcje tablicowe

Środowisko LabVIEW posiada zdolność organizowania grup zmiennych tego samego lub różnych typów danych w pojedyncze obiekty. Takie grupowanie dostarcza możliwości łatwego operowania na grupach zmiennych (wartości) oraz powoduje, że upraszcza się kod programu [14].

Pod pojęciem tablicy rozpatrujemy zbiór elementów (zmiennych) tego samego typu (liczbowych, logicznych, łańcuchowych, przebiegów czasowych (waveform), klastrów) (rys. 3.24). Nie ma możliwości tworzenia tablic zawierających tablice. Podstawowe zastosowanie tablic to gromadzenie danych z powtarzalnych operacji odczytu, obliczeń, pomiarów (magazynowanie serii wartości zmiennych np. z pętli tj. jedno wykonanie pętli = jedna wartość).

Tablice charakteryzują poniższe cechy, takie jak:

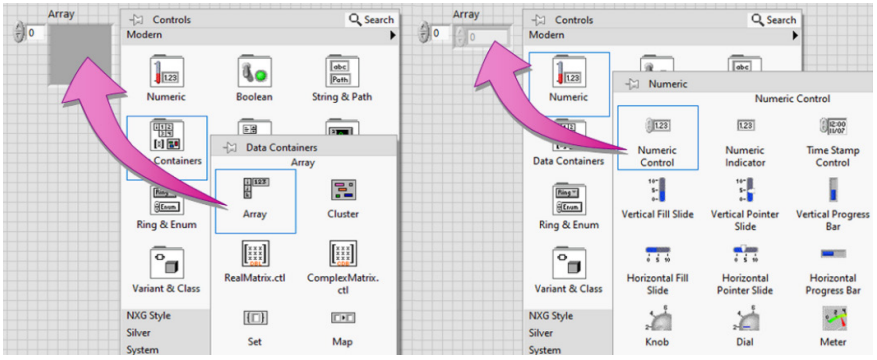
- możliwość magazynowania do (2^{31}-1) elementów na jeden wymiar tablicy (o ile pozwoli pamięć jednostki obliczeniowej),
- indeksowanie tablic od zera. Ostatni element jednowymiarowej tablicy *n*-elementowej to *n*-1. Indeks prezentowany we wskaźniku indeksu tablicy odnosi się do pierwszego widocznego elementu tablicy.



Rys. 3.24. Prosta tablica dwuwymiarowa typu double precision

Proces zamieszczania tablicy na panelu użytkownika to operacja dwustopniowa (rys. 3.25). Obejmuje ona:

- zamieszczenie powłoki tablicy,
- wprowadzenie do tablicy regulatorów lub wskaźników.



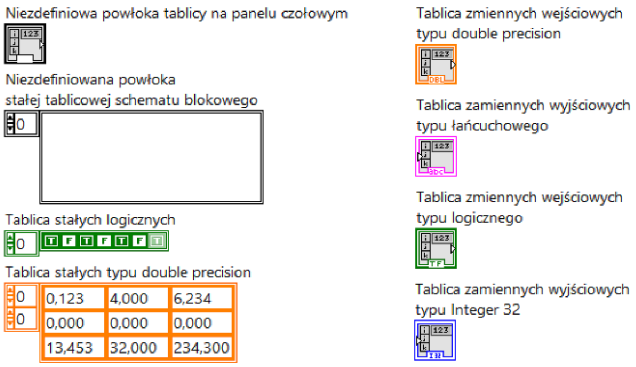
Rys. 3.25. Dwuetapowy proces zamieszczania tablicy na panelu

Elementy tablicy mogą być typu liczbowego, logicznego, łańcuchowego, klastrowego (regulatory lub wskaźniki) oraz mogą stanowić ścieżki dostępu. Pierwszy wprowadzony element określa typ danych tablicy (czarny symbol tablicy przyjmuje barwę zależną od typu zamieszczonych w tablicy zmiennych). Jeżeli obiektami tablicy są wykresy, to ich typ danych nie może być tablicą.

Większość operacji kształtowania i formatowania tablic przeprowadza się przez menu kontekstowe lub przez elementy graficzne symbolizujące tablicę i tak np. dodawanie wymiaru tablicy – menu kontekstowe, zmiana wielkości wyświetlanego obszaru tablicy mysz. Przy operacjach przeprowadzonych za pomocą wskaźnika myszy nale-

ży zwracać uwagę, czy formatowaniu podlega tablica, element wewnętrzny czy indeks tablicy. Dodawanie stałych tablicowych na diagramie programu można prowadzić (rys. 3.26):

- jak przy zwykłej tablicy – procedurą dwustopniową,
- przez przekształcenie tablicy we/we w stałą tablicową.

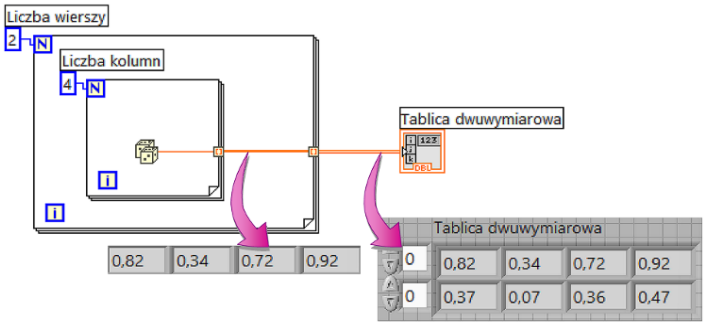


Rys. 3.26. Widok ikon stałych i zmiennych tablicowych w schemacie blokowym

Wypełnianie wartościami tablicy bardzo często stanowi wynik działania programu w pętli. W takim przypadku należy pamiętać o funkcji indeksowania (autoindeksowania), czyli samoczynnego indeksowania wartości otrzymanych z pętli. Autoindeksowanie jest domyślne dla pętli For, zaś niedomyślnie dla pętli While.

Przy obsłudze tablic dwuwymiarowych za pomocą dwóch pętli – pętla wewnętrzna tworzy elementy kolumn, pętla zewnętrzna – elementy wierszy (rys. 3.27). Występowanie autoindeksowania można rozpoznać przez:

- oznaczenie tunelu pętli przez nawias kwadratowy,
- grubszą linię typu danych,
- wynik przekazywany w każdej iteracji.

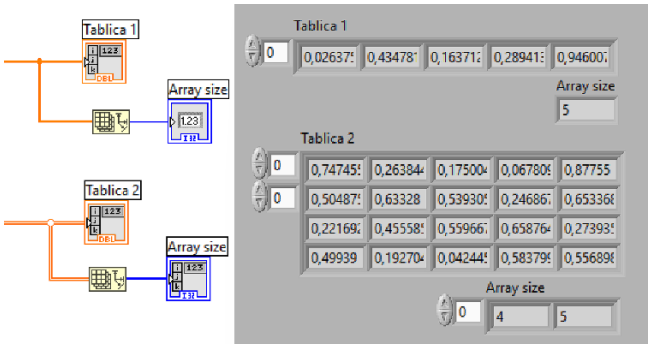


Rys. 3.27. Generowanie tablicy dwuwymiarowej za pomocą pętli

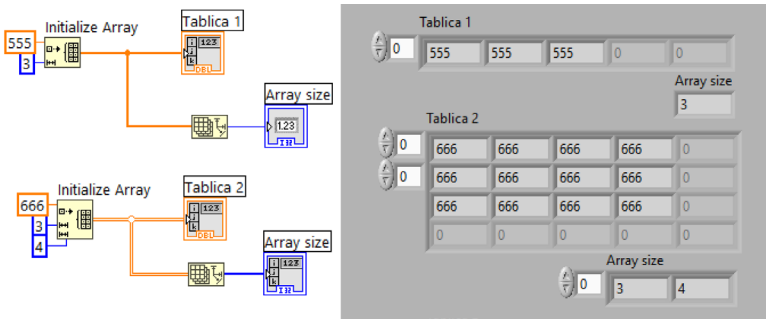
W tabeli 3.1. zamieszczono opis często stosowanych, wybranych funkcji tablicowych.

Tab. 3.1. Charakterystyka podstawowych funkcji operujących na tablicach

Funkcja	Opis
Array Size	zwraca liczbę elementów w każdym z wymiarów tablicy (rys. 3.28)
Initialize Array	tworzy n-elementową tablicę o zadanej wartości elementów (rys. 3.29)
Build Array	dołącza elementy do istniejącej tablicy. Domyślnie stosowane łączenie (dołączanie na końcu) elementów tablic. Możliwa jest modyfikacja domyślnego dołączania (konkatenacji) poprzez menu kontekstowe funkcji Build Array i polecenie Concatenate Inputs (rys. 3.30)



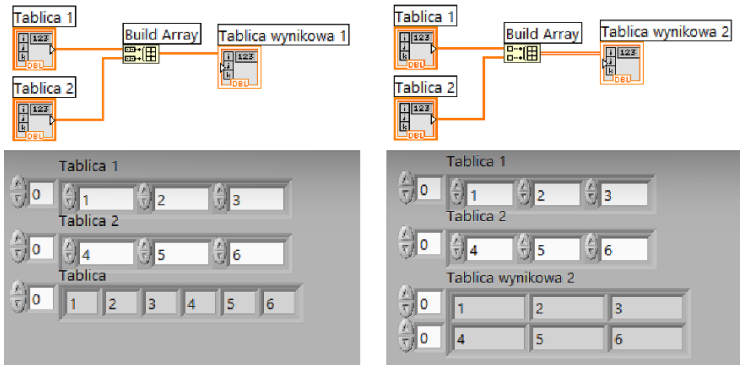
Rys. 3.28. Przykład działania funkcji Array Size



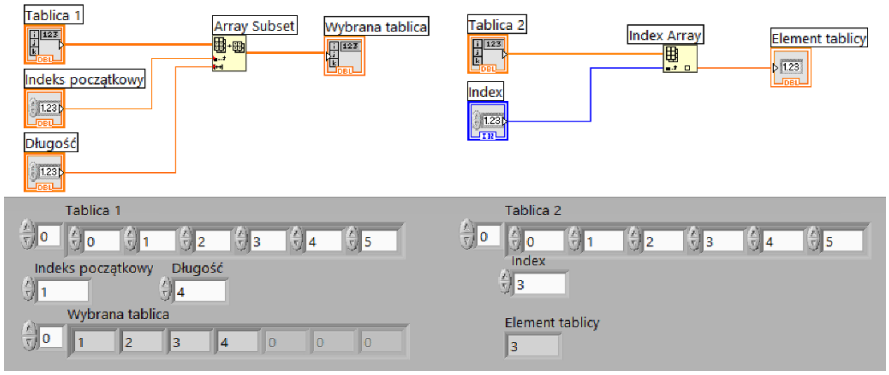
Rys. 3.29. Przykład działania funkcji Initialize Array

Tab. 3.2. Charakterystyka podstawowych funkcji operujących na tablicach

Funkcja	Opis
Array Subset	zwraca część tablicy, zaczynając od parametru Array index o długości Array length (stosuje indeksowanie od zera) (rys. 3.31)
Index Array	zwraca element(y) tablicy o podanym indeksie(ach) (rys. 3.31)

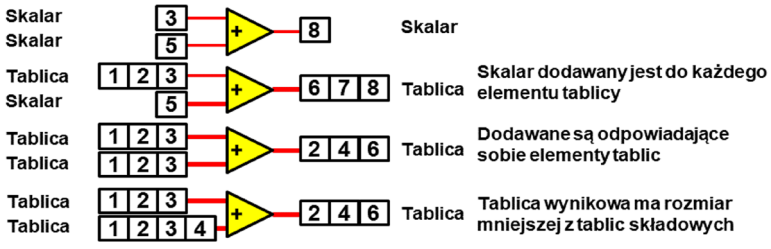


Rys. 3.30. Przykład działania funkcji Build Array



Rys. 3.31. Przykład działania funkcji Array Subset i Index Array

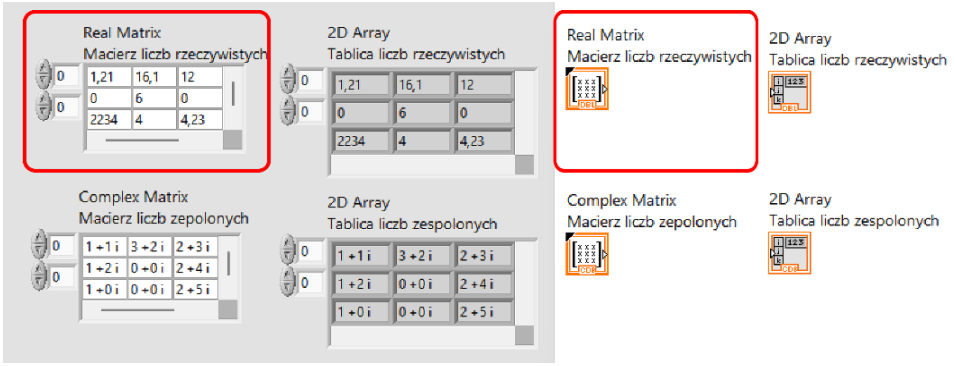
Tablice charakteryzuje polimorfizm, co w praktyce oznacza możliwość łączenia różnych struktur danych (tutaj wielkości skalarnych i tablic), w zakresie jednego sposobu przetwarzania tej samej funkcji programowej. Polimorfizm statyczny (inaczej programowanie generyczne) oznacza stosowalność tej samej procedury/funkcji do różnych typów argumentów. Przykłady polimorficznego działanie funkcji na zmiennych tablicowych zamieszczono na rysunku 3.32.



Rys. 3.32. Polimorficzne działanie funkcji na zmiennych tablicowych

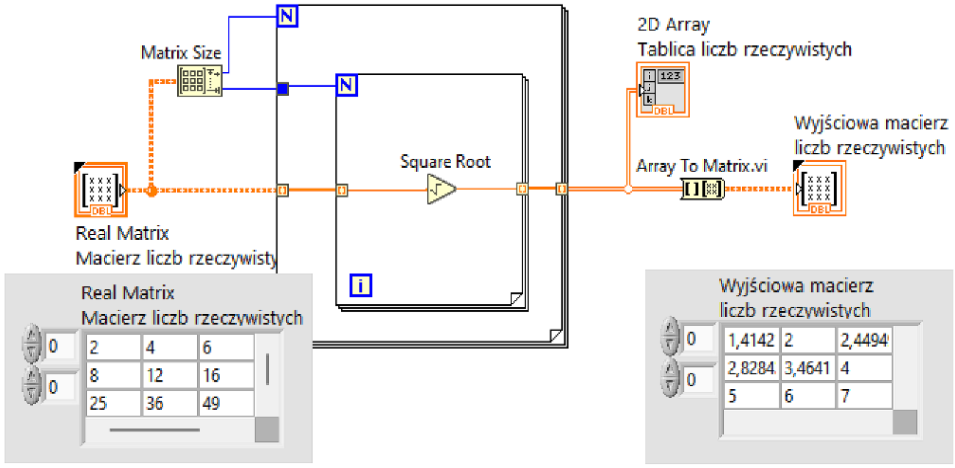
W środowisku LabVIEW funkcjonuje specjalny predefiniowany typ tablicy dwuwymiarowej o nazwie Real Matrix (rys. 3.33). Cechy charakterystyczne tej tablicy to:

- Real Matrix jest tablicą typu liczbowego (double precision floating point) lub liczb zespolonych tego samego typu,
- Real Matrix przeznaczona jest do wykonywania typowych działań algebry liniowej,
- macierze Real Matrix można łączyć w klastry.



Rys. 3.33. Różnice w prezentacji macierzy Real Matrix i tablic na panelu i w kodzie programu

Cechą specyficzną macierzy Real Matrix jest to, że autoindeksowanie stosowane w pętlach automatycznie przekształca macierz w dwuwymiarowe tablice (rys. 3.34). Na rysunku 3.34 zamieszczono funkcję Array To Matrix, która pozwala na przywrócenie przekształconej tablicy do postaci macierzy.



Rys. 3.34. Konwersja macierzy Real Matrix do postaci tablicy i przywrócenie macierzy

3.2.11. Klastry. Funkcje klastrowe

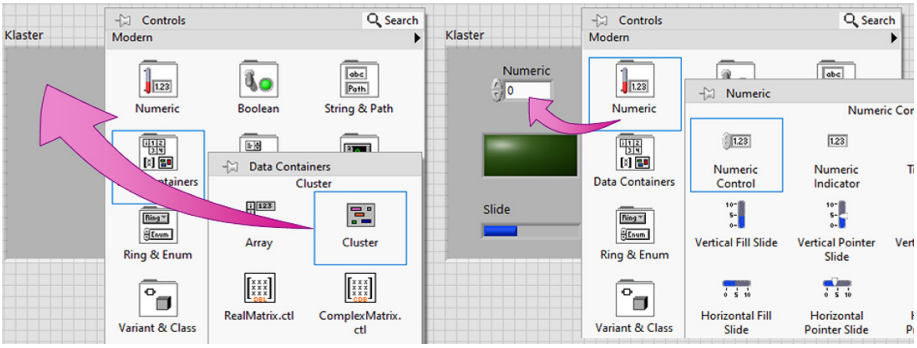
Klaster stanowi strukturę grupującą dane, podobną do struktury tablicowej. Różnica polega na tym, że dane grupowane w jednym obiekcie mogą być różnego typu. W odniesieniu do klastrów można stosować analogiczne podejście jak do przewodu wielożyłowego – jest to jeden główny przewód zawierający wiele różnych sygnałów [14].

Struktura klastrowa z LabVIEW wykazuje podobieństwo z obiektami record (Pascal), struct (C). Cechy charakterystyczne klastrów:

- wszystkie grupowane w klastrze elementy powinny mieć podobny charakter pod kątem kierunku przesyłania wartości (zmienne wejściowe – controls, zmienne wyjściowe – indicators),
- klaster pozwala przełamać ograniczenie liczby konektorów panelu czołowego (np. dla podprogramu (SubVI) (maksymalnie 28 we/wy),
- ułatwia doprowadzenie sygnału dla wykresów,
- klaster (zmienna klastrowa) powoduje, że kod programu staje się bardziej przejrzysty,
- powszechnie stosowanym klastrem jest klaster błędu (Error cluster), sterujący wykonywaniem programu,
- typowa barwa linii klastrów – różowa, klastry błędu są ciemnożółte.

Proces deklarowania (wstawiania) klastra jest dwuetapowy (jak w przypadku tablicy) i polega na (rys. 3.35):

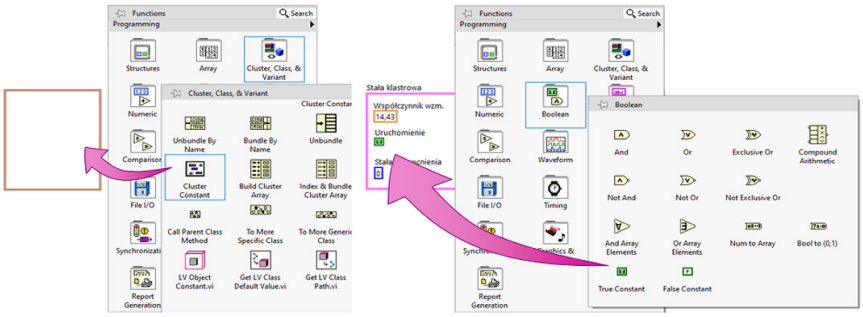
- wstawieniu obszaru klastra na panelu czołowym,
- wprowadzeniu elementów do klastra.



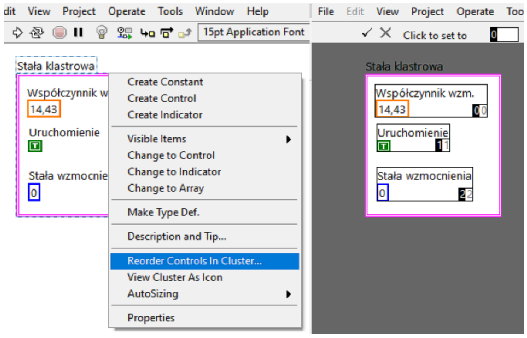
Rys. 3.35. Dwuetapowy proces wstawiania zmiennej klastrowej

Odpowiednikiem klastra z panelu czołowego jest stała klastrowa stosowana w kodzie programu. Można ją wykorzystywać równocześnie na wielu obiektach (np. skalowania wielu zmiennych pochodzących z procesu) (rys. 3.36).

Typ danych wchodzących/wychodzących z klastra musi się zgadzać z typem zamieszczonych w nim obiektów. Inaczej mówiąc, łącząc dwa klastry zachowujemy zgodność, jeśli elementy w klastrze reprezentują ten sam typ danych i występują w tej samej kolejności. Zmianę kolejności na pożądaną realizuje się przez menu kontekstowe i polecenie Reorder Controls in Cluster (rys. 3.37).



Rys. 3.36. Zamieszczanie powłoki stałej klastrowej i deklaracja kolejności obiektów w klastrze

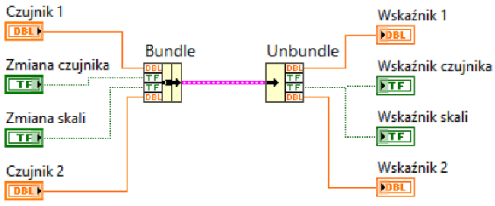


Rys. 3.37. Deklaracja kolejności obiektów w stałej klastrowej

Poniżej zamieszczono opis funkcji pozwalających na łączenie/rozłączanie pojedynczych zmiennych w zmienną (strukturę) klastrową (rys. 3.38 i 3.39).

Tab. 3.3. Charakterystyka funkcji klastrowych bazujących na kolejności elementów

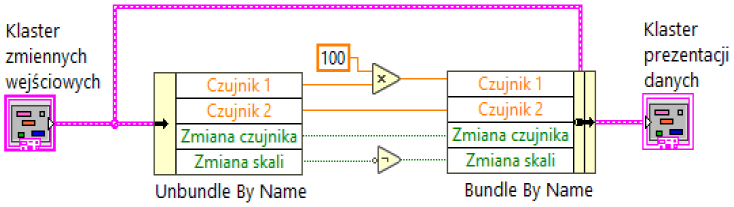
Funkcja	Opis
Bundle (rys. 3.38)	• łączenie danych w formę klastrową • kolejność tworzonego klastra zgodna z kolejnością przyłączania danych
UnBundle (rys. 3.38)	• rozłączanie z formy klastrowej • kolejność zgodna z kolejnością zmiennych w klastrze (przyłączania danych)



Rys. 3.38. Funkcje klastrowe Bundle i UnBundle

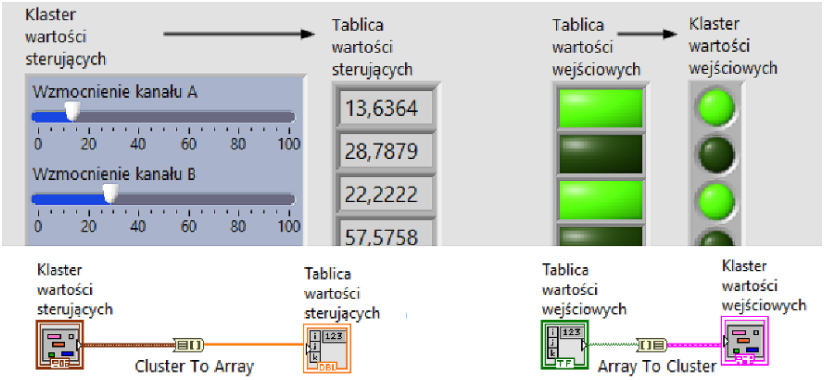
Tab. 3.4. Charakterystyka funkcji klastrowych bazujących na etykietach elementów

Funkcja	Opis
BundleByName (rys. 3.39)	• łączenie danych w formę klastrową • modyfikowanie danych klastrowych w oparciu o nazwę zmiennej (danej)
UnBundleByName (rys. 3.39)	• rozłączanie z formy klastrowej • rozłączanie postaci klastrowej z oznaczeniem nazwy dla konkretnego źródła danych



Rys. 3.39. Funkcje klastrowe BundleByName i UnBundleByName

W przypadku grup obiektów zmiennych, należących tego samego typu (ten sam typ danych, ten sam charakter zmiennych np. wejściowe), można konwertować klastry do postaci tablic i odwrotnie.



Rys. 3.40. Klastrowe funkcje konwersji

Funkcja Cluster to Array konwertuje klaster elementów tego samego typu danych na jednowymiarową tablicę danych tego samego typu. Funkcja Array to Cluster konwertuje jednowymiarową tablicę elementów do postaci klastra. Konieczna jest deklaracja rozmiaru klastra (domyślnie 9 elementów, maksymalnie 256). Deklarację przeprowadza się za pomocą menu kontekstowego węzła Array To Cluster i polecenia Cluster Size.

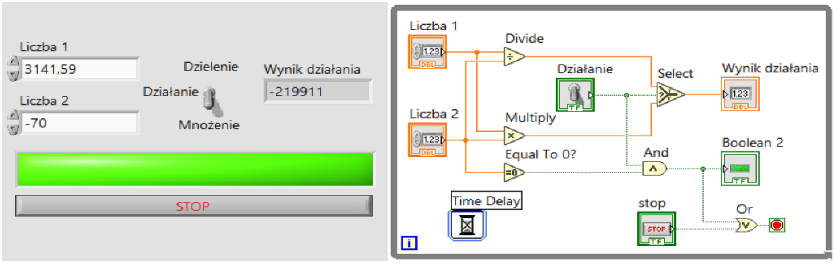
3.3. Zadania

3.3.1. Pętla While – zapewnienie ciągłości działania programu

Cel zadania: nabycie umiejętności stosowania pętli While do zapewnienia ciągłej pracy programu. Utrwalenie sposobu korzystania i konfigurowania Express VI.

Zakres zadania: utworzenie programu, którego zadaniem będzie realizowanie funkcji mnożenia lub dzielenia, w zależności od decyzji podjętej przez użytkownika. Ciągłość działania programu zapewni pętla While. Program będzie kończył pracę na podstawie decyzji użytkownika lub w przypadku próby wykonania dzielenia przez zero.

1. Uruchomić środowisko LabVIEW z pustym plikiem programu (Blank.vi). Celem kolejnych działań będzie utworzenie panelu czołowego wraz z kodem zaprezentowanym na kolejnych rysunkach.



2. **Zamieszczanie i formatowanie kontrolek i wskaźników panelu czołowego**. Z głównej palety panelu czołowego Controls wybrać subpaletę Numeric, a z niej element Numeric Control i zamieścić go na panelu. Kontrolka Numeric stanowić będzie zmienną wejściową Liczba 1. Operację wstawiania powtórzyć dla kontrolki Liczba 2 (Controls /Numeric /Numeric Control).
3. Z głównej palety panelu czołowego Controls wybrać subpaletę Numeric, a z niej element Numeric Indicator i zamieścić go na panelu. Wskaźnik Numeric stanowić będzie zmienną wyjściową Wynik działania.
4. Z głównej palety panelu czołowego Controls wybrać subpaletę Boolean, a z niej przełącznik Vertical Toggle Switch i zamieścić go na panelu.
5. Z głównej palety panelu czołowego Controls wybrać subpaletę Boolean, a z niej element Stop Button i zamieścić go na panelu.
6. Z głównej palety panelu czołowego Controls wybrać subpaletę Boolean, a z niej element Square LED i zamieścić go na panelu. Aktualnie na panelu czołowym powinno znajdować się 6 elementów.
7. Sformatować i uporządkować położenie elementów panelu czołowego:

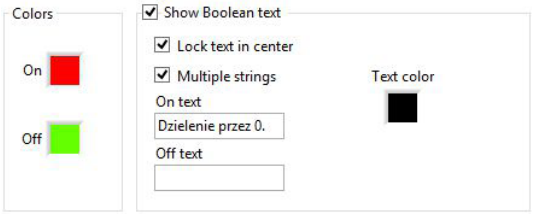
- kontrolka Numeric Control – modyfikacja etykiet z domyślnych na Liczba 1 i Liczba 2 (zmiana położenia, zmiana rozmiaru),
- wskaźnik Numeric Indicator – modyfikacja etykiety z domyślnej (np. Numeric 3) na Wynik działania (zmiana położenia, zmiana rozmiaru),
- przełącznik Vertical Toggle Switch – modyfikacja etykiety tak, aby zawierała tekst Działanie, zmiana rozmiaru i położenia,
- element Square LED – usunięcie etykiety (menu kontekstowe, polecenie Visible Items, usunąć check mark towarzyszący Label), zmiana rozmiaru i położenia,
- element Stop Button – usunięcie etykiety (menu kontekstowe, polecenie Visible Items, usunąć check mark towarzyszący Label), zmiana rozmiaru i położenia,
- bezpośrednio na panelu wprowadzić teksty Dzielenie i Mnożenie, a następnie ustawić je w otoczeniu przełącznika Działanie.

8. Ponieważ bezpośrednio po uruchomieniu programu zmienne środowiska LabVIEW przyjmują wartości zerowe, należy zapobiec sytuacji dzielenia przez zero, czyli zadeklarować domyślną wartość różną od zera. Taka sytuacja dotyczy zmiennej wejściowej Liczba 2, która będzie dzielnikiem. W celu zadeklarowania innej wartości domyślnej zmiennej Liczba 2, należy:

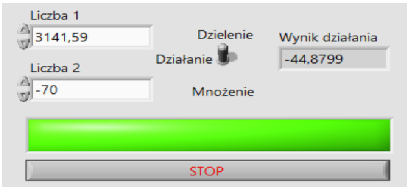
- wprowadzić do pola zmiennej wartość różną od zera (np. 4),
- wybrać z menu kontekstowego polecenie Data Operations, a następnie Make Current Value Default.

9. Celem wskaźnika Square LED jest sygnalizowanie, kiedy operacje mnożenia i dzielenia przebiegają bezproblemowo oraz kiedy zachodzi dzielenie przez zero. Należy skonfigurować wskaźnik tak, aby w jednym ze stanów był zielony (prawidłowe działanie), zaś w sytuacji dzielenia przez zero przyjmował czerwoną barwę i wyświetlał komunikat „Dzielenie przez 0. Koniec programu”. Aby uzyskać pożądany efekt należy:

- wyświetlić arkusz właściwości wskaźnika LED,
- na karcie Appearance w polu Colors zadeklarować kolor załączenia (On) jako czerwony, zaś kolor wyłączenia (Off) jako zielony,
- wybrać opcje Show Boolean text, Lock text in center, Multiple strings, a w polu On text wprowadzić łańcuch „Dzielenie przez 0. Koniec programu”,
- zatwierdzić zmiany przyciskiem [OK].



10. Zapisać plik programu pod nazwą (Z03 01 Mnozenie.vi) w lokalizacji wskazanej przez prowadzącego.



11. **Opracowanie kodu źródłowego programu**. Podstawą działania programu jest realizowanie funkcji mnożenia lub dzielenia, w zależności od decyzji podjętej przez użytkownika. Jednocześnie ciągłość funkcjonowania będzie zapewniała pętla While. Zakończenie działania pętli, a zarazem programu, będzie następować przez wybór przez użytkownika przycisku Stop lub w przypadku spełnienia warunku równoczesnego wystąpienia funkcji dzielenia oraz wartości zero w dzielniku. Ograniczenie monopolizacji zasobów systemu operacyjnego przez pętlę zostanie osiągnięte przez zamieszczenie w pętli funkcji opóźnienia (podprogram Express VI – Time Delay).
12. Przejść do okna schematu blokowego. Jeśli okno schematu blokowego nie jest aktywne, z menu Window wybrać polecenie Show Block Diagram lub skorzystać ze skrótu Ctrl + E. W oknie schematu powinny znajdować się ikony kontrolek zamieszczonych uprzednio na panelu czołowym.
13. Zamieszczanie funkcji logicznych. Z głównej palety Programming wybrać subpaletę Boolean, a z niej funkcje:

- Or,
- And.

14. Zamieszczanie funkcji porównawczej. Z głównej palety Programming wybrać subpaletę Comparison, a z niej funkcję Equal To 0?
15. Zamieszczanie funkcji matematycznych. Z głównej palety Programming wybrać subpaletę Numeric, a z niej funkcje:

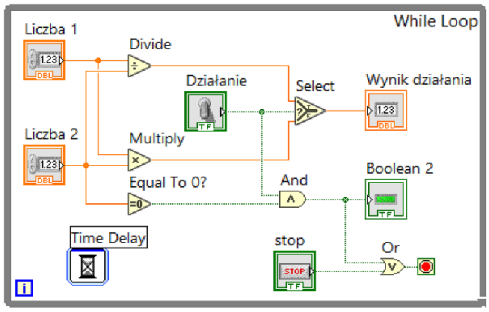
- Divide,
- Multiply.

16. Zamieszczanie funkcji porównawczych (wyboru). Z głównej palety Programming wybrać subpaletę Comparison, a z niej funkcje:

- Equal To 0?,
- Select.

17. Zamieszczanie podprogramu Express VI – Time Delay. Z głównej palety Programming wybrać subpaletę Timing, a z niej funkcję Time Delay. Bezpośrednio po zamieszczeniu Time Delay zostanie otworzony panel konfiguracyjny. Wprowadzić czas 0,1 sekundy i zatwierdzić zmiany przyciskiem [OK].

18. Zamieszczanie pętli While. Z głównej palety Programming wybrać subpaletę Structures, a z niej pętlę While Loop. Przy wstawianiu pętli wymagane jest otoczenie dotychczas zamieszczonych elementów schematu blokowego. Bezpośrednio po wybraniu z palety ikony pętli While, należy z wciśniętym przyciskiem myszy wybrać obszar od lewego górnego do prawego dolnego rogu schematu blokowego.
19. Wykonać połączenia pomiędzy ikonami zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).



20. Zapisać plik programu pod aktualną nazwą (031_Mnozenie.vi).

21. **Uruchamianie programu**. W oknie panelu czołowego uruchomić program w trybie pojedynczym za pomocą ikony Run. Za ciągłość funkcjonowania programu odpowiada pętla While. Przeprowadzić obserwację działania programu w przypadku:

- prowadzenia obliczeń przy zadeklarowaniu wartości wejściowych dodatnich, różnych od zera (mnożenie, dzielenie),
- prowadzenia obliczeń przy zadeklarowaniu wartości wejściowych dodatnich i ujemnych, różnych od zera,
- wyłączania programu za pomocą przycisku Stop,
- samoczynnego przerywania programu w przypadku spełnienia warunku równoczesnego wystąpienia funkcji dzielenia oraz wartości zero w dzielniku (Liczba 2).

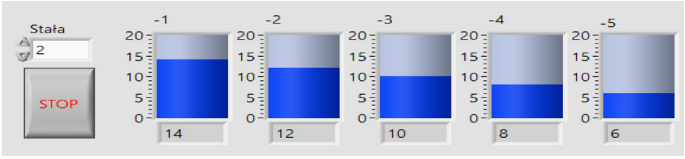
22. Zamknąć plik programu po zakończeniu testowania.

3.3.2. Rejestr przesuwny

Cel zadania:	wprowadzenie do sposobu funkcjonowania rejestru przesuwnego. Nabycie umiejętności obsługi rejestru przesuwnego w pętli.
Zakres zadania:	utworzenie programu, którego zadaniem będzie prezentacja sposobu przekazywania danych w komórkach rejestru przesuwnego. Program będzie kończył pracę na podstawie decyzji użytkownika lub w przypadku osiągnięcia przez pierwszą komórkę rejestru wartości 20.

1. Uruchomić środowisko LabVIEW z pustym plikiem programu.

2. **Zamieszczanie i formatowanie kontrolek i wskaźników panelu czołowego**. Z głównej palety panelu czołowego Controls wybrać subpaletę Numeric, a z niej element Numeric Control i zamieścić go na panelu. Kontrolka Numeric stanowić będzie zmienną wejściową o nazwie Stała. W każdej pętli suma będzie powiększała się o wartość zmiennej Stała.
3. Z głównej palety panelu czołowego Controls wybrać subpaletę Numeric, a z niej element Tank i zamieścić go na panelu.
4. Zmienić górny zakres skali wskaźnika Tank na 20 (edycja w miejscu, podwójne kliknięcie na maksymalnej wartości skali).
5. Wyświetlić wskaźnik liczbowy wskaźnika Tank (menu kontekstowe, polecenie Visible Items, polecenie Digital Display) i zamieścić do pod wskaźnikiem Tank. Zmienić nazwę wskaźnika Tank na „-1".



6. Skopiować czterokrotnie wskaźnik „-1" (Tank). Zmienić nazwy skopiowanym wskaźnikom na -2, -3, -4 oraz -5. Uporządkować wskaźniki zgodnie z rysunkiem 4. Położenie można zmieniać przez przeciąganie elementów do wymaganej pozycji. Precyzyjne wyrównanie i odstępy można uzyskać, stosując polecenie paska narzędzi z grup Align Objects oraz Distribute Objects.
7. Z głównej palety panelu czołowego Controls wybrać subpaletę Boolean, a z niej element Stop Button i zamieścić go na panelu. Usunąć etykietę przycisku (menu kontekstowe, polecenie Visible Items, usunąć check mark towarzyszący Label). Zmienić jego rozmiar i położenie na zbliżone do zamieszczonego na rysunku prezentującym panel.
8. Zapisać plik programu pod nazwą (032_Rejestr.vi) w lokalizacji wskazanej przez prowadzącego.

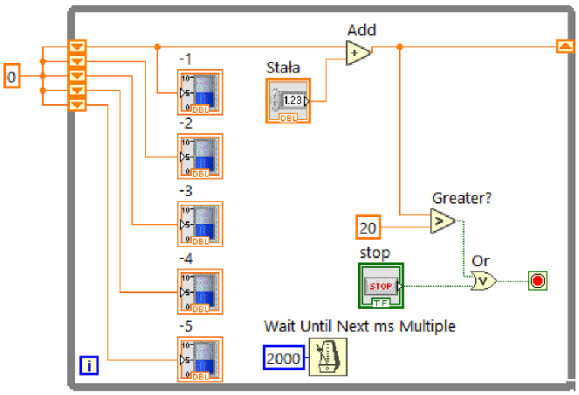
9. **Opracowanie kodu źródłowego programu**. Celem programu jest prezentacja przekazywania (przesuwania) wartości pomiędzy komórkami rejestru. Rejestr zostanie zamieszczony na pętli While. Zakończenie działania pętli, a zarazem programu, będzie następować poprzez wybór przez użytkownika przycisku Stop lub w przypadku, kiedy suma kolejnych wartości „Stałej" będzie równa lub większa 20. Ograniczenie monopolizacji zasobów systemu operacyjnego przez pętlę zostanie osiągnięte przez zamieszczenie w pętli funkcji opóźnienia Wait Until Next ms Multiple.

10. Przejść do okna schematu blokowego. W oknie schematu powinny znajdować się ikony kontrolek zamieszczonych uprzednio na panelu czołowym.

11. Zamieszczanie funkcji logicznej Or. Z głównej palety Programming wybrać subpaletę Boolean, a z niej funkcję Or.
12. Zamieszczanie funkcji porównawczej. Z głównej palety Programming wybrać subpaletę Comparison, a z niej funkcję Greater.
13. Zamieszczanie funkcji matematycznej dodawania. Z głównej palety Programming wybrać subpaletę Numeric, a z niej funkcję Add.
14. Zamieszczanie funkcji opóźnienia Wait Until Next ms Multiple. Z głównej palety Programming wybrać subpaletę Timing, a z niej funkcję Wait Until Next ms Multiple.
15. Zamieszczanie stałych liczbowych. Z głównej palety Programming wybrać subpaletę Numeric, a z niej stałą DBL Numeric Constant. Wymagane będą trzy stałe równe:

- 0 – do inicjacji rejestru,
- 20 – do wymuszania warunku zakończenia pętli,
- 2000 – opóźnienie czasowe 2000 ms funkcji Wait Until Next ms Multiple.

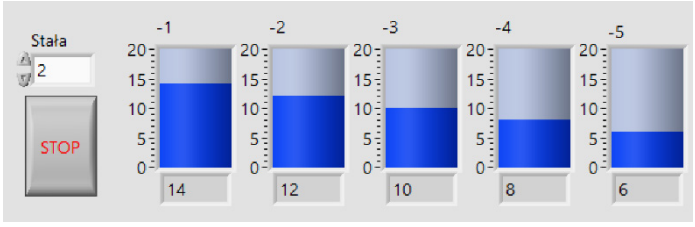
16. Zamieszczanie pętli While. Z głównej palety Programming wybrać subpaletę Structures, a z niej pętlę While Loop. Przy wstawianiu pętli wymagane jest otoczenie dotychczas zamieszczonych elementów schematu blokowego. Bezpośrednio po wybraniu z palety ikony pętli While, należy, z wciśniętym przyciskiem myszy, wybrać obszar od lewego górnego do prawego dolnego rogu schematu blokowego.
17. Zamieszczanie rejestru przesuwnego. Wybrać prawym przyciskiem myszy prawą krawędź pętli. Z menu kontekstowego wybrać polecenie Add Shift Register. W efekcie na prawej i lewej krawędzi zostanie zamieszczony czarny symbol rejestru. Symbol zmieni barwę po przyłączeniu zmiennej (stałej) do dowolnej komórki rejestru.
18. Poszerzanie historii rejestru przesuwnego. Wskazać wskaźnikiem myszy dolną krawędź symbolu rejestru na lewej krawędzi pętli. Wskaźnik myszy przyjmie kształt dwukierunkowej strzałki. Rozciągnąć symbol rejestru tak, aby zawierał 5 komórek.
19. Wykonać połączenia pomiędzy ikonami zgodnie z realizowanymi funkcjami programu (oraz rysunkiem)



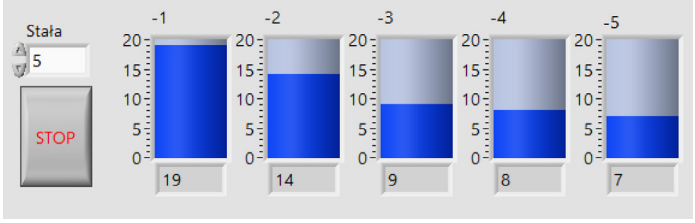
20. Zapisać plik programu pod aktualną nazwą (Z03 02 Rejestr.vi)

21. **Uruchamianie programu**. W oknie panelu czołowego wprowadzić rozsądną wartość zmiennej Stała. Uruchomić program w trybie pojedynczym za pomocą ikony Run. Przeprowadzić obserwację działania programu pod kątem przekazywania wartości kolejnym komórkom rejestru, w przypadku:

- niezmieniania wartości Stałej w czasie działania programu,



- zmieniania wartości Stałej w czasie działania programu,



- wyłączania programu za pomocą przycisku Stop przed jego samoczynnym wyłączeniem.

22. Zamknąć plik programu po zakończeniu testowania.

3.3.3. Wykres XY Graph

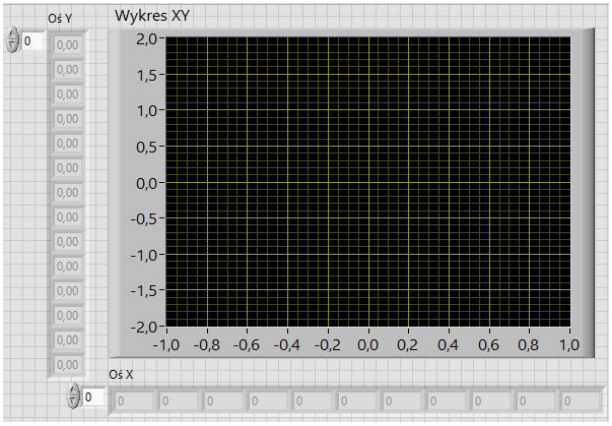
Cel zadania: zapoznanie z cechami i funkcjonowaniem obiektu wykresu XY Graph.

Zakres zadania: wykres typu XY Graph na swoim obszarze prezentuje punkt o zadeklarowanych w programie współrzędnych x i y. Wyróżnia go konieczność deklaracji dwóch współrzędnych, w przeciwieństwie do wykresów Waveform Chart i Waveform Graph, w których odcięta definiowana jest na podstawie kolejności lub czasu wprowadzanych wartości. Opracowywany program pozwoli na wyświetlenie wartości funkcji arc sinus w funkcji sinus dla kąta zmieniającego się w zakresie od 0 do 360 stopni.

1. Uruchomić środowisko LabVIEW z pustym plikiem programu.

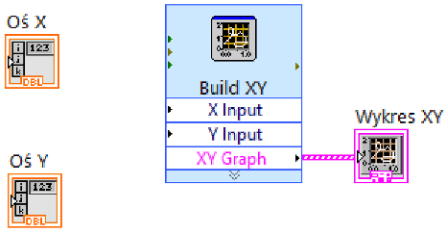
2. **Budowa panelu czołowego**. Z niżej wymienionych palet należy wybrać i sformatować poniższe elementy panelu:

- wykres XY Graph (palety Controls /Modern /Graph i obiekt Express XY Graph. Za pomocą poleceń kontekstowych usunąć: opisy osi X i Y, legendę wykresu (plot legend). Zmienić tytuł na Wykres XY,
- tablicę Array prezentującą wartości osi odciętych X. Zamieścić powłokę tablicy (kolejno palety Controls /Modern /Array, Matrix & Cluster i obiekt Array). Zmienić nazwę tablicy na Oś X,
- w tablicy zamieścić zmienną wyjściową Numeric Indicator (kolejno palety Controls /Modern /Numeric i obiekt Numeric Indicator). Zmienić rozmiar tablicy tak, aby była ułożona poziomo pod wykresem wzdłuż osi X,
- tablicę Array prezentującą wartości osi rzędnych Y. Zamieścić powłokę tablicy (kolejno palety Controls /Modern /Array, Matrix & Cluster i obiekt Array). Zmienić nazwę tablicy na Oś Y,
- w tablicy zamieścić zmienną wyjściową Numeric Indicator (kolejno palety Controls /Modern /Numeric i obiekt Numeric Indicator). Zmienić rozmiar tablicy tak, aby była ułożona pionowo obok wykresu, wzdłuż osi Y.



3. Zapisać plik programu pod nazwą (033_XY_graph.vi) w lokalizacji wskazanej przez prowadzącego.

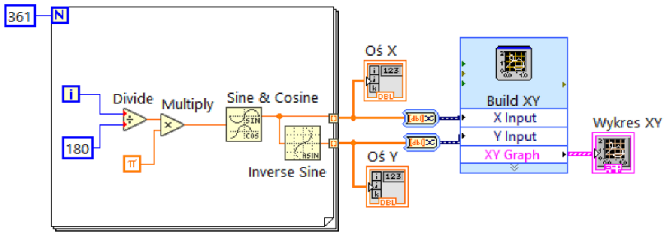
4. **Schemat blokowy programu**. Przejść do okna schematu blokowego.



5. Aplikacja realizuje operacje: przeliczenia wartości kąta w stopniach na radiany, obliczania wartości funkcji sinus i arc sin, powtarzania obliczeń tak, aby obliczenia objęły kąt 360° z interwałem 1° oraz prezentację danych na wykresie XY i w tablicach.
6. Zamieszczanie funkcji matematycznych. Z głównej palety Programming wybrać subpaletę Numeric, a z niej:

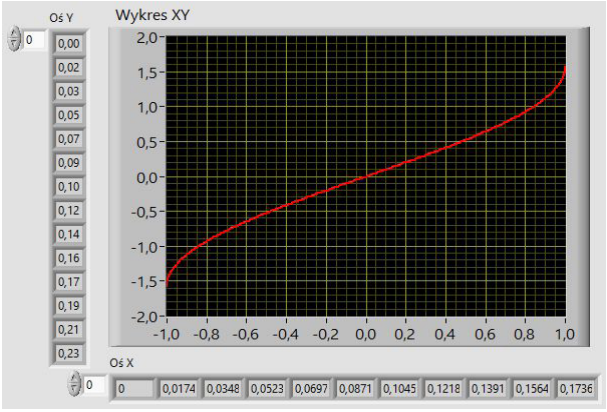
- funkcję mnożenia Multiply,
- funkcję dzielenia Divide,
- stałą Numeric Constant (dwa obiekty) i zadeklarować wartości 361 oraz 180,
- stałą π – subpalety Math & Scientifics Constants.

7. Zamieścić węzły funkcji Sine oraz Inverse Sine z subpalety Mathematics i kolejnych palet Elementary & Special Functions / Trigonometric Functions.
8. Zamieścić pętlę For (Functions /Programming /Structures /For Loop).
9. Wykonać połączenia pomiędzy ikonami zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).



10. Zapisać plik programu pod aktualną nazwą (033_XY_graph.vi).

11. **Uruchamianie programu**. Uruchomić program w trybie jednokrotnym. Wynikiem działania programu powinien być przebieg funkcji arc sinus w funkcji sinus dla kąta zmieniającego się w zakresie od 0 do 360 stopni. Jednocześnie w tablicach wyświetlane będą początkowe wartości zmiennych osi X i Y.



12. Zamknąć plik programu po zakończeniu testowania.

3.3.4. Wykres Intesity Graph

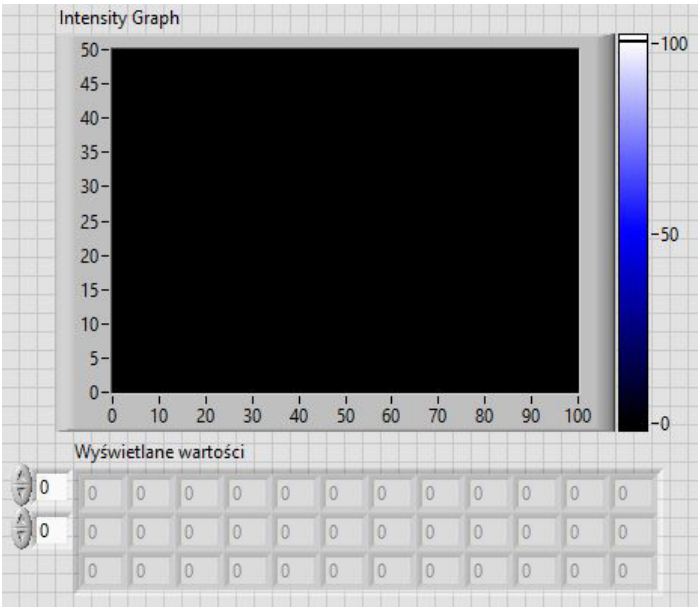
Cel zadania: zapoznanie z cechami i funkcjonowaniem obiektu wykresu Intensity Graph.

Zakres zadania: Wykres typu Intesity Graph na swoim obszarze prezentuje wartość w postaci barwy, zadeklarowanej na belce barw dla współrzędnych x i y, wynikających z indeksu położenia wartości w dwuwymiarowej tablicy. Opracowywany program pozwoli na wyświetlenie wartości z zakresu 0–10, wygenerowanych w pętli dla trzech zmiennych.

1. Należy uruchomić środowisko LabVIEW z pustym plikiem programu.

2. **Budowa panelu czołowego**. Z niżej wymienionych palet wybrać i sformatować poniższe elementy panelu:

- wykres Intesity Graph (palety Controls /Modern /Graph i obiekt Intesity Graph. Za pomocą poleceń kontekstowych usunąć: opisy osi X i Y (Visible Scale Label) oraz opis belki kolorów,



- tablicę Array prezentującą wartości zamieszczane na wykresie. Zamieścić powłokę tablicy (kolejno palety Controls /Modern /Array, Matrix & Cluster i obiekt Array). Zmienić nazwę tablicy na „Wyświetlane wartości",
- w tablicy zamieścić zmienną wyjściową Numeric Indicator (kolejno palety Controls /Modern /Numeric i obiekt Numeric Indicator),
- dodać drugi wymiar tablicy – menu kontekstowe wskaźnika indeksu z poleceniem Add Dimension,
- zmienić rozmiar tablicy tak, aby były widoczne trzy wiersze, a sama tablica zamieszczona była pod wykresem Intesity Graph.

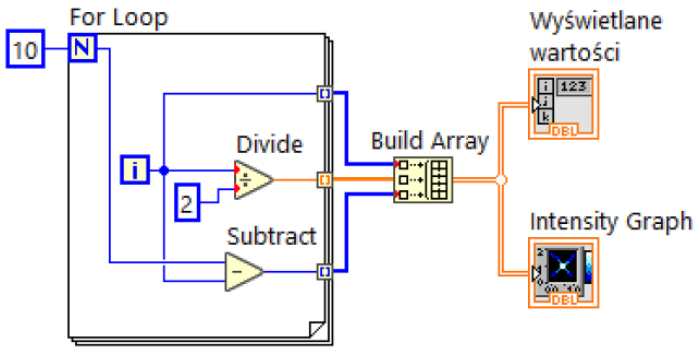
3. Zapisać plik programu pod nazwą (034_Intesity_Graph.vi) w lokalizacji wskazanej przez prowadzącego.

4. **Schemat blokowy programu**. Przejść do okna schematu blokowego.

5. W pętli aplikacji realizowane są operacje generowania przykładowych danych w postaci ciągu liczb stanowiących: a) iterację wykonywanej pętli, b) iterację pętli podzielonej przez dwa, c) liczbę reprezentującą różnicę pomiędzy zadeklarowaną liczbą pętli a „numerem" aktualnie wykonywanej pętli. Tak wygenerowane dane zestawiane są w postać zmiennej tablicowej, a ostatecznie prezentowane na wykresie Intensity Graph i w tablicy.
6. Zamieszczanie funkcji matematycznych. Z głównej palety Programming wybrać subpaletę Numeric a z niej:

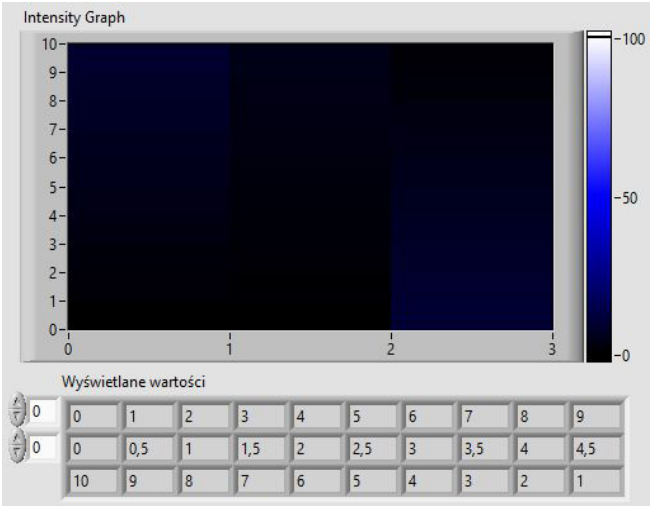
- funkcję odejmowania Subtract,
- funkcję dzielenia Divide,
- tałą Numeric Constant (2 obiekty) i zadeklarować wartości 10 oraz 2.

7. Zamieścić węzeł funkcji tablicowej Build Array z subpalety Programming /Array.
8. Zamieścić pętlę For (Functions /Programming /Structures /For Loop).
9. Wykonać połączenia pomiędzy ikonami zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).



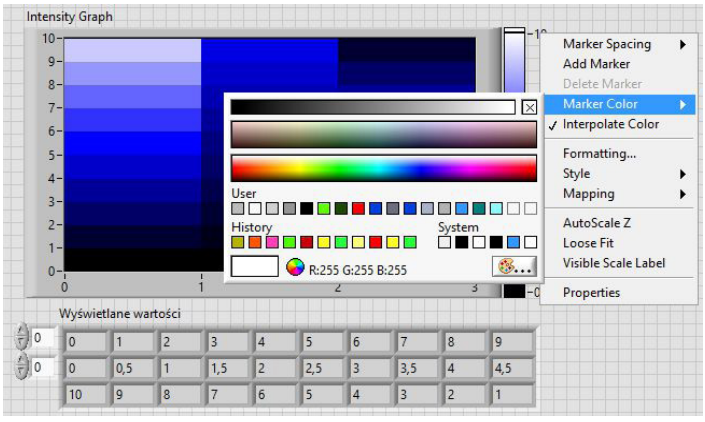
10. Zapisać plik programu pod aktualną nazwą (034_Intesity_Graph.vi).

11. **Uruchamianie programu**. Uruchomić program w trybie jednokrotnym. Wynik działania programu powinien być zbliżony do zamieszczonego na rysunku.



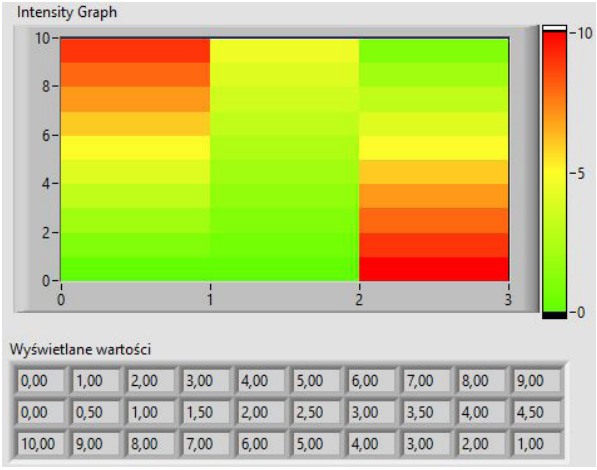
12. Modyfikowanie sposobu prezentacji danych.

- Zmiana zakresu prezentowanych wartości z 0–100 na 0–10. Przeprowadzenie edycji w miejscu wartości 100 (zmienić na 10).
- Zmiana barw przypisanych do wartości. Docelowo do wartości 0 ma być przypisana barwa zielona, do wartości 5 – barwa żółta, zaś do 10 – barwa czerwona. Zmiany dokonuje się za pomocą menu kontekstowego markera (np. liczby 10) i polecenia Marker Color.



13. Zapisać plik programu pod aktualną nazwą (034_Intesity_Graph.vi).

14. Przeprowadzić kilka dodatkowych prób działania programu ze zmodyfikowaną liczbą zadeklarowanych pętli (zmiana w kodzie programu) i zakresów prezentowanych danych (zmiana zakresu belki barw).



15. Zamknąć plik programu po zakończeniu testowania.

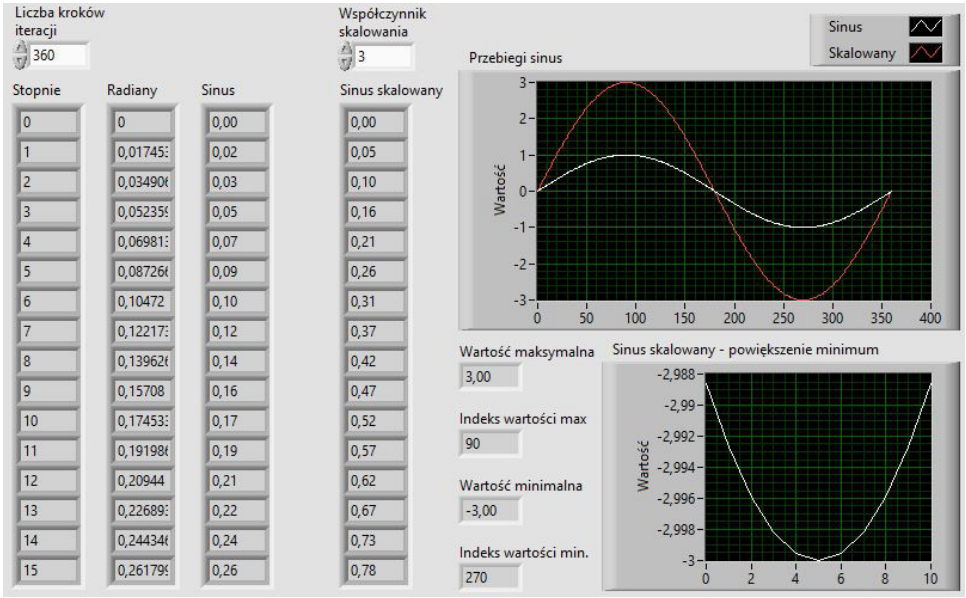
3.3.5. Tablice

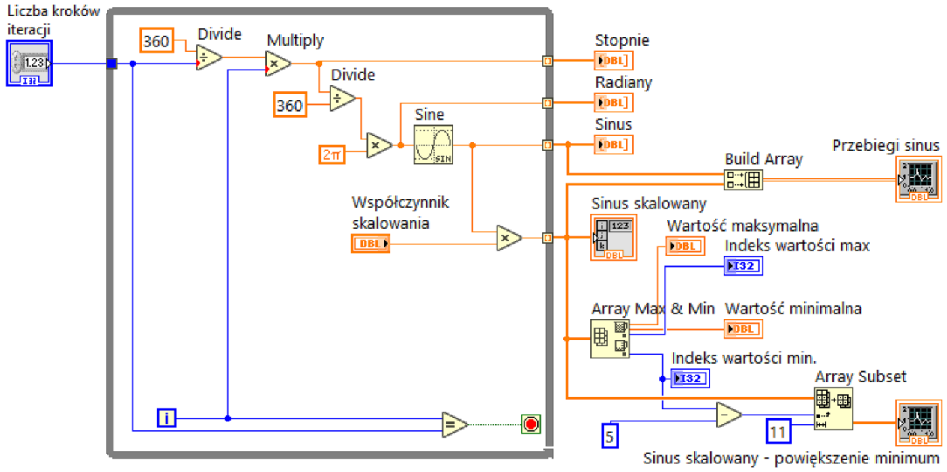
Cel zadania: pozyskanie podstawowych umiejętności obsługi tablic i macierzy, tj. obsługi obiektów na panelu czołowym oraz stosowania wybranych funkcji tablicowych.

Zakres zadania: opracowanie programu, którego zadaniem będzie wygenerowanie przebiegu sinus z zadanym interwałem, przeskalowanie go zadeklarowanym współczynnikiem oraz pozyskanie informacji o minimalnej wartości w tablicy wraz z prezentacją minimum przebiegu na wykresie.

1. Uruchomić środowisko LabVIEW z pustym plikiem programu (Blank.vi). Celem kolejnych działań, które zaprezentowano na następnych rysunkach, będzie utworzenie panelu czołowego wraz z kodem.

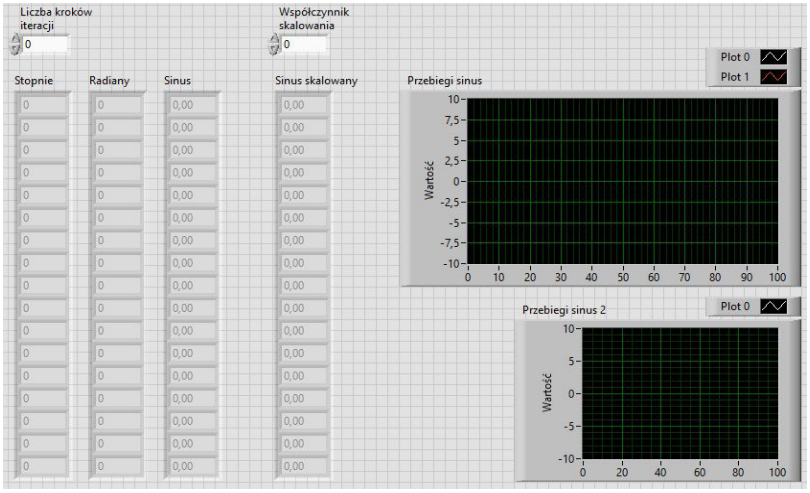
2. **Zamieszczanie i formatowanie kontrolek oraz wskaźników panelu czołowego**. Z głównej palety panelu czołowego Controls wybrać subpaletę Array, Matrix & Cluster, a z niej powłokę tablicy Array. Z subpalety Numeric wybrać wskaźnik Numeric Indicator i zamieścić go wewnątrz powłoki tablicy. Poszerzyć tablicę w pionie tak, aby prezentowała 10–15 wartości. Usunąć wskaźnik indeksu poprzez menu kontekstowe Visible Items /Index Display. Nadać tablicy nagłówek Stopnie.





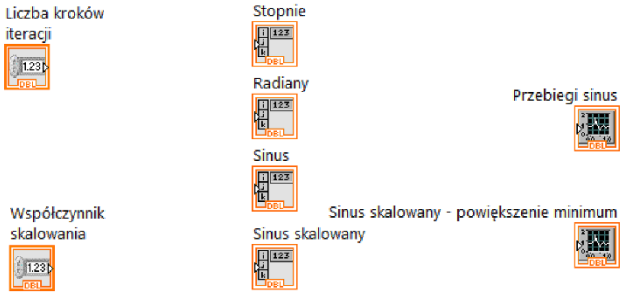
3. W analogiczny sposób utworzyć tablice Radiany, Sinus, Sinus skalowany. W przypadku zmiennych tablic Sinus i Sinus skalowany zmienić sposób prezentacji wartości na 2 liczby po przecinku, np. menu kontekstowe wartości z tablicy i dalej Display Format: Type = Floating Point, Digits = 2, Precision Type = Digits of precision, Hide trailing zeros = nieaktywne (odznaczone, opcja niewybrana).
4. Z głównej palety panelu czołowego Controls wybrać subpaletę Numeric, a z niej element Numeric Control i zamieścić go na panelu. Zmiennej nadać nazwę „Liczba kroków iteracji". Operację powtórzyć dla kontrolki wejściowej „Współczynnik skalowania".

5. Z subpalety Graph wybrać wykres Waveform Graph i zamieścić go na panelu czołowym. Zmienić jego nazwę na „Przebieg sinus”. Wstawianie powtórzyć dla wykresu „Sinus skalowany – powiększenie minimum”. Usunąć opisy osi odciętych oraz zmienić opis osi rzędnych na „Wartość”.



6. Zapisać plik programu pod nazwą (035_Sinus_skalowanie.vi) w lokalizacji wskazanej przez prowadzącego.

7. **Schemat blokowy programu**. Przejść do okna schematu blokowego. W oknie schematu powinny znajdować się ikony obiektów zamieszczonych uprzednio na panelu czołowym.

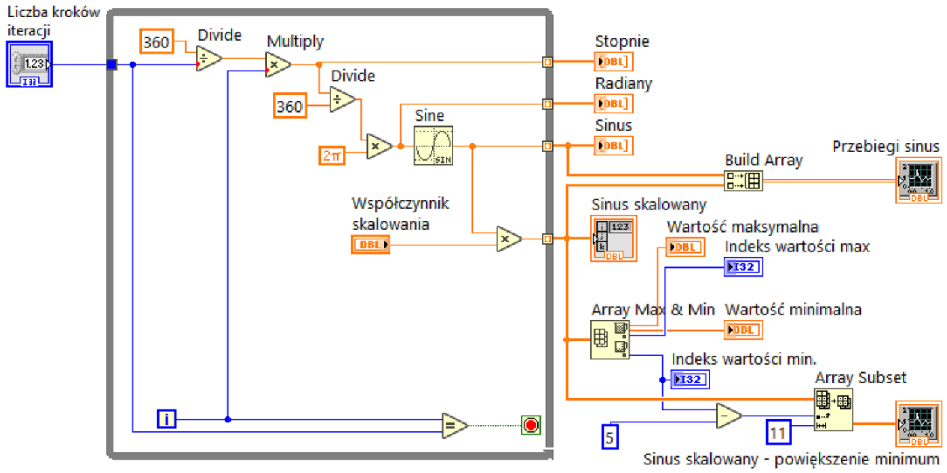


8. Aplikacja realizuje trzy podstawowe operacje: dopasowanie kroku obliczeń tak, aby obliczenia objęły kąt 360° wraz z przeliczeniem wartości kąta w stopniach na radiany, wyszukiwanie wartości minimalnej przebiegu, prezentację danych w postaci tablic i wykresów.

9. Zamieszczanie funkcji matematycznych. Z głównej palety Programming wybrać subpaletę Numeric, a z niej:
- funkcję mnożenia Multiply (dwa obiekty),
- funkcję dzielenia Divide (dwa obiekty),
- stałą DBL Numeric Constant (dwa obiekty) i zadeklarować wartość 360,
- stałą Numeric Constant (dwa obiekty) i zadeklarować wartości 5 oraz 11,
- funkcję odejmowania Subtract,
- stałą 2π – subpalety Math & Scientifics Constants.

10. Z głównej palety Programming wybrać subpaletę Comparison, a z niej funkcję porównania Equal?
11. Zamieścić węzeł funkcji sinus z subpalety Mathematics i kolejnych palet Elementary & Special Functions / Trygonometric Functions.
12. Zamieścić pętlę While (Functions /Programming /Structures /Wile Loop).
13. Zamieszczanie funkcji tablicowych. Z głównej palety Programming wybrać subpaletę Array, a z niej:
- funkcję łączenia tablic Build Array,
- tworzenia podzbioru tablicy Array Subset,
- wyszukiwania ekstremów Array Max & Min.

14. Wykonać połączenia pomiędzy ikonami zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).



15. Wskaźniki (zmienne wyjściowe) funkcji tablicowej Array Max & Min o nazwach Wartość maksymalna, Indeks wartości max, Wartość minimalna, Indeks wartości min utworzyć za pomocą menu kontekstowych wyjść ikony funkcji Array Max & Min.
16. Zapisać plik programu pod aktualną nazwą (035_Sinus_skalowanie.vi).

17. **Uruchamianie programu**. Zadeklarować Liczbę kroków iteracji (interwał podziału kąta 360 stopni) oraz Współczynnik skalowania i uruchomić program. Obserwować zmiany:
- wartości prezentowanych w tablicach,
- przebiegów na wykresach,
- poprawności wyszukiwania wartości minimalnej i maksymalnej.

Obserwacje można prowadzić dla przykładowych wartości:
- liczba kroków iteracji = 360, Współczynnik skalowania = 2,
- liczba kroków iteracji = 30, Współczynnik skalowania = 2,5,
- liczba kroków iteracji = 6, Współczynnik skalowania = 5.

18. Zamknąć plik programu po zakończeniu testowania.

3.3.6. Macierze. Układ równań

Cel zadania: praktyczne zastosowanie zmiennych macierzowych do rozwiązywania układów równań.

Zakres zadania: Opracowanie aplikacji do rozwiązywania układu równań linowych metodą macierzową. W przypadku zadeklarowania układu równań w poniższej postaci

$$\begin{aligned} a_1x + b_1y + c_1z + ... &= W_1 \\ a_2x + b_2y + c_2z + ... &= W_2 \\ a_3x + b_3y + c_3z + ... &= W_3 \\ &... \end{aligned} \quad ...$$

możliwe jest przedstawienie go w postaci iloczynu macierzy (głównej) współczynników [A] i wektora niewiadomych [X] oraz wektora wyrazów wolnych [W].

$$[A] \cdot [X] = [W].$$

Jeżeli macierz (główna) współczynników nie jest macierzą jednostkową, tzn.:
- macierz odwrotna [A] istnieje,
- wyznacznik macierzy jest różny od zera,
- wiersze i kolumny macierzy są liniowo niezależne,

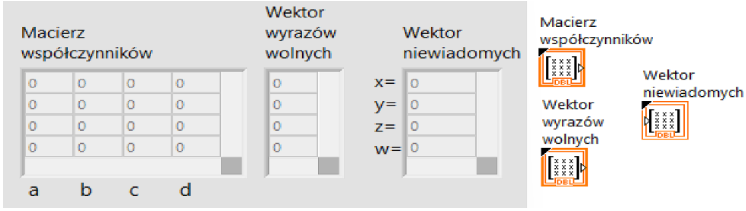
to możliwe jest uzyskanie rozwiązania (obliczenie wartości [X]) jako iloczynu odwrotnej macierzy współczynników [A] i macierzy wyrazów wolnych [W]:

$$[X] = A^{-1} \cdot [W].$$

Celem ćwiczenia będzie opracowanie aplikacji pozwalającej na zadeklarowanie macierzy współczynników [A], macierzy (wektora) wyrazów wolnych [W], w celu obliczenia macierzy (wektora) niewiadomych [X].

1. Uruchomić środowisko LabVIEW z pustym plikiem programu (Blank.vi). Celem kolejnych działań, które zaprezentowano na następnych rysunkach, będzie utworzenie panelu czołowego wraz z kodem.
2. Na panelu czołowym:

- zamieścić z palety panelu czołowego Array, Matrix & Cluster kontrolkę macierzy RealMatrix.ctl. Obiekt tablicy poszerzyć w ten sposób, aby prezentował tablicę 4x4 elementy. Usunąć wskaźnik indeksu (menu kontekstowe tablicy Visible Items / Index Display). Zmienić nazwę tablicy na „Macierz współczynników",
- zamieścić z palety panelu czołowego Array, Matrix & Cluster kontrolkę macierzy RealMatrix.ctl. Obiekt tablicy poszerzyć w ten sposób, aby prezentował tablicę 1x4 elementy. Usunąć wskaźnik indeksu (menu kontekstowe tablicy Visible Items / Index Display). Zmienić nazwę tablicy na „Wektor wyrazów wolnych",
- zamieścić z palety panelu czołowego Array, Matrix & Cluster kontrolkę macie-rzy RealMatrix.ctl. Obiekt tablicy poszerzyć w ten sposób, aby prezentował tabli-cę 1x4 elementy. Zmienić typ elementu z wejściowego na wyjściowy za pomocą menu kontekstowego i polecenia Change to Indicator. Usunąć wskaźnik indeksu (menu kontekstowe tablicy Visible Items /Index Display). Zmienić nazwę tablicy na „Wektor niewiadomych",
- wprowadzić dodatkowe opisy tablic *(a, b, c, d, x, y, z, w)*.



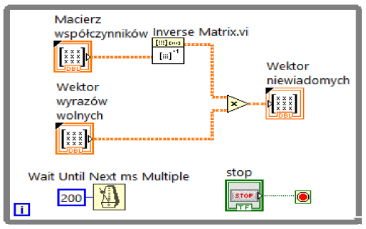
3. Zapisać plik programu pod nazwą (036_Uklad_rownan.vi) w lokalizacji wskazanej przez prowadzącego.

4. **Schemat blokowy programu**. Przejść do okna schematu blokowego, gdzie powinny znajdować się ikony obiektów zamieszczonych uprzednio na panelu czołowym.
5. Zamieszczanie elementów kodu programu:

- podprogram Inverse Matrix.vi – palety Functions /Mathematics /Linear Algebra,
- funkcję mnożenia Multiply – palety Functions /Programming /Numeric,
- węzeł opóźnienia Wait Until Next msMultple – palety Functions /Programming / Timing,
- stałą Numeric Constant i zadeklarować wartość 200,
- funkcję odejmowania Subtract,

6. Zamieścić pętlę While (np. Functions /Execution Control /Wile Loop with Button).

7. Wykonać połączenia pomiędzy ikonami zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).



8. Zapisać plik programu pod aktualną nazwą (036_Uklad_rownan.vi).

9. **Uruchamianie programu**. Zadeklarować wartości macierzy współczynników i wektora wyrazów wolnych dla poniższego układu równań. Uruchomić program. Sprawdzić, czy obliczone wartości zmiennych wynoszą: $x = -2$, $y = 2$.

$$2x + 3y = 2$$
$$4x + 5y = 2$$

10. Zadeklarować wartości macierzy współczynników i wektora wyrazów wolnych dla poniższego układu równań. Oczekiwane wartości niewiadomych: $x = 22{,}918$, $y = -36{,}105$, $z = 135{,}237$.

$$8x + y - z = 12$$
$$5y - 2z = -451$$
$$x + y + 5z = 663$$

11. Zadeklarować wartości macierzy współczynników i wektora wyrazów wolnych dla poniższego układu równań. Oczekiwane wartości niewiadomych: $x = -0{,}540$, $y = -8{,}143$, $z = 1{,}145$, $w = -37{,}815$.

$$16x - 12y - 3z + 2w = 10$$
$$12y - 2z = -100$$
$$2x + 3y + 5z - 12w = 434$$
$$-2x + 14z - 6w = 244$$

12. Zamknąć plik programu po zakończeniu testowania.

3.3.7. Stała klastrowa

Cel zadania: utrwalenie wiadomości o korzystaniu ze zmiennych klastrowych. Wprowadzenie do stosowania stałej klastrowej.

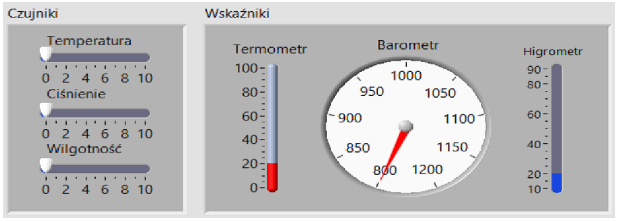
Zakres zadania: budowa programu symulującego odczyt, składającego się z trzech czujników o ustalonym zakresie sygnału 0–10, pozwalającego na przeskalowanie sygnału do wartości rzeczywistych i jego prezentację. Do przekazywania i prezentacji wartości zostaną wykorzystane klastry, zaś do przeskalowania sygnału obiekty stałej klastrowej.

1. **Budowa panelu czołowego.** Z niżej wymienionych palet należy wybrać i sformatować poniższe elementy panelu:

- powłokę klastra (palety Controls /Modern /Array, Matrix & Cluster I Cluster). Zmienić nazwę klastra na Czujniki. Do tego klastra zostaną wprowadzone trzy kolejne obiekty,
- suwak poziomy Slide (palety Controls /Modern /Numeric i Horizontal Pointer Slide). Zmienić nazwę na Temperatura,
- suwak poziomy Slide. Zmienić nazwę na Ciśnienie,
- suwak poziomy Slide. Zmienić nazwę na Wilgotność.

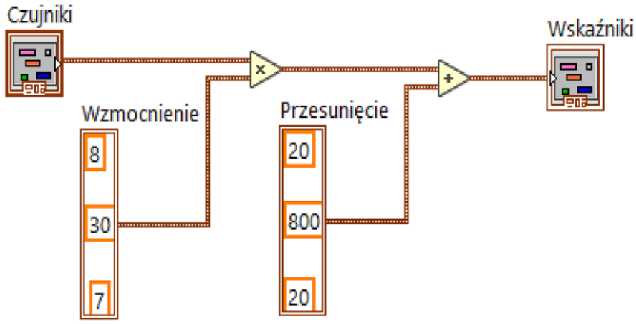
2. Zamieścić drugą powłokę klastra (palety Controls /Modern /Array, Matrix & Cluster i Cluster). Zmienić nazwę klastra na Wskaźniki. Do tego klastra zostaną wprowadzone trzy kolejne obiekty,

- wskaźnik Termometr (palety Controls /Modern /Numeric i Thermometer). Zmienić nazwę na Temperatura. Zadeklarować zakres 0–100,
- wskaźnik Gauge (palety Controls /Modern /Numeric i Gauge). Zmienić nazwę na Barometr. Zadeklarować zakres 800–1200,
- wskaźnik Wypełnienia (palety Controls /Modern /Numeric i Vertical Fill Slide). Zmienić nazwę na Higrometr. Zadeklarować zakres 10–90.



3. Skontrolować zgodność kolejności adekwatnych czujników i wskaźników w klastrach Czujniki i Wskaźniki. Zmiana kolejności możliwa jest za pomocą polecenia kontekstowego klastra Reorder Controls in Cluster.
4. Zapisać plik programu pod nazwą (037_Klaster_stala.vi) w lokalizacji wskazanej przez prowadzącego.

5. **Schemat blokowy programu**. Przejść do okna schematu blokowego.



6. Zamieścić obiekt stałej klastrowej Cluster Constant (Functions /Programming / Cluster, Class & Variant). Zmienić nazwę stałej klastrowej na Wzmocnienie.
7. Zamieścić drugą stałą klastrową i zmienić jej nazwę na Przesunięcie.
8. Wprowadzić do zmiennych klastrowych stałe liczbowe (paleta Programming / Numeric i stała DBL Numeric Constant) i zadeklarować ich wartości zgodnie ze schematem blokowym.
9. Kolejność stałych w stałej klastrowej na schemacie blokowym jest zgodna z kolejnością występowania wartości w klastrze czujników i wskaźników.

Tab. 3.1. Dane stałych klastrowych

Czujniki	Wzmocnienie	Przesunięcie	Wskaźniki
Temperatura	8	20	Termometr
Ciśnienie	30	800	Barometr
Wilgotność	7	20	Higrometr

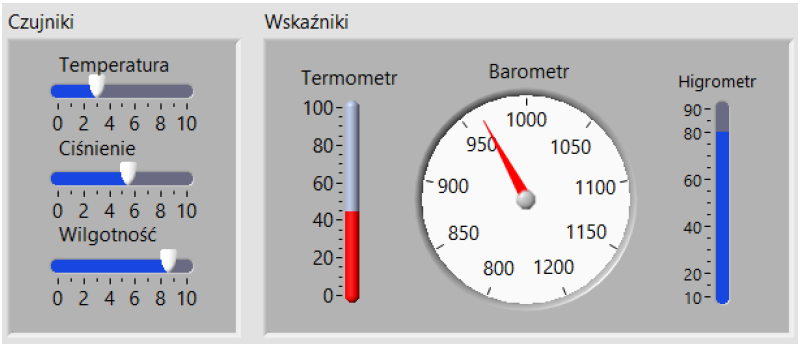
Zmiana kolejności elementów stałej klastrowej możliwa jest za pomocą użycia polecenia kontekstowego obiektu stałej klastrowej Reorder Controls in Cluster.

10. Zamieszczanie funkcji matematycznych. Z głównej palety Programming wybrać subpaletę Numeric, a z niej:

- funkcję mnożenia Multiply,
- unkcję dodawania Add,

11. Wykonać połączenia pomiędzy ikonami zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
12. Zapisać plik programu pod aktualną nazwą (037_Klaster_stala.vi)

13. **Uruchamianie programu**. Uruchomić program w trybie ciągłym. Wynikiem działania programu powinna być reakcja zmiennych wyjściowych, zamieszczonych w klastrze Wskaźniki, na zmiany elementów wejściowych z klastra Czujniki. Zmiany powinny być rozciągnięte prawie na cały zakres wskaźników wyjściowych.



14. Zamknąć plik programu po zakończeniu testowania.

3.4. Pytania kontrolne

1. Która z dwóch popularnych pętli środowiska LabVIEW nazywana jest warunkową i dlaczego?
2. Która z wartości w odniesieniu do liczby pętli For będzie nadrzędna, jeśli dojdzie do niezgodności rozmiaru tablicy przyłączonej do pętli z wartością zadeklarowaną w terminalu iteracyjnym?
3. Na czym polega autoindeksowanie w przypadku pętli?
4. Po czym rozpoznaje się charakter tunelu pętli?
5. Co jest powodem prezentowania rejestru przesuwnego w kolorze czarnym?
6. Jaką wartość mają komórki rejestru przed pierwszym uruchomieniem programu?
7. Co może być przyczyną problemów z dodaniem rejestru przesuwnego do pętli za pomocą wskaźnika myszy?
8. Co oznacza wyświetlanie wykresu w trybie Scope?
9. Jak można regulować długość historii danych prezentowanych na wykresie Waveform graph?
10. Ile typów danych i jakich danych może równocześnie magazynować tablica?

4. Sterowanie kodem programu. Wykorzystanie funkcji obliczeniowych. Zastosowanie zmiennych łańcuchowych do komunikacji i zapisu wartości

4.1. Cel zajęć

W tym rozdziale zostaną poruszone zagadnienia sterowania aplikacją, pozwalające na wskazywanie kodu w oparciu o bieżący stan zmiennych oraz wymuszanie kolejności wykonywania poleceń. Zostaną przedstawione węzły, umożliwiające swobodne prowadzenie działań matematycznych bez korzystania z gotowych węzłów środowiska programistycznego. Zaprezentowane zmienne łańcuchowe będą pozwalały na naturalną komunikację z użytkownikiem oraz magazynowanie danych w postaci plików.

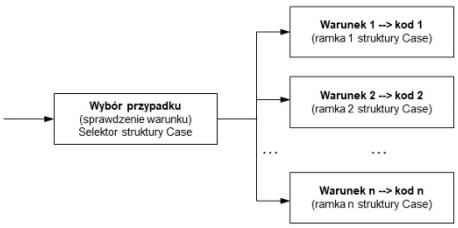
Celem zajęć jest praktyczne zapoznanie się z:

- rozgałęzianiem kodu za pomocą struktury wyboru (Case Structure),
- deklarowaniem kolejności zdarzeń nie mających naturalnego następstwa za pomocą struktury sekwencyjnej (Sequence Structure),
- prowadzeniem operacji matematycznych bez wykorzystania standardowych węzłów matematycznych za pomocą węzłów Expression Node oraz Formula Node,
- stosowaniem zmiennych typu łańcuchowego (String) do komunikacji z użytkownikiem i przechowywaniem danych dyskowych,
- prostymi i złożonymi węzłami formatowania, zapisywania i odczytywania plików.

4.2. Wstęp

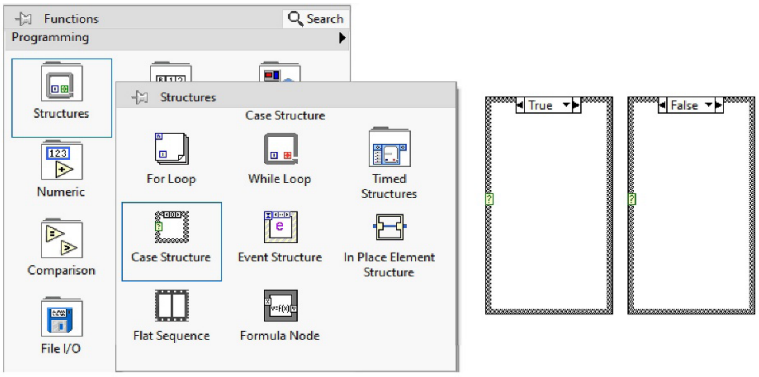
4.2.1. Struktura wyboru (Case Structure)

W LabVIEW występuje struktura podobna do polecenia „if … then … else", znanego z innych środowisk programistycznych. Struktura wyboru Case Structure zawiera dwa lub więcej diagramy (przypadki, możliwości) (rys. 4.1).



Rys. 4.1. Struktura wyboru – idea funkcjonowania

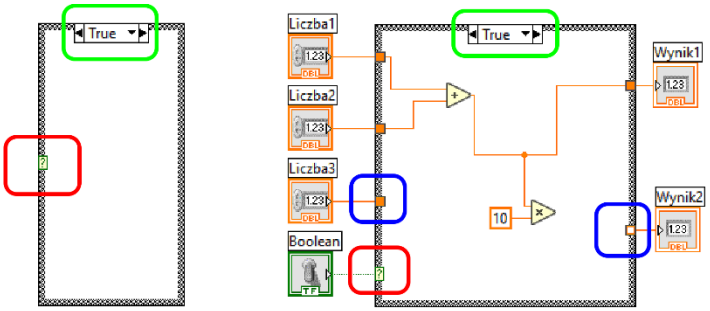
Domyślnie w kodzie zamieszczana jest struktura logiczna (rys. 4.2). Na schemacie blokowym widoczny jest tylko jeden diagram i tylko jeden z diagramów jest wykonywany – w zależności od spełnionego warunku.



Rys. 4.2. Struktura wyboru – zamieszczanie struktury w kodzie

Graficzna reprezentacja struktury wyboru zawiera poniższe elementy (rys. 4.3):

- identyfikator (Selector label) – wartość wybranego przypadku, warunku (np. False i True),
- selektor (Case selector) – terminal wejściowy dostarczający informacji o spełnionym warunku. Barwa selektora sygnalizuje typ wielkości wejściowej (logiczna, całkowita, łańcuchowa), a tym samym pośrednio wskazuje na liczbę przypadków (np. zmienna logiczna = 2 przypadki).

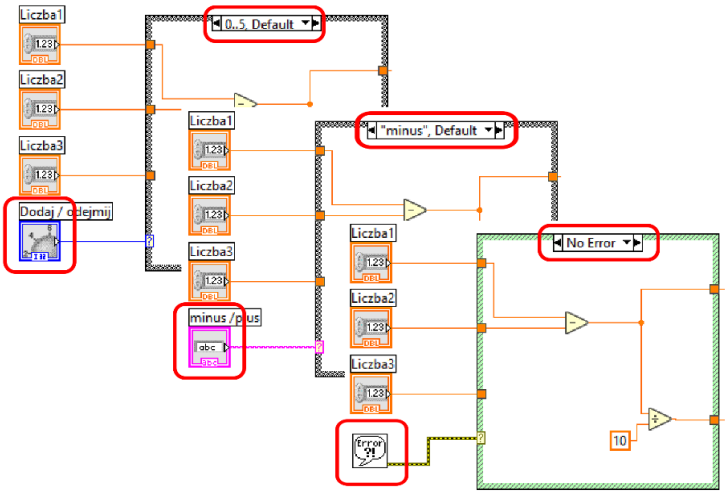


Rys. 4.3. Struktura wyboru – elementy struktury

Dostępna liczba „przypadków, wyborów" równa jest 2^{31}-1 (poza strukturą logiczną) [12]. Wartości tuneli wejściowych dostępne są w każdym przypadku i nie muszą być zawsze używane. Tunele wyjściowe muszą być podłączone (wykorzystane), ponieważ w innym przypadku program generuje błąd. Niepodłączone (w dowolnym z przypadków) tunele wyjściowe są oznaczane pustym kwadratem, co pozwala na ich łatwą identyfikację.

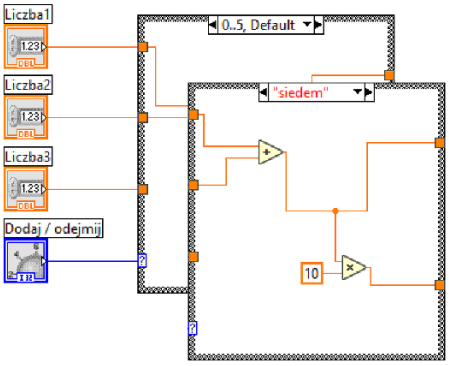
Przy braku zgodności identyfikatora z selektorem wykonywana jest struktura domyślna (Default). Brak zadeklarowania kodu struktury domyślnej (Default) traktowany jest przez środowisko jako błąd.

W zależności od typu zmiennej przypisanej do selektora można wyróżnić kilka typów struktur wyboru (rys. 4.4).



Rys. 4.4. Wybrane typy struktur wyboru

Definiowanie przypadków, warunków wykonywania kodu polega na modyfikacji (edycja w miejscu) identyfikatora. W przypadku gdy typ wartości identyfikatora jest niezgodny z typem selektora, występuje sygnalizacja niezgodności za pomocą czerwonej barwy zapisu w identyfikatorze (rys. 4.5).



Rys. 4.5. Struktura wyboru – niezgodność typu danych selektora i identyfikatora

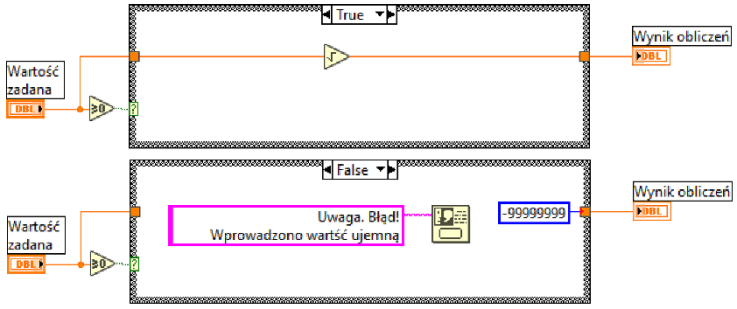
Dla przypadków sterowanych typem Integer istnieje możliwość deklarowania:

- indywidualnego (1,5,7,12),
- zakresami ..10, 100.., 10..20 (włącznie),
- mieszania indywidualnego i zakresów 1..5, 7, 10..,

Dołączenie floating point do integer do selektora powoduje zaokrąglanie warunku do najbliższej całkowitej parzystej. Dla przypadków sterowanych typem łańcuchowym (string) można zadeklarować rozpoznawanie wielkich znaków (menu kontekstowe Case Insensitive Match).

Poniżej zamieszczono przykładowy program do obliczania pierwiastka kwadratowego, wykorzystujący strukturę wyboru (rys. 4.6). W programie tym:

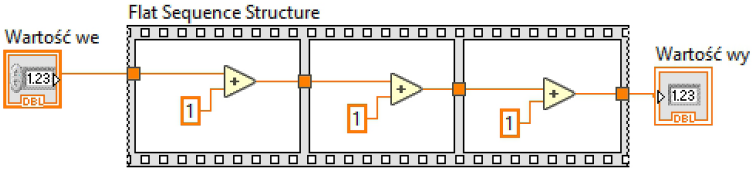
- dla liczb większych lub równych zeru obliczany jest pierwiastek kwadratowy (case True),
- dla liczb mniejszych od zera wyświetlany jest komunikat błędu (case False).



Rys. 4.6. Przykładowy program wykorzystujący strukturę wyboru

4.2.2. Struktura sekwencyjna (Sequence Structure)

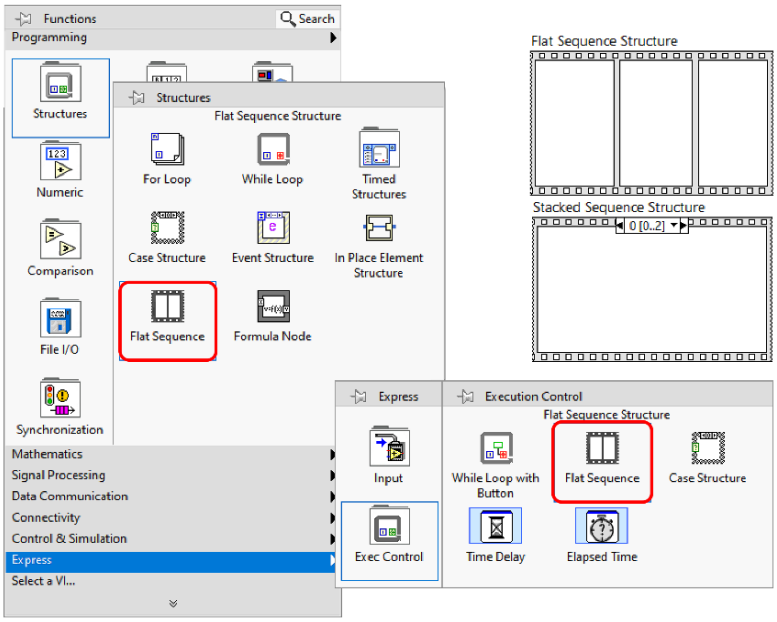
Do deklarowania kolejności zdarzeń, niemających naturalnego następstwa, należy stosować strukturę sekwencyjną (Sequence Structure). Zalecana jest ona szczególnie w sytuacjach, w których pozyskuje się wartości zmiennych bez stosowania „przewodów", np. poprzez zmienne lokalne, globalne, węzły właściwości. Struktura sekwencyjna zawiera jeden lub więcej diagramów kodu wykonywanych kolejno po sobie (rys. 4.7).



Rys. 4.7. Struktura wyboru – idea działania

Strukturę sekwencyjną charakteryzują dwie cechy (ograniczenia) [12]:

- wykorzystanie tuneli wyjściowych możliwe jest we wszystkich „ramkach" (jak w Case), ale wynik dostępny jest po wykonaniu całej struktury,
- struktura nie zwraca żadnych danych do momentu wykonania pełnej sekwencji ramek w strukturze.



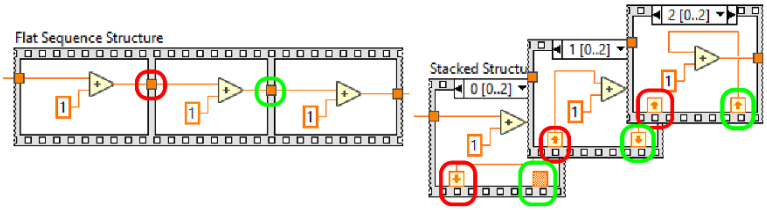
Rys. 4.8. Struktura wyboru – lokalizacja struktury na paletach

Struktura występuje w dwóch postaciach (rys. 4.8 i 4.9):

- domyślnej, płaskiej (Flat Sequence) – bardziej przejrzystej,
- stosowej (Stacked) – ograniczającej miejsce prezentacji programu, trudniej jest w niej analizować kod na schemacie.

Większość operacji można przeprowadzić przez menu kontekstowe tj.:

- dodawanie ramek,
- usuwanie ramek,
- kopiowanie,
- zmianę kolejności ramek struktury.



Rys. 4.9. Struktura w postaci stosowej i płaskiej

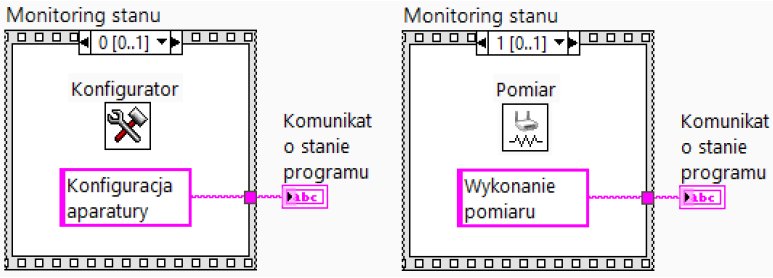
Obiekt „Lokalne zmienne sekwencyjne" służy do przekazywania danych pomiędzy ramkami (kierunek strzałki = kierunek danych, zacienione = niewykorzystane w ramce) (rys. 4.9).

Istnieje możliwość konwersji struktury płaskiej do stosowej i powrotna konwersja do płaskiej. Taka konwersja zmienia układ kodu bez zmiany funkcjonalności.

W związku z tym, że struktura sekwencyjna nie zwraca żadnych danych do momentu wykonania pełnej sekwencji ramek w strukturze, istnieje możliwość jej błędnego wykorzystania. Jej negatywne cechy można opisać w poniższy sposób:

- struktura sekwencyjna przeszkadza w korzystaniu z niezależnej równoległości wykonywania operacji (np. GPIB, DAQ),
- blokuje dostęp („fałszuje" dane) – udostępnia wartości po pełnym wykonaniu struktury.

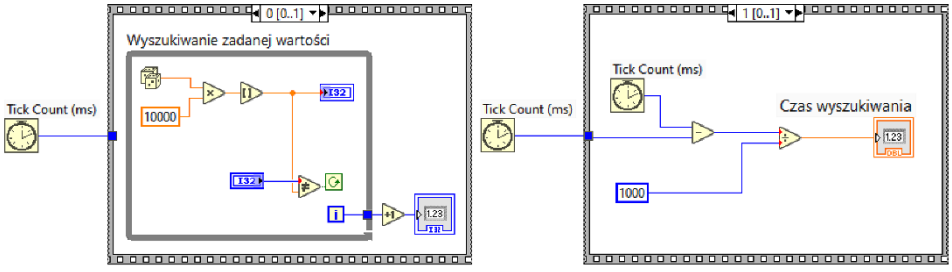
W programie z rysunku 4.10 komunikat o stanie wykonywania kodu dostępny jest dopiero po wyjściu ze struktury, zamiast po każdym etapie wykonywania kodu w poszczególnych ramkach [12, 13].



Rys. 4.10. Błędne użycie struktury wyboru

Na rysunku 4.11 zamieszczono przykład wykorzystania Sequence Structure. Struktura sekwencyjna użyta jest tutaj do obliczania czasu wykonania zadania (tutaj wyszukiwania liczby równej zadanej). Zadania realizowane w poszczególnych fazach:

- faza 1 (ramka 0) pobranie czasu i wykonanie zadania/programu,
- faza 2 (ramka 1) pobranie czasu i obliczenie różnicy w porównaniu z wartością czasu zachowaną w tunelu struktury.

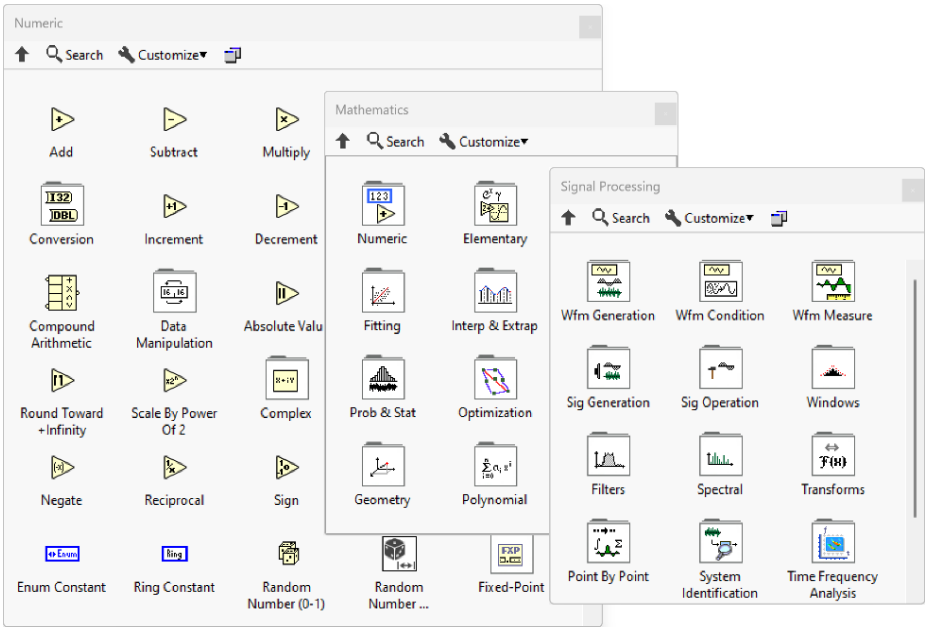


Rys. 4.11. Sequence Structure – przykładowy program

4.2.3. Funkcje obliczeniowe – węzeł Expression Node

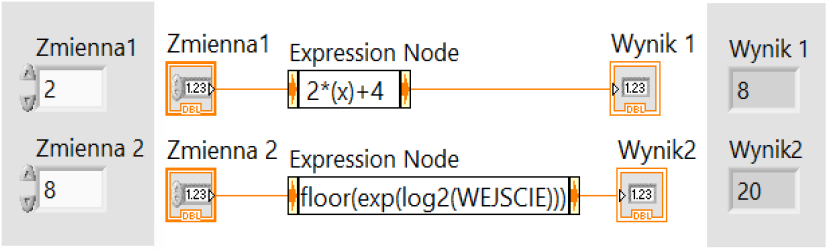
Najprostsze funkcje matematyczne środowiska, takie jak dodawanie, odejmowanie, potęgowanie czy zaokrąglanie, dostępne są z poziomu palety Numeric. Większość typowych operacji matematycznych można przeprowadzić za pomocą grup funkcji z palety Mathematics. Bardziej skomplikowane działania możliwe są do wykonania za pomocą dedykowanych dodatków, takich jak np. zgromadzone na palecie Signal Processing (rys. 4.12). Obiektami o pośredniej skali złożoności, tj. obiektami, które pozwalają na prowadzenie w jednym punkcie obliczeń nieco bardziej złożonych niż proste funkcje Numeric, są:

- węzły wyrażenia – Expression Node,
- węzły podprogramów typu express Formula,
- węzły formuły (wzoru) – Formula Node.



Rys. 4.12. Lokalizacja funkcji matematycznych o różnym stopniu złożoności

Węzeł Expression Node pozwala na prowadzenie obliczeń w wyrażeniach posiadających tylko jedną zmienną (rys. 4.13). Terminal wejściowy węzła traktuje przypisaną wartość jako wartość zmiennej wejściowej, na której prowadzone są operacje wewnątrz węzła. Każdy ciąg znaków, który nie jest rozpoznawany jako funkcja, traktowany jest jako zmienna (oznaczenie zmiennej) (tab. 4.1). Może to być znak np. „x", jak i ciąg znaków np. „wartość". Węzeł Expression Node można zamieścić z poziomu subpalety Numeric.



Rys. 4.13. Przykład użycia węzła Expression Node

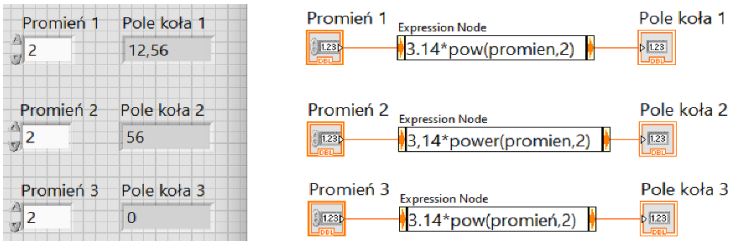
Tab. 4.1. Zestawienie wybranych funkcji obsługiwanych przez węzły Expression Node

abs(*x*)	Wartość bezwzględna *x*
cos(*x*)	Kosinus *x* (*x* w radianach)
cosh(*x*)	Kosinus hiperboliczny *x*
acos(*x*)	Oblicza odwrotność cosinusa (arcus cos) *x* (wynik w rad)
acosh(*x*)	Oblicza odwrotność cosinusa hiperbolicznego *x*
Pozostałe funkcje trygonometryczne jak powyższy cos(*x*)	
ceil(*x*)	Zaokrąglenie w górę do najmniejszej liczby całkowitej
exp(*x*)	Oblicza e do potęgi *x*
expm1(*x*)	Oblicza e do potęgi *x* pomniejszonej o wartość 1
floor(*x*)	Zaokrąglenie w dół do największej liczby całkowitej
getexp(*x*)	Zwraca potęgę *x*
getman(*x*)	Zwraca mantysę wartości *x*
int(*x*)	Zaokrągla *x* do najbliższej liczby całkowitej
ln(*x*)	Oblicza logarytm naturalny z *x* (o podstawie e)
log(*x*)	Oblicza logarytm z *x* (o podstawie 10)
log2(*x*)	Oblicza logarytm z *x* (o podstawie 2)
max(*x*,*y*)	Porównuje *x* i *y* i zwraca większą wartość
min(*x*,*y*)	Porównuje *x* i *y* i zwraca mniejszą wartość
mod(*x*,*y*)	Oblicza resztę z *x*/*y*
pow(*x*,*y*)	Oblicza *x* podniesione do potęgi *y*
rand( )	Generuje liczbę zmiennoprzecinkową z zakresu od 0 do 1
sign(*x*)	Zwraca 1, dla $x > 0$, 0 dla $x = 0$, -1, dla $x < 0$
sqrt(*x*)	Pierwiastek kwadratowy z *x*

Węzły Expression Node charakteryzują poniższe cechy:

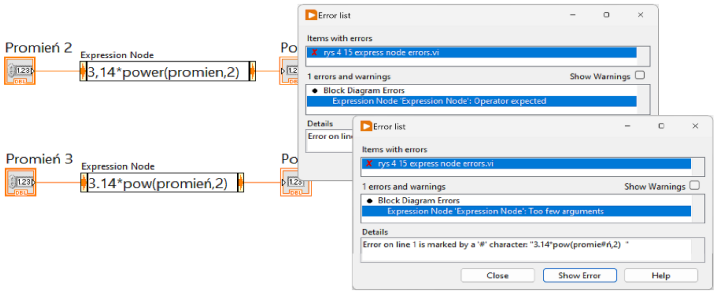
- nie wykonują obliczeń na liczbach zespolonych,
- jako separator dziesiętny akceptowany jest jedynie znak kropki „ . ”. Ignorowane są ustawienia systemu operacyjnego,
- w zmiennych węzłów rozróżniana jest wielkość znaków,
- możliwe jest stosowanie jedynie stałej π jako „pi”. W przypadku tej stałej również rozróżniana jest wielkość znaków,
- znaki diakrytyczne nie są poprawnie rozpoznawane.

Na rysunku 4.14 zaprezentowano węzeł wykorzystany do obliczania pola koła, w oparciu o wprowadzaną wartość promienia. Zastosowano w nim funkcję podnoszenia do potęgi – pow(x,y) = x^y. Pierwszy z przypadków dotyczy sytuacji poprawnego obliczenia pola. W drugim przypadku wprowadzono błędny separator dziesiętny, co skutkowało błędnym wynikiem. W trzecim przypadku zadeklarowano zmienną z wykorzystaniem znaku „ń”, który został zinterpretowany jako dodatkowa zmienna i środowisko LV nie pozwoliło na uruchomienie kodu.



Rys. 4.14. Węzeł Expression Node użyty poprawnie i błędnie

W przypadku popełnienia błędu polegającego np. na wprowadzeniu błędnej nazwy funkcji, błędnego oznaczenia zmiennej (np. zastosowanie znaków diakrytycznych), środowisko wygeneruje przy próbie uruchomienia listę błędów z oznaczeniem miejsca, które nie pozwala na wykonanie kodu (rys. 4.15).



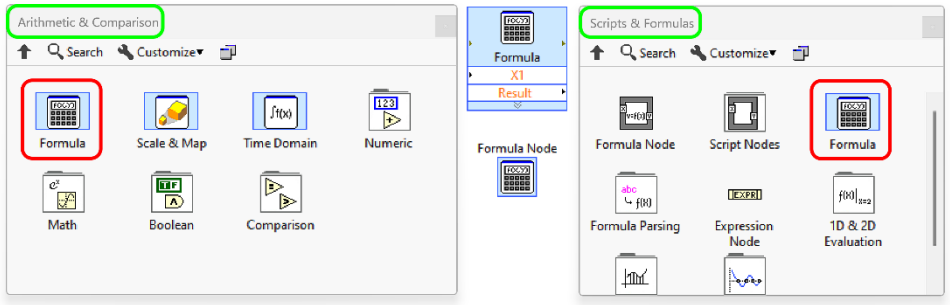
Rys. 4.15. Identyfikacja błędów za pomocą Error List

4.2.4. Funkcje obliczeniowe – Formula Express VI

Zaletą uprzednio opisanych węzłów Expression Node jest łatwość obsługi, polegająca na prostym wpisaniu wyrażenia matematycznego, symbolizującego funkcję lub zestaw funkcji, która ma zostać wykonana na pojedynczej zmiennej wejściowej. Do najpoważniejszych wad węzłów Expression Node należy zaliczyć brak identyfikacji prostych błędów literowych, które mogą spowodować, że błędna nazwa funkcji może zostać zakwalifikowana jako nazwa zmiennej. Do poważnych ograniczeń należy zakwalifikować możliwość operowania na pojedynczej zmiennej. Powyższe wady i ograniczenia usuwa podprogram Express vi o nazwie Formula (rys. 4.16).

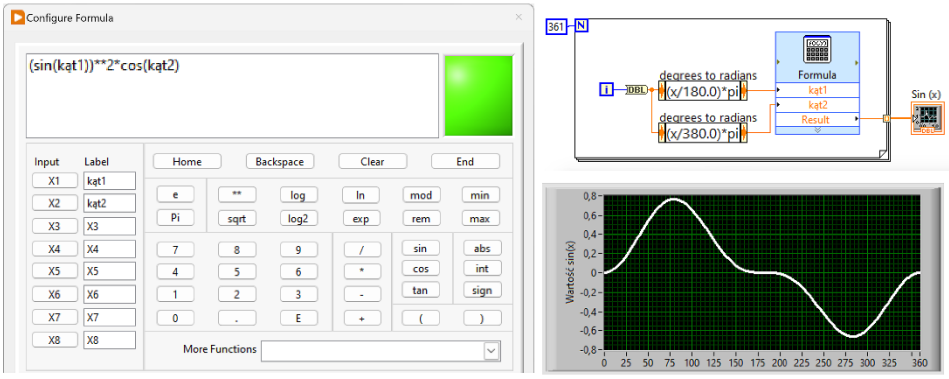
Węzeł (podprogram) Formula Express VI dostępny jest z poziomu subpalet:

- Functions /Express /Arithmetic & Comparison,
- Functions /Mathematics /Scripts & Formulas.



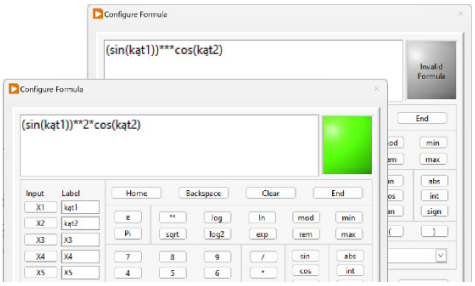
Rys. 4.16. Lokalizacja węzła Formula Node na subpaletach schematu blokowego

Express VI Formula posiada interfejs typowego kalkulatora naukowego (rys. 4.17). Minimalizuje ograniczenia węzła Formula Node przez wprowadzenie obsługi więcej niż jednej zmiennej wejściowej (możliwość stosowania 8 zmiennych wejściowych) oraz wprowadzanie funkcji za pomocą klawiszy i listy rozwijalnej.



Rys. 4.17. Interfejs i przykładowe wykorzystanie Formula Express VI w obliczeniach

Formula Express VI minimalizuje możliwość popełnienia błędów przy wprowadzaniu nazw funkcji, przez korzystanie z klawiszy funkcji oraz listy rozwijalnej. Oczywiście możliwe jest „ręczne" wprowadzenie zarówno nazw funkcji, jak i zmiennych. Błędne deklarowanie zmiennych, nazw funkcji, składni wyrażenia jest automatycznie identyfikowane i sygnalizowane na panelu „kalkulatora" (rys. 4.18).

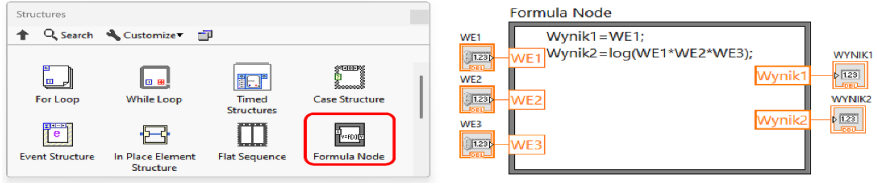


Rys. 4.18. Sygnalizacja błędu składni wyrażenia (prawy górny róg)

4.2.5. Funkcje obliczeniowe – węzeł Formula Node

Węzeł Formula Node ma charakter węzła, który można wypełniać formułami matematycznymi, podobnie jak Expression Node i Formula Express VI. Przy czym możliwe jest prowadzenie operacji matematycznych, jak i programowanie, w sposób podobny do stosowanego w języku C.

Węzeł Formula Node znosi ograniczenie w zakresie liczby zmiennych wejściowych i wyjściowych (rys. 4.19). Najważniejsza wydaje się możliwość korzystania z wielu zmiennych wyjściowych. Jedynym ograniczeniem jest ograniczenie geometryczne, czyli dostępna ilość powierzchni na ramkach węzła.

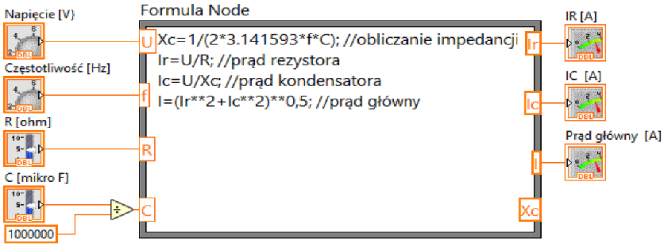


Rys. 4.19. Lokalizacja węzła Formula Node na subpalecie Structures

W przypadku węzłów Formula Node konieczne jest stosowanie się do poniższych wymagań (rys. 4.20):

- zmienne występujące w Formula Node muszą zostać uprzednio zadeklarowane na ramkach struktury,
- każda zmienna wejściowa musi mieć przypisaną wartość w programie,
- w nazwach zmiennych rozpoznawana jest wielkość znaków,
- wyrażenia muszą kończyć się znakiem średnika (;),

- zmienne wykorzystywane wewnątrz struktury, których wartości nie muszą być wykorzystywane poza strukturą Formula Node, muszą zostać zadeklarowane jako zmienne wyjściowe,
- możliwe jest stosowanie komentarzy oznaczonych /*komentarz*/ lub //komentarz,
- dopuszczalnym separatorem dziesiętnym jest znak kropki.



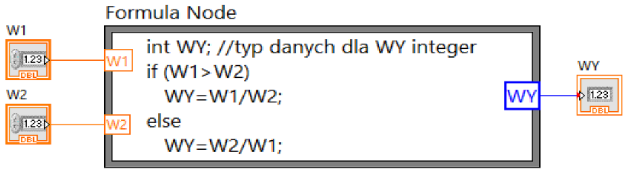
Rys. 4.20. Przykład węzeł Formula Node zastosowany do obliczeń

Węzły Formula Node można wzbogacać o funkcje programowania. W ich zakres wchodzą funkcje sterujące, warunkowe, iteracyjne oraz wyboru (przełączania). Dopuszczalne funkcje zamieszczono w tabeli 4.2.

Tab. 4.2. Funkcje obsługiwane przez węzły Formula Node

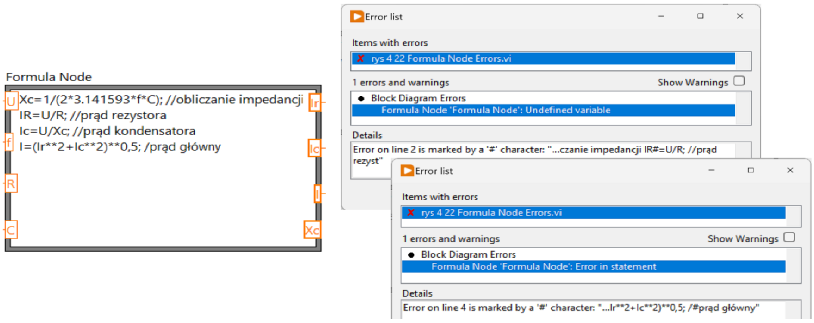
Typ wyrażenia	Wyrażenie	Składnia
Sterujące	Break	break (np. wyjście z pętli)
	Continue	continue (np. przejście do następnej iteracji pętli)
Warunkowe	If	if (warunek) instrukcja
	If-Else	if instrukcja else instrukcja
Iteracyjne	Pętla Do	do instrukcja while (warunek)
	Pętla For	for ([warunek], [warunek], [warunek]) instrukcja
	Pętla While	while (warunek) instrukcja
Przełączania	Switch	switch (przypisanie) {lista przypadków (case-list)}
	Case List	case-list: liczba instrukcja liczba instrukcja
	Case	case liczba: lista-instrukcji-case default: instrukcja domyślna

Na rysunku 4.21 zamieszczono przykład zastosowania struktury warunkowej IfElse. W jej ramach porównywane są dwie wielkości zmiennoprzecinkowe (W1 oraz W2), a w zależności od wyniku porównania prezentowany jest iloraz zmiennych. Wartość wyjściowa ma postać całkowitą (integer) zadeklarowaną w pierwszym wierszu Formula Node.



Rys. 4.21. Przykład węzeł Formula Node zastosowany do obliczeń

Podobnie jak w węźle Expression Node, w przypadku popełnienia błędu deklaracji zmiennej, błędnej nazwy funkcji, błędnej składni poleceń, itp., środowisko wygeneruje przy próbie uruchomienia listę błędów z oznaczeniem punktów uniemożliwiających wykonanie kodu (rys. 4.22).

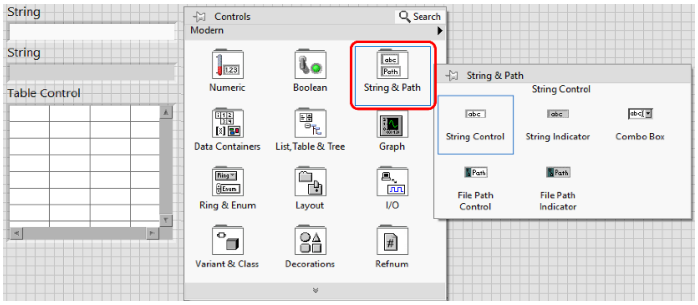


Rys. 4.22. Identyfikacja błędów za pomocą Error List

4.2.6. Zmienne łańcuchowe

Zmienna typu łańcuchowego String to niezależna od platformy sekwencja wyświetlanych lub niewyświetlanych znaków ASCII. Znajduje zastosowanie przy:

- prezentowaniu komunikatów,
- sterowaniu sprzętem,
- deklarowaniu ścieżek dostępu,
- przechowywaniu danych na dysku.



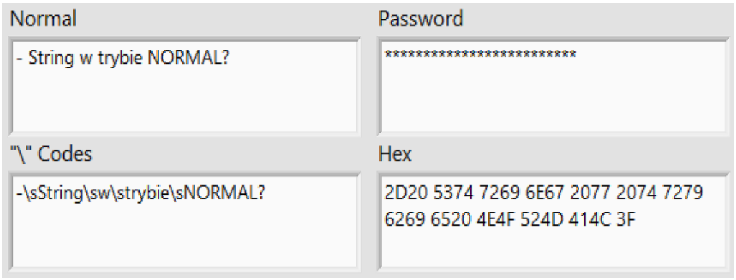
Rys. 4.23. Zmienne łańcuchowe – paleta panelu czołowego

Zamieszczanie kontrolek i wskaźników typu string zapewnia kilka lokalizacji (palet) o różnej zawartości (rys. 4.23). Nie wszystkie obiekty z palet oznaczonych String bazują na zmiennych String. Typ łańcuchowy obsługuje wartości zamieszczone w takich obiektach, jak:

- tabele,
- pola tekstowe,
- wskaźniki tekstowe.

Wartości zmiennych łańcuchowych mogą być wyświetlane w czterech opisanych niżej stylach, modyfikowanych za pomocą menu kontekstowego obiektu (rys. 4.24), są to:

- styl normalny (domyślny) – wyświetlane są w nim znaki drukowane, a znaki niedrukowane to kwadraty.
- kody/backslash – wyświetlanie znaków niedrukowanych poprzedzonych ukośnikiem.
- hasła – gwiazdki dla wszystkich znaków.
- styl szesnastkowy – kody szesnastkowe znaków.

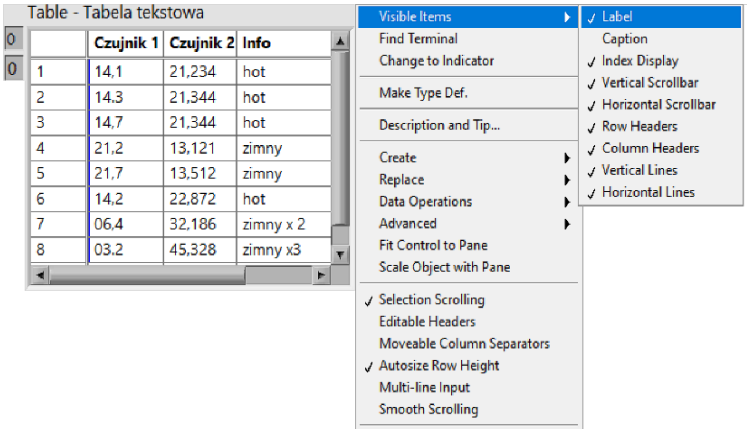


Rys. 4.24. Style prezentacji wartości zmiennych typu string

Tabela jest przykładem kontenera grupującego zmienne łańcuchowe. Każda komórka tabeli stanowi łańcuch, dlatego tabelę można traktować jak dwuwymiarową tablicę typu łańcuchowego (rys. 4.25).

Cechy charakterystyczne tabel to:

- cyfry w tablicach są przechowywane jako typ string. Wprowadza możliwość pomylenia tabeli z tablicą typu liczbowego.
- nagłówki kolumn i wierszy nie są wyświetlane automatycznie.
- nagłówki tabel to jednowymiarowe tablice.
- przed wyświetleniem danych tabeli konieczna jest konwersja tablicy liczbowej do tablicy string.
- operacje mające wpływ na wygląd dostępne są za pomocą menu kontekstowego.



Rys. 4.25. Zmienne łańcuchowe zamieszczone w tabeli

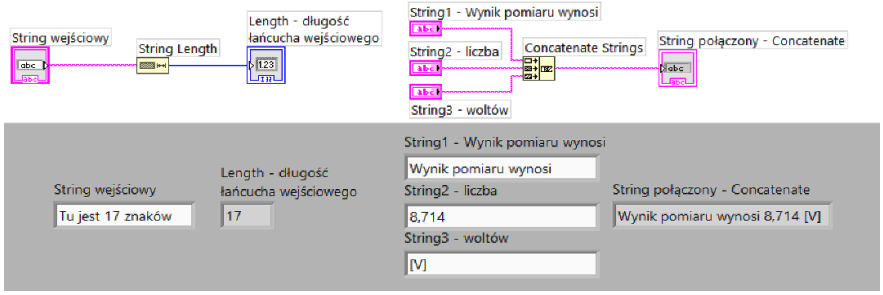
W celu ograniczenia możliwości popełnienia pomyłki przy zapisie lub odczycie danych z tabeli zalecane jest:

- zadeklarowanie nagłówków kolumn i wierszy,
- wyświetlenie wskaźnika indeksu.

4.2.7. Wybrane funkcje zmiennych łańcuchowych

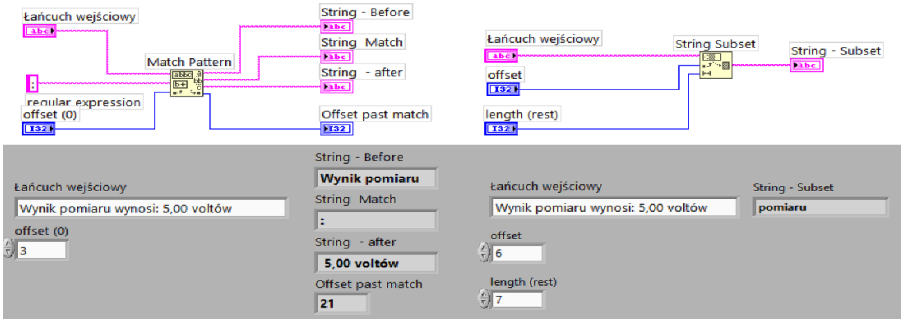
Poniżej zamieszczono opis często stosowanych funkcji operujących na zmiennych typu łańcuchowego. W większości przypadków nazwa identyfikuje działanie funkcji:

- String Length – zwraca liczbę znaków łańcucha. Wliczane są spacje (rys. 4.26),
- Concatenate Strings – łączy zmienne łańcuchowe w pojedynczy ciąg znaków (rys. 4.26),



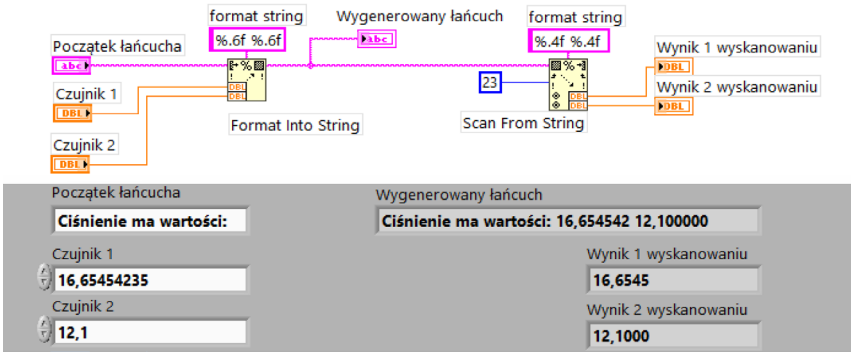
Rys. 4.26. Widok kodu i panelu wybranych funkcji łańcuchowych

- Match Pattern – wykrywa zadeklarowany łańcuch i zwraca ciąg wykryty, przed, po i długość (rys. 4.27),
- String Subset – zwraca część łańcucha począwszy od offset (od zera) do długości length (rys. 4.27).



Rys. 4.27. Widok kodu i panelu funkcji łańcuchowych Match Pattern i String Subset

- Format Into String – konwertuje argumenty dowolnego formatu do postaci łańcucha (rys. 4.28),
- Scan From String – wyszukuje dane numeryczne (0–9, plus, minus, separator, e/E (rys. 4.28).



Rys. 4.28. Widok ikon funkcji konwersyjnych Format Into String oraz Scan From String

4.2.8. Składnia wskaźnika formatu (Format Specifier)

Istnieją funkcje operujące na zmiennych łańcuchowych. Zmienne łańcuchowe mogą zawierać znaki cyfr lub liczby. Niektóre z tych funkcji obsługują formatowanie wyjściowych zmiennych liczbowych lub deklarowanie sposobu interpretacji liczbowych danych wejściowych. Informacji o formatowaniu dostarcza tzw. Format Specifier przekazywany np. do wejść format string z rysunku 4.28. I tak wybrane elementy wskaźnika formatu opisano poniżej.

Na przykład dla funkcji Scan From String:

- %[Szerokość]Kod_konwersji
- Szerokość > 0
- . (kropka) Precyzja i _Cyfr_Znaczących >= 0

Na przykład dla funkcji Format Into String:

- %[-][+][#][^][0][Szerokość][.Precyzja || _Cyfr_Znaczących] Kod_konwersji [{Jednostka}]
- % - początek elementu Format Specifier
- [-][+][#][^][0] – opcjonalne elementy formatowania (lewo, prawo, …, usuwanie zer)
- Jednostka – nadpisuje oryginalną jednostkę, o ile wprowadziło ją formatowanie.

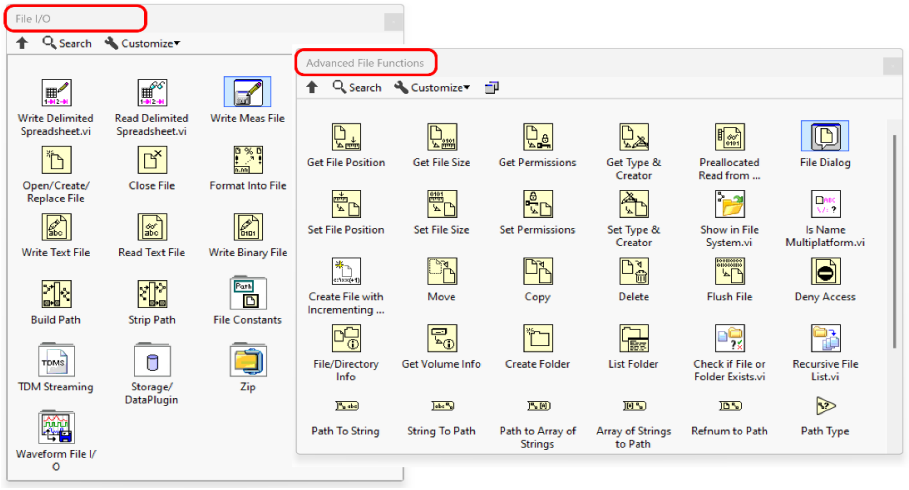
Kody:

- x – szesnastkowy,
- o – ósemkowy,
- b – dwójkowy,
- d – dziesiętny (oznaczony +/-),
- u – nieoznaczony dziesiętny,
- f – zmiennoprzecinkowy, ułamkowy (np. 12,345),
- e – zmiennoprzecinkowy w notacji naukowej (np. 1,234E1).

4.2.9. Obsługa zasobów dyskowych. Obsługa plików

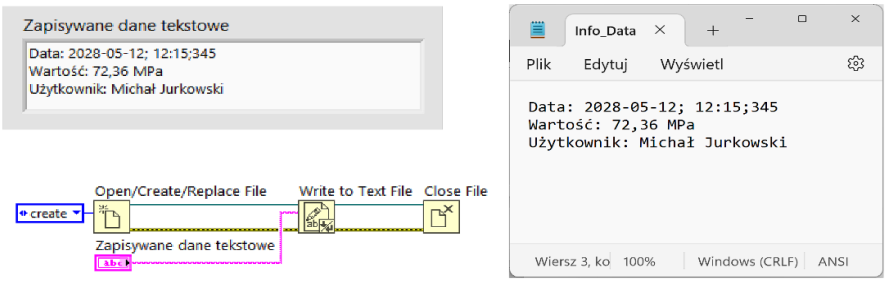
Środowisko programistyczne byłoby niepełne, gdyby nie zapewniało możliwości magazynowania danych po zakończeniu pracy programu. Takie możliwości udostępniają węzły, funkcje i podprogramy obsługi plików. Można dokonać zgrubnego podziału tych obiektów na:

- Podstawowe. Taką grupę stanowią funkcje plikowe, które wymagają od użytkownika postępowania polegającego na świadomym tworzeniu, modyfikowaniu zawartości oraz zamykaniu plików (rys. 4.29). Do takich funkcji można zaliczyć węzły: Open /Create /Replace File, Write Text File, Close Text File oraz podobne.
- Złożone. Do tej grupy funkcji wchodzą węzły, które są bardziej skomplikowane pod względem struktury wewnętrznej. Z poziomu użytkownika są łatwiejsze w obsłudze, ponieważ automatyzują proces formatowania wprowadzanych i odczytywanych danych. Do takich funkcji można zaliczyć np. podprogramy do formatowania do postaci arkusza kalkulacyjnego wraz z zapisem i odczytem Write (Read) Delimited Spreadsheet.vi, czy zapisu i odczytu danych w wewnętrznym formacie LabVIEW.
- Zaawansowane. Zastosowane dla grupy poleceń w środowisku LabVIEW pojęcie Advanced File Functions można lepiej opisać mianem funkcji szczegółowych lub cząstkowych. Są to dedykowane do wykonania bardziej szczegółowych operacji w systemie plików np. pozyskiwania informacji o atrybutach plików, tworzenia katalogów czy generowania sum kontrolnych.



Rys. 4.29. Główne subpalety funkcji plikowych

Jak uprzednio nadmieniono, typowa obsługa generowania pliku jest trzyetapowym procesem tworzenia, modyfikowania zawartości (zapisu danych) oraz zamykania pliku (rys. 4.30). Przy odczycie zawartości pliku proces ma identyczny charakter, minimalnie różniący się otwarciem pliku w pierwszym etapie, odczytem w drugim oraz zamknięciem w trzecim etapie.

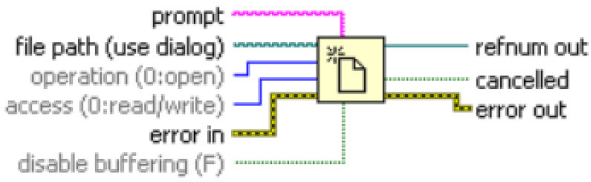


Rys. 4.30. Typowa kolejność zapisu pliku tekstowego

Opracowanie prostego programu do zapisu danych tekstowych wymaga minimalnej aktywności użytkownika. Poza wstawieniem trzech podstawowych bloków do obsługi plików oczekiwane jest (rys. 4.31, tab. 4.3):

- zadeklarowanie jednej funkcji, którą może realizować uniwersalny węzeł Open / Create /Replace File,
- dołączenie do węzła Write zapisywanych danych,
- połączenie węzłów zmiennymi Refnum i Error Cluster, w celu uniknięcia ciągłego podawania informacji o pliku oraz uniknięcia wielokrotnego reagowania na pojedynczy błąd (w przypadku jego wystąpienia).

Najbardziej rozbudowanym pod względem liczby konektorów wejściowych i wyjściowych wydaje się węzeł Open /Create /Replace File, na bazie którego zostanie przeanalizowana funkcjonalność wejść i wyjść.

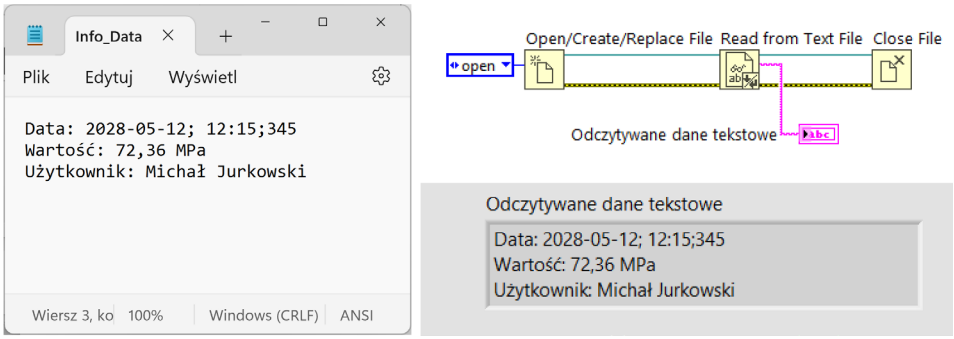


Rys. 4.31. Konektory we/wy węzła Open /Create /Replace File

Tab. 4.3. Funkcje konektorów we/wy węzła obsługi plików Open /Create /Replace File

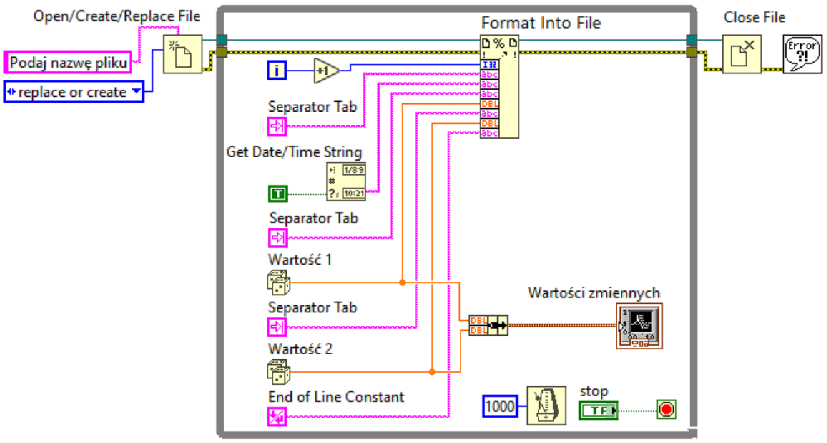
Konektor we/wy	**Funkcja**
prompt	Tekst komunikatu okna dialogowego dostępu do systemu plików
file path	Bezwzględna ścieżka dostępu do pliku. Brak wprowadzenia zmiennej spowoduje wyświetlenie okna dialogowego monitującego o podanie ścieżki dostępu. Wprowadzenie ścieżki względnej lub brak wyboru wygeneruje komunikat błędu
operation	Definiuje sposób pracy węzła. Brak definicji zmiennej generuje błąd. Typowe wartości to: 0 open (default) – otwiera istniejący plik. 1 replace – zastępuje istniejący plik. 2 create – tworzy nowy plik. 3 open or create – otwiera istniejący plik lub tworzy nowy plik w przypadku braku istniejącego. 4 replace or create – tworzy nowy plik lub nadpisuje istniejący. 5 replace or create with confirmation – tworzy nowy plik lub nadpisuje istniejący, wymagając potwierdzenia operacji przez użytkownika
access	Definiuje sposób dostępu do pliku. Domyślnie stosowany jest dostęp read/write. 0 – read/write, 1 read-only, 2 write-only
error in	Opisuje warunki istnienia błędu przed wykonaniem węzła. Pracuje w trybie standardowego klastra błędu środowiska LabVIEW tzn. nie uruchamia węzła przy występowaniu błędu, zaś informację o błędzie przekazuje dalej
disable buffering	Określa czy plik ma zostać otwarty bez buforowania. Domyślna wartość to False (z buforowaniem). W przypadku pracy z macierzami dysków, kiedy wymagany jest wysoki transfer danych do i z pliku, zalecane jest pominięcie buforowania. Brak widocznych efektów przy małych rozmiarach plików. Systemy operacyjnej OS X i Linux ignorują ten parametr
refnum	Numer (liczba) identyfikacyjny obiektu (pliku). Identyfikuje plik dla pozostałych węzłów obsługi pliku
cancelled	Przekazuje wartość TRUE, jeśli operacja plikowa zostanie anulowana przez użytkownika w oknie dialogowym wyboru pliku
error out	Przekazuje dalszym węzłom informacje o występowania błędu. Dzięki sygnalizacji występowania błędu nie uruchamia się kolejnego węzła przy występowaniu błędu

Odczyt zawartości pliku, podobnie jak zapis do pliku, jest procesem trzyetapowym. Wymaga otwarcia, odczytu oraz zamknięcia pliku (rys. 4.32).



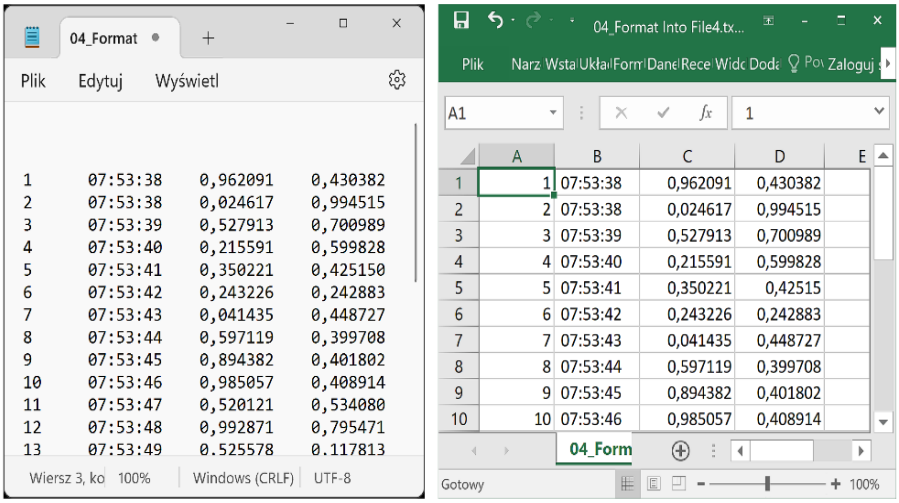
Rys. 4.32. Typowa kolejność odczytu pliku tekstowego

W grupie węzłów automatyzujących obsługę pomiarów na uwagę zasługuje funkcja Format Into File. Ten wielowejściowy węzeł pozwala na jednoczesne łączenie, formatowanie i zapis danych wielu typów (liczbowe, logiczne, łańcuchowe, znaczniki czasu itp.) (rys. 4.33). Za jego pomocą można generować pliki gromadzące dane w postaci kolumn, które obsługiwane są przez większość programów do analizy danych.



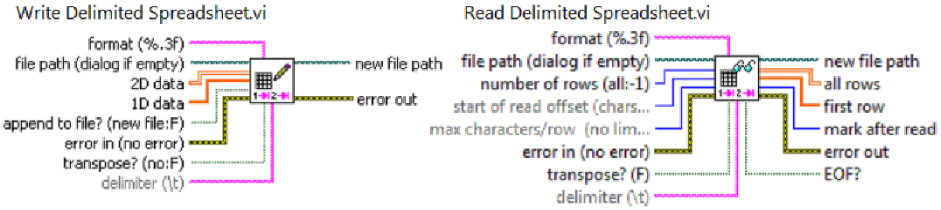
Rys. 4.33. Zastosowanie funkcji Format Into File przy zapisie danych do pliku

Wynik działania programu z rysunku 4.33 zaprezentowano na rysunku 4.34. Zgodnie z kodem działania programu, wewnątrz utworzonego pliku znajdują się kolumny zawierające liczbę porządkową (iteracja pętli), informacje o czasie, 2 wartości pseudolosowe z zakresu (0–1). Poszczególne kolumny rozdzielone są separatorem w postaci znaku tabulacji.



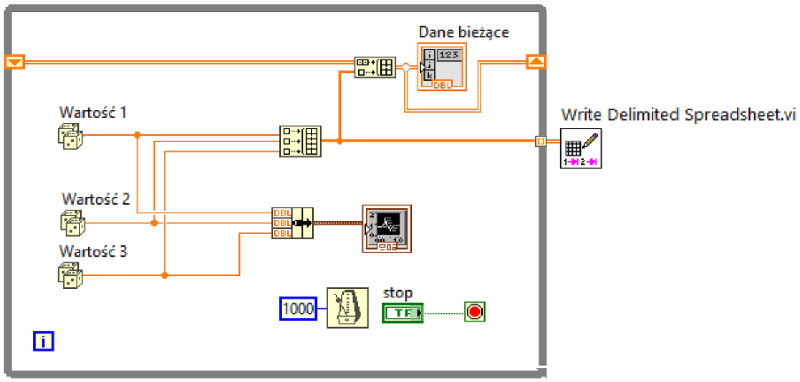
Rys. 4.34. Wynik działania programu z zastosowaną funkcją Format Into File

O ile funkcja Format Into File wymaga świadomego utworzenia (otwarcia istniejącego) pliku, sformatowania danych, połączonego z zapisem oraz zamknięcia pliku, to złożone podprogramy Write Delimited Spreadshhet.vi oraz Read Delimited Spreadshhet.vi wymagają jedynie wskazania źródła zapisu bądź odczytu danych. Węzły funkcji oferują bogaty zestaw parametryzujący działanie (rys. 4.35). Jednak nawet przy ustawieniach domyślnych parametrów możliwy jest skuteczny zapis i odczyt danych sformatowanych w postaci kolumn arkusza kalkulacyjnego.



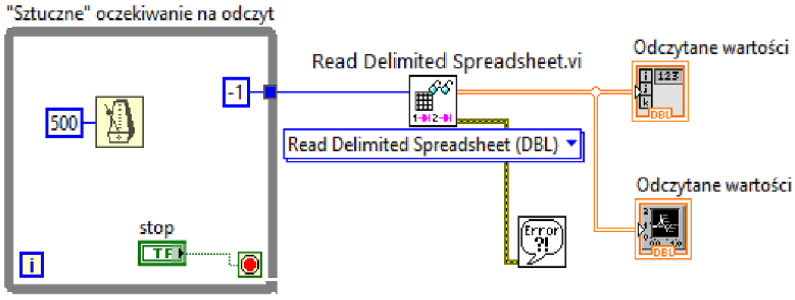
Rys. 4.35. Ikony podprogramów Write (Read) Delimited Spreadshhet.vi z pełną listą parametrów

Poniższy kod programu prezentuje przypadek zapisu 3 serii pseudolosowych danych z zakresu wartości 0–1. Wartości zmiennych kierowane są bezpośrednio do podprogramu Write Delimited Spreadshhet.vi, który nie jest skonfigurowany przed dodatkowymi parametrami (rys. 4.36).

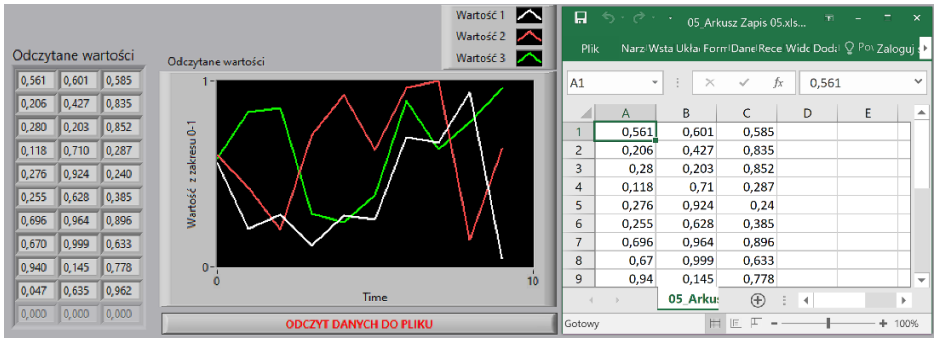


Rys. 4.36. Zapis danych za pomocą Write Delimited Spreadshhet.vi

Wartości zapisane podprogramem Write Delimited Spreadshhet.vi można odczytać bliźniaczym podprogramem Read Delimited Spreadshhet.vi (rys. 4.37), arkuszem kalkulacyjnym (rys. 4.38), czy dowolnym programem obsługującym dane typu łańcuchowego.



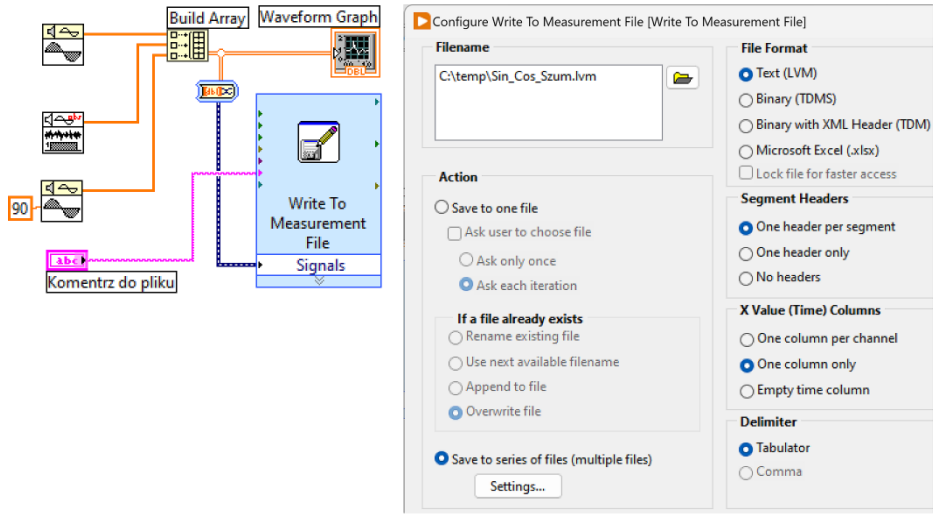
Rys. 4.37. Kod programu do odczyt danych za pomocą Read Delimited Spreadshhet.vi



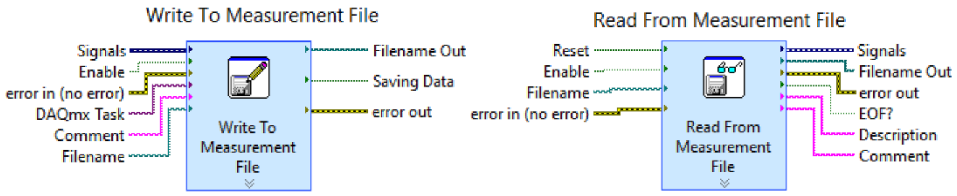
Rys. 4.38. Odczyt danych za pomocą Read Delimited Spreadshhet.vi oraz arkusza kalkulacyjnego

Jeszcze dalej posunięte w ułatwianiu rejestracji danych są gotowe podprogramy do rejestracji danych w wewnętrznym formacie LabVIEW, np. w plikach o rozszerzeniu „lvm" (LabVIEW Measurement File), formatach TDM, TDMS lub formacie arkusza XLSX. Programy typu Express VI o nazwie „Write To (Read From) Measurement File.vi", podobnie jak podprogramy Write Delimited Spreadshhet.vi oraz Read Delimited Spreadshhet.vi, wymagają minimalnej konfiguracji przez użytkownika (rys. 4.40). Jeśli taka konfiguracja jest wymagana, przeprowadza się ją za pomocą arkusza właściwości podprogramu typu express VI, czyli poprzez edycję panelu programu.

Na rysunku 4.39 zamieszczono kod programu pozwalającego na zapis trzech serii danych prezentujących dane funkcji trygonometrycznych wraz z funkcją szumu oraz okno konfiguracji programu Write To Measurement File.vi.

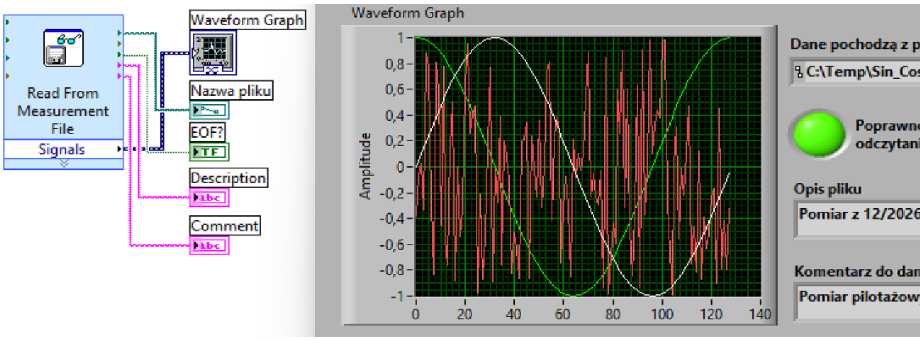


Rys. 4.39. Zapis danych za pomocą Write To Measurement File.vi z elementem okna konfiguracyjnego



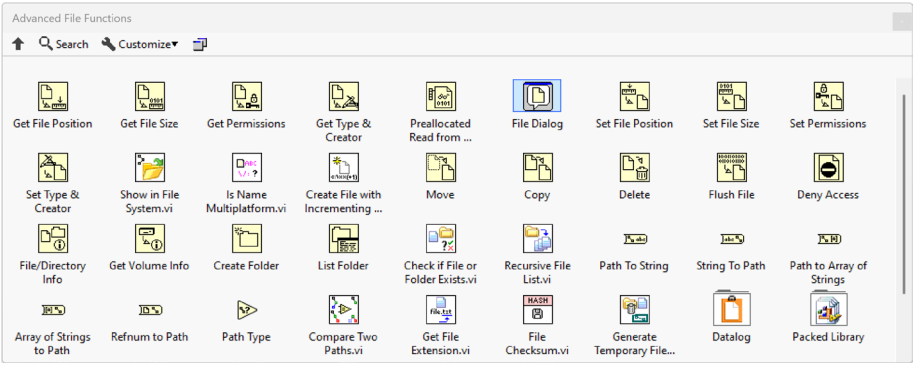
Rys. 4.40. Ikony podprogramów express VI Write To (Read From) Measurement File.vi

Rysunek 4.41 prezentuje minimalny kod programu wymagany do odczytu wartości zapisanych w programie z rysunku 4.39.

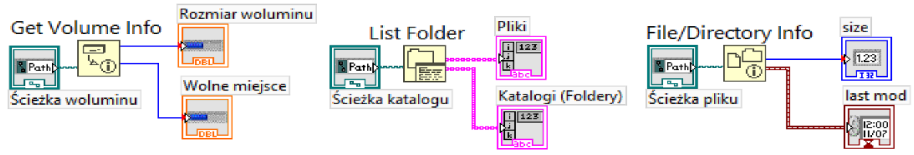


Rys. 4.41. Odczyt danych za pomocą Read From Measurement File.vi

Funkcje i podprogramy zgrupowane na palecie Advanced File Functions są przeznaczone do wykonania szczegółowych operacji w systemie plików, np. pozyskiwania informacji o atrybutach plików, tworzenia katalogów czy generowania sum kontrolnych (rys. 4.42). Ich funkcjonalność jest wyraźnie wskazana w nazwie węzła lub podprogramu.

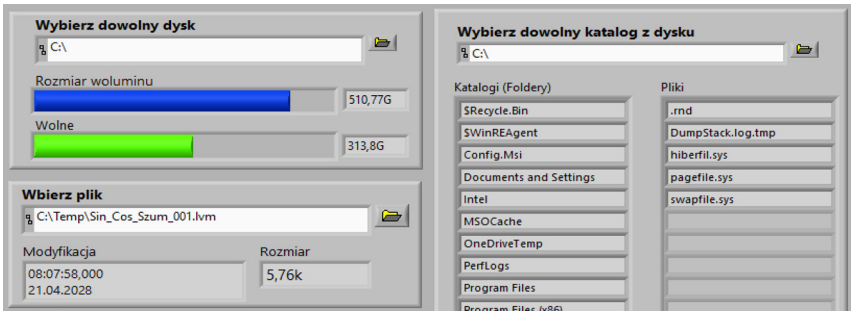


Rys. 4.42. Subpaleta „zaawansowanych" funkcji plików



Rys. 4.43. Wybrane „zaawansowane" funkcje w kodzie programu

W przypadku kodu programu z rys. 4.43 zastosowano 3 wybrane, informacyjne plikowe funkcje „zaawansowane". Zgodnie z nazwą wyświetlają one parametry wskazanego woluminu, prezentują zawartość wskazanego katalogu oraz informacje o wybranym pliku (rys. 4.44).



Rys. 4.44. Panel czołowy aplikacji wyświetlającej dane o woluminie, pliku i zawartości woluminu

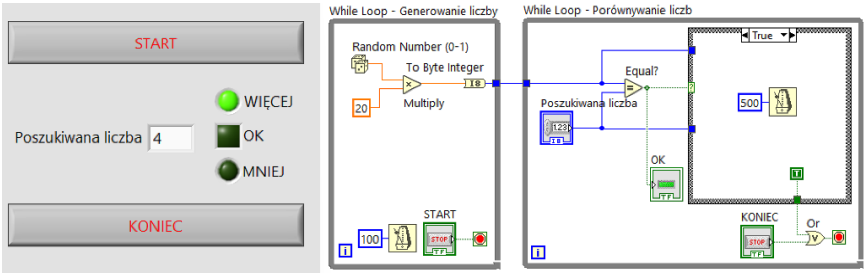
4.3. Zadania

4.3.1. Struktura wyboru

Cel zadania: wprowadzenie do korzystania z węzła struktury wyboru. W ćwiczeniu zostanie wykorzystana logiczna struktura wyboru.

Zakres zadania: opracowywany program będzie generował liczbę pseudolosową z zakresu 0–20. Zadaniem użytkownika będzie wprowadzenie liczby o wartości zgodnej z wygenerowaną przez program. Struktura wyboru wykorzystana będzie do porównywania liczby wprowadzanej przez użytkownika z wygenerowaną programowo oraz sterowania warunkiem zakończenia pętli While, a zarazem całego programu.

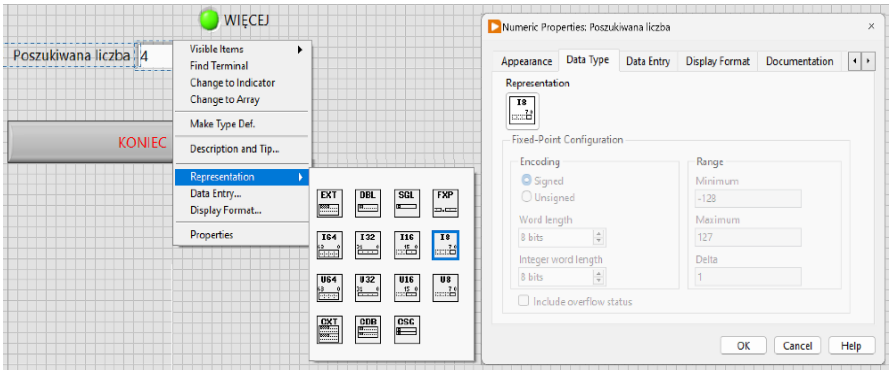
1. Uruchomić środowisko LabVIEW z pustym plikiem programu (Blank.vi). Celem kolejnych działań będzie utworzenie panelu czołowego wraz z kodem, zaprezentowanymi na kolejnych rysunkach.



2. **Budowa panelu czołowego**. Z niżej wymienionych palet wybrać i sformatować poniższe elementy panelu:

- dwie kontrolki logiczne – przyciski Stop (palety Controls /Modern /Boolean i Stop Button). Zmienić nazwę jednego przycisku na START, zaś drugiego na KONIEC,

- dwa wskaźniki logiczne – okrągłe diody LED (palety Controls /Modern /Boolean i Round LED). Zmienić nazwę jednej diody na WIĘCEJ, zaś drugiej na MNIEJ,
- jeden wskaźnik logiczny – kwadratową diodę LED (palety Controls /Modern / Boolean i Square LED). Zmienić nazwę diody na OK,

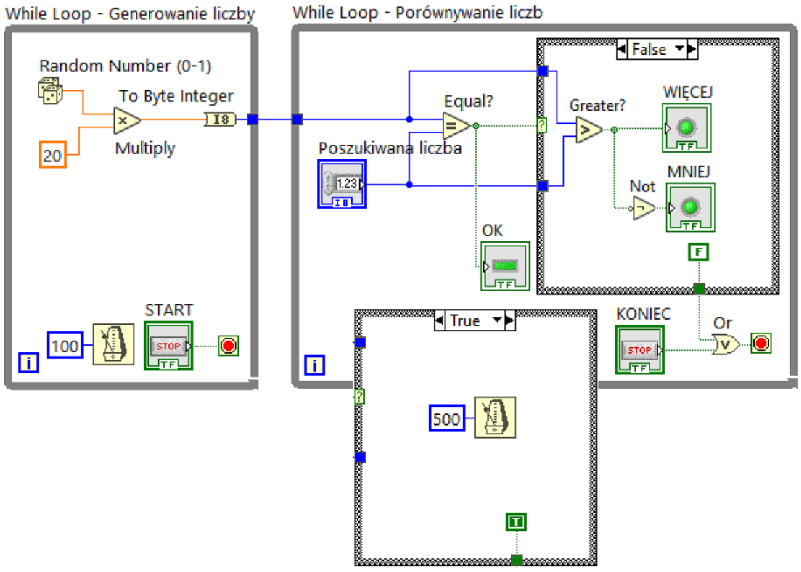


- zmienną liczbową Numeric Controls (palety Controls /Modern /Numeric I Horizontal Numeric Control). Zmienić nazwę na Poszukiwana liczba. Usunąć strzałki zmiany wartości (menu kontekstowe /Visible Items /Icreament /Decrement (nieaktywne),
- zmienić typ danych zmiennej na wartości całkowite np. I8 (menu kontekstowe / Representation /I8). Zadeklarować zakres wartości zmiennej na 0–20 z krokiem 1 (menu kontekstowe /Properties /karta Data Entry /Minimum = 0 /Maksimum = 20 / Increament = 1 /Response to value outside limits = Coerce.

3. Uporządkować elementy na panelu czołowym. Dopasować rozmiary tak, aby były zbliżone do zamieszczonych na rysunku.

4. Zapisać plik programu pod nazwą (041_Podaj_liczbe.vi) w lokalizacji wskazanej przez prowadzącego.

5. **Schemat blokowy programu**. Przejść do okna schematu blokowego.
6. Aplikacja zawiera dwie pętle While i jedną strukturę wyboru Case Structure. W pierwszej pętli generowana jest liczba pseudolosowa z zakresu 0–20. Wartość zmiennoprzecinkowa konwertowana jest na całkowitą. W chwili zakończenia pierwszej pętli (przycisk START) wygenerowana wartość przekazywana jest tunelem do drugiej pętli i porównywana z wartością wprowadzoną z panelu przez użytkownika. Jeśli wartości są zgodne (spełniony warunek struktury wyboru True), to po zwłoce równej 500 ms stała logiczna kończy pracę drugiej pętli i całego programu.



Jeśli porównanie skutkuje wartością False, to wartość wygenerowana porównywana jest z wartością wprowadzoną przez użytkownika, a wynik porównania prezentowany jest za pomocą wskaźników logicznych (diód) i wykonywana jest kolejna pętla pozwalająca na porównanie z kolejną wartością wprowadzoną z panelu. Drugą z pętli można zakończyć przyciskiem STOP, mimo że wartość wprowadzona i wygenerowana różnią się.

7. Zamieszczanie funkcji i stałych matematycznych. Z głównej palety Programming wybrać subpaletę Numeric, a z niej:

- funkcję mnożenia Multiply,
- stałą Numeric Constant (3 obiekty), a następnie zadeklarować wartości 20, 100 oraz 500,
- węzeł Random Number (0–1).

8. Zamieścić dwie pętle While (Functions /Programming /Structures /While Loop).

9. Wprowadzić węzły:
- opóźnienia – dwa elementy Wait Until Next ms Multiple (Programming /Timing oraz Wait Until Next ms Multiple),
- konwersji wartości zmiennoprzecinkowej do całkowitej To Byte Integer (Programming /Numeric /Conversion oraz To Byte Integer).

10. Wprowadzić węzły funkcji porównań i funkcji binarnych (Programming /Boolean):
- porównania Equal?,
- porównania Greater?,
- negacji Not,
- sumy logicznej Or,
- stałych logicznych True Constant i False Constant.

11. Zamieścić w kodzie strukturę wyboru – (Functions /Programming /Structures / Case Structure).
12. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
13. Zapisać plik programu pod aktualną nazwą (041_Podaj_liczbe.vi).
14. Uruchomić program w trybie pojedynczym (jednokrotnym):
- wygenerować liczbę przyciskiem START,
- wprowadzać kolejne wartości, aż do uzyskania zgodności liczby wygenerowanej i wprowadzonej oraz zakończenia programu.

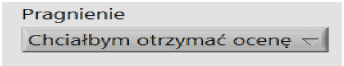
15. Ponownie uruchomić program w trybie pojedynczym (jednokrotnym):
- wygenerować liczbę przyciskiem START,
- zakończyć działanie programu przyciskiem KONIEC.

16. Zamknąć plik programu po zakończeniu testowania.

4.3.2. Wielokrotna struktura wyboru

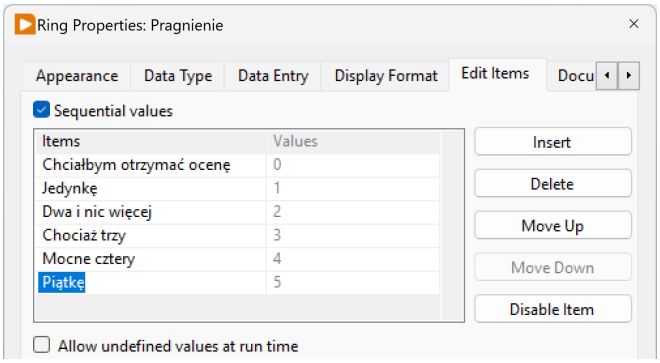
Cel zadania:	utrwalenie korzystania ze struktur wyboru. W ćwiczeniu zostanie użyta struktura umożliwiająca podejmowanie decyzji w więcej niż dwóch przypadkach.
Zakres zadania:	zadaniem programu będzie generowanie komunikatów adekwatnych do aktywności wykazywanej przez użytkownika. Zostanie wykorzystany węzeł struktury wyboru sterowany za pomocą wartości całkowitych.

1. Uruchomić środowisko LabVIEW z pustym plikiem programu (Blank.vi). Celem kolejnych działań będzie utworzenie panelu czołowego wraz z kodem zaprezentowanymi na kolejnych rysunkach.



2. **Budowa panelu czołowego**. Z niżej wymienionej palety wybrać i sformatować poniższy element panelu:

- menu rozwijalne (menu ing) (palety Controls /Modern /Ring & Enum i Menu Ring). Zmienić nazwę menu na Pragnienie,
- wprowadzić opcje Menu Ring, za pomocą arkusza właściwości menu i karty Edit Items, takie jak na poniższym rysunku.

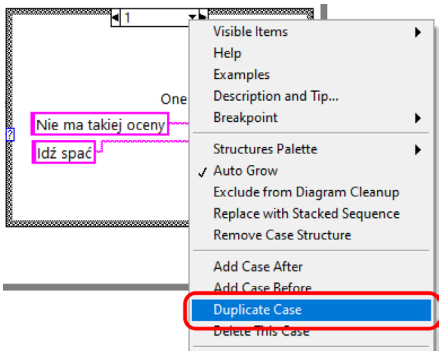


3. Zapisać plik programu pod nazwą (042_Ocena.vi) w lokalizacji wskazanej przez prowadzącego.

4. **Schemat blokowy programu**. Przejść do okna schematu blokowego.
5. Aplikacja zawiera pętlę While obejmującą jedną strukturę wyboru – Case Structure. Struktura Case posiada sześć przypadków kodu, wykonywanego w zależności od opcji wybranej przez użytkownika z rozwijalnej listy menu Pragnienie. Wybraniu konkretnej opcji towarzyszy wygenerowanie zmiennej w postaci liczby całkowitej, odpowiadającej przypadkowi ze struktury case. W przypadku domyślnym, oznaczonym jako 0, przypadek dostarcza zmiennej logicznej False pozwalającej na wykonanie kolejnej pętli i ponowne „odczytanie" stanu menu Pragnienie. W pozostałych przypadkach wyświetlany jest tekst okna dialogowego. Zamknięcie okna dialogowego skutkuje wygenerowaniem zmiennej True, która kończy działanie pętli programu.
6. Zamieścić pętlę While (Functions /Programming /Structures /While Loop).
7. Wprowadzić węzeł opóźnienia – Wait Until Next ms Multiple (Programming / Timing oraz Wait Until Next ms Multiple). Za pomocą menu kontekstowego zadeklarować wartość stałej opóźnienia na 500 ms.
8. Zamieścić w kodzie strukturę wyboru – (Functions /Programming /Structures / Case Structure).
9. W pierwszej, domyślnej (zerowej), ramce struktury wyboru zamieścić stałą logiczną Fałsz – False Constant (Programming /Boolean).

10. W kolejnej ramce struktury zamieścić okno dialogowe z jednym przyciskiem (Programming /Dialog & User Interface /One Button Dialog. Za pomocą menu kontekstowego (prawy przycisk myszy) zadeklarować teksty komunikatu i przycisku.
11. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).

12. Kolejne ramki struktury wyboru można uzyskać wstawiając nowe (ramki) przypadki i zamieszczając w nich odpowiednią zawartość lub przez kopiowanie i modyfikowanie raz zamieszczonego przypadku.



13. Zapisać plik programu pod aktualną nazwą (042_Ocena.vi).

14. **Uruchamianie programu**. Uruchomić program w trybie pojedynczym (jednokrotnym):
- wybrać jedną z opcji menu i sprawdzić, czy został wyświetlony zadeklarowany uprzednio komunikat i czy program samoczynnie zakończył działanie,
- uruchomienie powtórzyć dla każdej z opcji struktury wyboru.

15. Zamknąć plik programu po zakończeniu testowania.

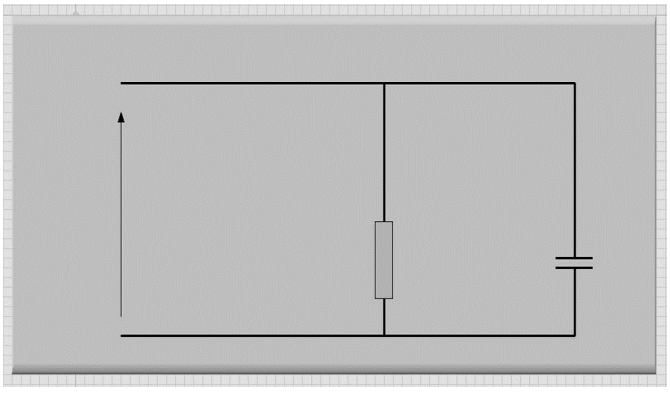
4.3.3. Węzeł formuły. Obwód równoległy RC

Cel zadania: wprowadzenie do korzystania z obiektu węzła formuły Formula Node. Węzeł Formula Node wykorzystywany będzie do prowadzenia operacji obliczeniowych w kodzie programu.

Zakres zadania: celem ćwiczenia będzie opracowanie programu, który pozwala na przeprowadzenie symulacji pracy obwodu równoległego RC zasilanego prądem sinusoidalnym.

1. Uruchomić środowisko LabVIEW z pustym plikiem programu (Blank.vi).
2. **Budowa panelu czołowego**. Wykorzystując narzędzia z palety Decorations (palety Controls /Modern /Decorations), sporządzić schemat obwodu równoległego RC podobny do zamieszczonego na rysunku.
   W prezentowanym przypadku wykorzystano:

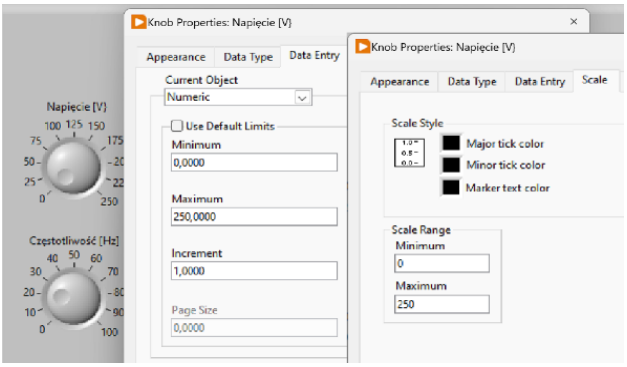
- Thin Line,
- Flat Box,
- Thin Line With Arrow,
- Vertical Smooth Box.



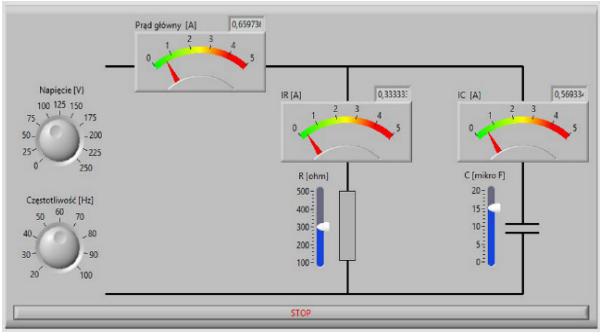
3. Zapisać plik programu pod nazwą (043_Rownolegle_RC.vi) w lokalizacji wskazanej przez prowadzącego.
4. Na panelu czołowym zamieścić i skonfigurować poniższe elementy:

- kontrolkę liczbową (gałkę) Knob (palety Controls /Modern /Numeric I Knob). Zmienić nazwę zmiennej na „Napięcie [V]”. Za pomocą kart arkusza właściwości zmodyfikować: zakres skali na 0–250, zakres wartości 0–250 z interwałem 1 z zaokrąglaniem do najbliższej wartości. Dopasować rozmiar gałki tak, aby uzyskać przejrzysty opis wartości gałki.

- kontrolkę liczbową (gałkę) Knob (palety Controls /Modern /Numeric i Knob). Zmienić nazwę zmiennej na „Częstotliwość [Hz]”. Za pomocą kart arkusza właściwości zmodyfikować: zakres skali na 0–100, zakres wartości 0–100 z interwałem 1 z zaokrąglaniem do najbliższej wartości. Dopasować rozmiar gałki tak, aby uzyskać przejrzysty opis wartości gałki.



- wskaźniki liczbowe Meter (palety Controls /Modern /Numeric i Meter). Będą zamieszczone trzy tego rodzaju wskaźniki. Zmienić ich nazwy na „Prąd główny [A]”, „IR [A]” oraz „IC [A]”. Za pomocą edycji w miejscu lub karty „Scale” arkusza właściwości zmodyfikować: zakres skali na 0–5. Dla każdego wskaźnika wyświetlić dodatkowy wskaźnik liczbowy (menu kontekstowe /Visible Items /Digital Display).
- kontrolki liczbowe (suwaki) Vertical Fill Slide (palety Controls /Modern /Numeric i Vertical Fill Slide). Zmienić nazwę pierwszej kontrolki na „R [ohm]” oraz zadeklarować zakres na 100–500. Zmienić nazwę drugiej kontrolki na „C [mikro F]” oraz zadeklarować zakres na 0–20.
- przycisk Stop (palety Controls /Modern /Numeric /Boolean i Stop Button).



5. Zapisać plik pod istniejącą nazwą (043_Rownolegle_RC.vi).

6. **Schemat blokowy programu**. Przejść do okna schematu blokowego.
7. Zamieścić pętlę While (Functions /Programming /Structures /While Loop).
8. Wprowadzić węzeł opóźnienia wykonywania pętli Wait (ms) (Programming / Timing oraz Wait (ms)). Za pomocą menu kontekstowego zadeklarować wartość stałej opóźnienia na 100 ms.
9. Zamieszczanie funkcji i stałych matematycznych. Z głównej palety Programming wybrać subpaletę Numeric, a z niej:

- funkcję dzielenia Divide. Będzie ona pozwalała na wprowadzanie wartości pojemności w jednostkach całkowitych, a przy podzieleniu przez 10^6 na operowanie w wartościach μF,
- stałą Numeric Constant i zadeklarować wartość 1000000.

10. Węzeł formuły (Formula Node):

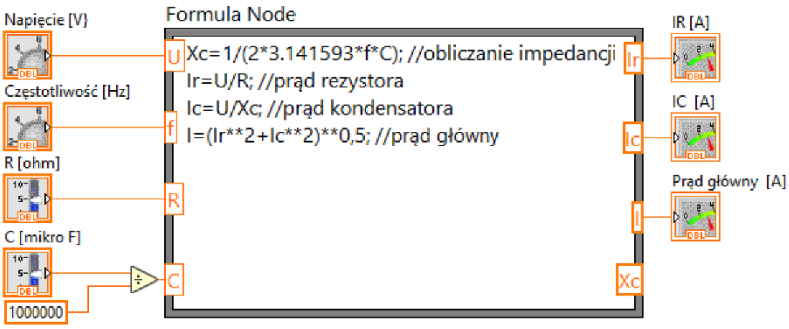
- zamieszczanie węzła (Functions /Programming /Structures /Formula Node),
- deklarowanie zmiennych wejściowych. Za pomocą menu kontekstowego wywoływanego na krawędzi pętli i polecenia Add Input zadeklarować zmienne *U*, *F*, *R*, *C*,
- deklarowanie zmiennych wejściowych. Za pomocą menu kontekstowego wywoływanego na krawędzi pętli i polecenia Add Ooutput zadeklarować zmienne *Ir*, *Ic*, I oraz *Xc*,
- w polu formuły wpisać wyrażenia:

$$X_c = \frac{1}{2 \cdot 3.141593 \cdot f \cdot C},$$
$$I_r = \frac{U}{R},$$
$$I_c = \frac{U}{X_c},$$
$$I = \sqrt{I_r^2 + I_c^2}.$$

Przy deklarowaniu zmiennych i wpisywaniu zależności wewnątrz węzła formuły należy pamiętać o tym, że:

- rozróżniane są wielkości znaków,
- separatorem dziesiętnym jest kropka,
- każde wyrażenie powinno być zakończone średnikiem,
- jeśli wewnątrz wyrażenia węzła formuły stosowana jest zmienna, która nie będzie potem wykorzystywana w programie, to i tak musi zostać ona zdefiniowana.

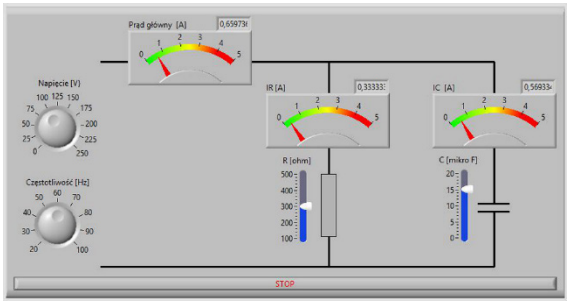
11. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).



12. Zapisać plik programu pod aktualną nazwą (043_Rownolegle_RC.vi).

13. **Uruchamianie programu**. Program pozwala na symulowanie zachowania obwodu równoległego RC przy zmianach wartości napięcia i częstotliwości zasilającego, wartości rezystancji i pojemności występujących w obwodzie. Możliwa jest obserwacja wartości natężenia prądów płynących przez każdy z elementów oraz prądu stanowiącego sumę dwóch prądów składowych.
14. Uruchomić program w trybie pojedynczym (jednokrotnym).

- Zadeklarować wartości: napięcia = 230 V, częstotliwości = 50 Hz, rezystancji = 100 Ω, pojemności = 10 μF.
- Porównać otrzymane wyniki z wynikami uzyskanymi na innym stanowisku ćwiczeniowym.



15. Przeprowadzić obserwacje dla samodzielnie zadeklarowanych wartości, ze szczególnym uwzględnieniem oddzielnych przypadków, w których częstotliwość = 0 Hz, pojemność = 0 μF.
16. Dla zmiennych wejściowych, za pomocą poleceń menu kontekstowego Data Operations /Make Current Value Default, zadeklarować wartości domyślne kontrolek na U = 200 V, f = 100 Hz, R = 100 Ω oraz C = 20 μF.
17. Zapisać plik programu pod aktualną nazwą (043_Rownolegle_RC.vi) oraz zamknąć plik programu po zakończeniu testowania.

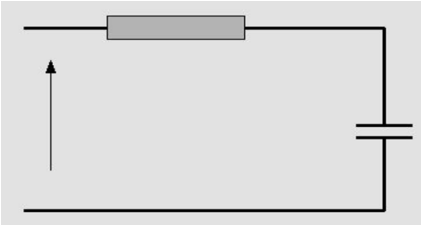
4.3.4. Węzeł formuły. Ładowanie kondensatora

Cel zadania: utrwalenie umiejętności korzystania z obiektu węzła formuły Formula Node. Węzeł Formula Node wykorzystywany będzie do prowadzenia operacji obliczeniowych w kodzie programu.

Zakres zadania: celem ćwiczenia będzie opracowanie programu, który pozwala na przeprowadzenie symulacji procesu ładowania kondensatora w obwodzie szeregowym RC zasilanym napięciem stałym.

1. Uruchomić środowisko LabVIEW z pustym plikiem programu (Blank.vi).
2. **Budowa panelu czołowego**. Wykorzystując narzędzia z palety Decorations (palety Controls /Modern /Decorations) sporządzić schemat obwodu szeregowego RC podobny do zamieszczonego na rysunku. W prezentowanym przypadku wykorzystano:

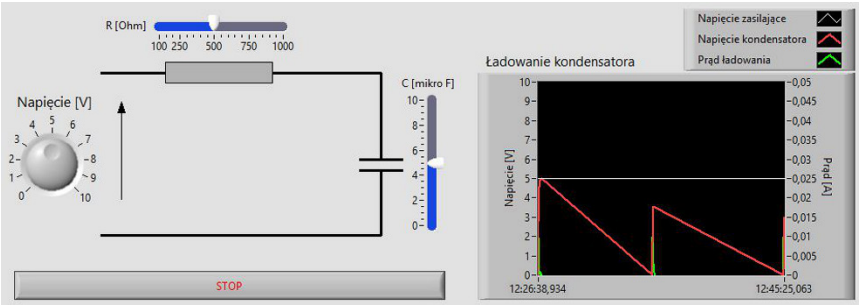
- Thin Line,
- Flat Box,
- Thin Line With Arrow.



3. Zapisać plik programu pod nazwą (044_Ladowanie_C.vi) w lokalizacji wskazanej przez prowadzącego.
4. Na panelu czołowym zamieścić i skonfigurować poniższe elementy:

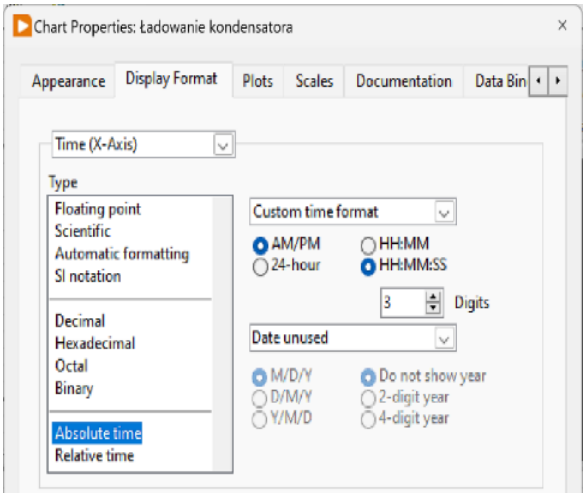
- kontrolkę liczbową (gałkę) Knob (palety Controls /Modern /Numeric I Knob). Zmienić nazwę zmiennej na „Napięcie [V]”. Za pomocą kart arkusza właściwości zmodyfikować: zakres skali na 0–10, zakres wartości 0–10 z interwałem 1 z zaokrąglaniem do najbliższej wartości. Deklaracja wartości domyślnej na 5 V: ustawić wartość kontrolki na 5, a następnie z menu kontekstowego wybrać polecenie Data Operations /Make Current Value Default. Dopasować rozmiar gałki tak, aby uzyskać przejrzysty opis wartości gałki.
- kontrolkę liczbową (suwak) Vertical Fill Slide (palety Controls /Modern /Numeric i Vertical Fill Slide). Zmienić nazwę pierwszej kontrolki na „R [ohm]”. Za pomocą kart arkusza właściwości zmodyfikować: zakres skali na 100–1000, zakres wartości 100-1000 z interwałem 1 z zaokrąglaniem do najbliższej wartości. Deklaracja wartości domyślnej na 500 Ω: ustawić wartość kontrolki na 500, a następnie z menu kontekstowego wybrać polecenie Data Operations /Make Current Value Default,

- kontrolkę liczbową (suwak) Horizontal Fill Slide (palety Controls /Modern / Numeric i Horizontal Fill Slide). Zmienić nazwę pierwszej kontrolki na „C [mikro F]”. Za pomocą kart arkusza właściwości zmodyfikować: zakres skali na 0–10, zakres wartości 0–10 z interwałem 1 z zaokrąglaniem do najbliższej wartości. Deklaracja wartości domyślnej na 5 μF: ustawić wartość kontrolki na 5, a następnie z menu kontekstowego wybrać polecenie Data Operations /Make Current Value Default,
- przycisk Stop (palety Controls /Modern /Numeric /Boolean i Stop Button.



5. Na panelu czołowym zamieścić i skonfigurować obiekt wykresu postępując zgodnie z poniższymi instrukcjami:

- zamieścić obiekt wykresu Waveform Chart (palety Controls /Modern /Graph oraz Waveform Chart),
- zmienić tytuł wykresu na „Ładowanie kondensatora”,
- poszerzyć legendę przebiegów do trzech pozycji i nadać im nazwy „Napięcie zasilające, „Napięcie kondensatora”, „Prąd ładowania”,
- dodać dodatkową oś dla „Prądu ładowania” – menu kontekstowe osi Y /Duplicate Scale. Zostanie zamieszczona druga oś Y. Menu kontekstowe dodatkowej osi Y i polecenie kontekstowe Swap Sides,
- modyfikacja pierwszej osi Y: wyłączyć autoskalowanie (menu kontekstowe / Autoscale Y ma pozostać nieaktywne), poprzez edycję w miejscu zmienić nazwę na „Napięcie [V]”, poprzez edycję w miejscu zmienić zakres na 0–10,
- modyfikacja drugiej (dodatkowej) osi Y: wyłączyć autoskalowanie (menu kontekstowe /Autoscale Y ma pozostać nieaktywne), poprzez edycję w miejscu zmienić nazwę na „Prąd [A]”, poprzez edycję w miejscu zmienić zakres na 0–0,05,
- modyfikacja osi X: usunięcie nazwy poprzez menu kontekstowe i polecenie Visible Scale Legend, zmiana sposobu opisu osi poprzez arkusz właściwości i kartę Display format na format czasu z trzema miejscami precyzji,

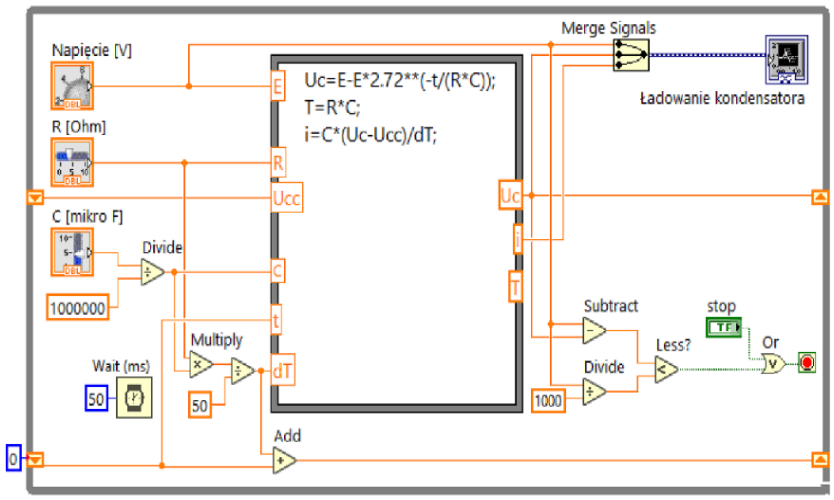


6. Zapisać plik pod istniejącą nazwą (044_Ladowanie_C.vi).

7. **Schemat blokowy programu**. Przejść do okna schematu blokowego.
8. Zamieścić pętlę While (Functions /Programming /Structures /While Loop).
9. Wprowadzić węzeł opóźnienia wykonywania pętli Wait (ms) (Functions /Programming /Timing oraz Wait (ms)). Za pomocą menu kontekstowego zadeklarować wartość stałej opóźnienia na 100 ms.
10. Zamieścić funkcje matematyczne. Z głównej palety Programming, wybrać subpaletę Numeric, a z niej:

- funkcję dzielenia Divide (3 węzły),
- funkcję dzielenia Multiply (1 węzeł),
- funkcję dodawania Add (1 węzeł),
- funkcję odejmowania Subtract (1 węzeł),

11. Zamieścić wymienione dalej stałe. Wartości stałych należy zamieszczać, korzystając z poleceń kontekstowych wejść węzłów. Wartość stałych należy deklarować poprzez edycję w miejscu.
12. Zamieścić funkcję logiczną Or. Z głównej palety Functions /Programming wybrać subpaletę Boolean, a z niej węzeł Or (Lub).
13. Zamieścić funkcję porównania. Z głównej palety Functions /Programming wybrać subpaletę Comparison, a z niej węzeł Less?
14. Zamieścić węzeł łączenia sygnałów Merge (Functions /Express /Signal Manipulation oraz Merge Signals). Zmianę liczby wejść uzyskuje się poprzez zmianę rozmiaru ikony obiektu („rozciąganie” w pionie).



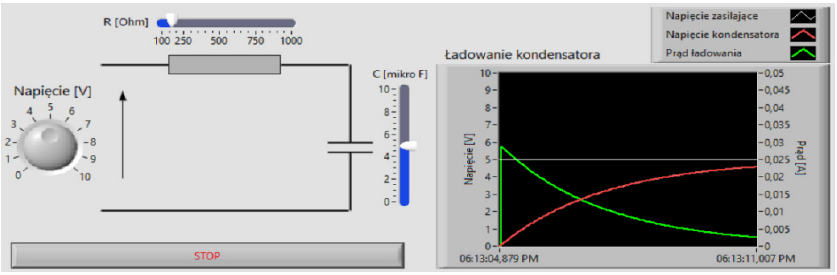
15. Zamieścić rejestry przesuwne. Wybrać prawym klawiszem myszy pionową krawędź pętli While oraz polecenie Add Shift Register. Przed przyłączeniem do rejestru zmiennej symbol rejestru pozostaje czarny. Operację powtórzyć dla kolejnego rejestru.
16. Wprowadzić i skofiguować węzeł formyły (Formula Node) zgodnie z poniższymi instrukcjami:

- zamieszczanie węzła (Functions /Programming /Structures /Formula Node),
- deklarowanie zmiennych wejściowych. Za pomocą menu kontekstowego wywoływanego na krawędzi pętli i polecenia Add Input zadeklarować zmienne *E*, *R*, *Ucc*, *C*, *t* oraz *dT*,
- deklarowanie zmiennych wejściowych. Za pomocą menu kontekstowego wywoływanego na krawędzi pętli i polecenia Add Ooutput zadeklarować zmienne *Uc*, *i* oraz *T*,
- w polu formuły wpisać wyrażenia:

$$U_C = E - E * 2.72 ** (-t/(R * C)),$$
$$T = R * C,$$
$$i = C * (U_C - U_{CC})/dT.$$

17. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
18. Zapisać plik programu pod aktualną nazwą (044_Ladowanie_C.vi).

19. **Uruchamianie programu**. Program pozwala na symulowanie procesu ładowania kondensatora w typowym, szeregowym układzie RC. Użytkownik może modyfikować wartości napięcia zasilającego, wartości rezystancji i pojemności występujących w obwodzie. Obiekt wykresu przedstawia stałą wartość napięcia zasilającego, wartość napięcia na zaciskach kondensatora oraz natężenie prądu ładującego. Program funkcjonju do momentu, w którym wartość napięcia na kondensatorze nie różni się o mniej niż 0,1% napięcia zasilającego. Działanie programu można w dowolnym momencie przerwać przyciskiem Stop.
20. Zadeklarować wartości napięcia, rezystancji oraz pojemności (np. 5 V, 500 Ω, 5 μF).
21. Uruchomić program w trybie pojedynczym (jednokrotnym).
22. Obserwować przebieg czasowy napięcia zasilającego, napięcia kondensatora oraz natężenie prądu ładującego. Program powinien samoczynnie zakończyć działanie, kiedy napięcie kondensatora zbliży się do napięcia zasilającego. W przypadku „długotrwałego ładowania" zakończyć program przyciskiem Stop.
23. „Wyczyścić" pole wykresu poleceniem kontekstowym Data Operations /Clear Chart.
24. Zadeklarować nowe wartości napięcia, rezystancji oraz pojemności. Dokonać ponownej obserwacji procesu ładowania w nowych warunkach.
25. Wyciągnąć wnioski w odniesieniu do wpływu wzrostu wartości rezystancji i pojemności na prąd i czas ładowania.



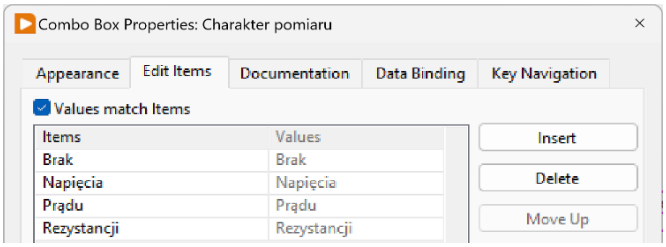
26. Zamknąć plik programu po zakończeniu testowania.

4.3.5. Zmienne łańcuchowe

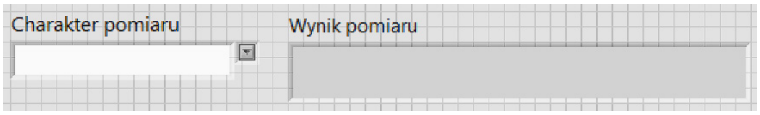
Cel zadania:	nabycie umiejętności korzystania ze zmiennych łańcuchowych łączących wartości tekstowe i liczbowe. Utrwalenie umiejętności wykorzystania wybranych stałych i funkcji zmiennych łańcuchowych.
Zakres zadania:	opracowywanie programu pozwalającego na wyświetlanie komunikatów łączących losowo generowaną, odpowiednio sformatowaną wartość liczbową z informacją o czasie pobrania danych i charakterze „pomiaru".

1. **Budowa panelu czołowego**. Uruchomić środowisko LabVIEW z pustym plikiem programu. Z niżej wymienionych palet wybrać i sformatować poniższe elementy panelu:

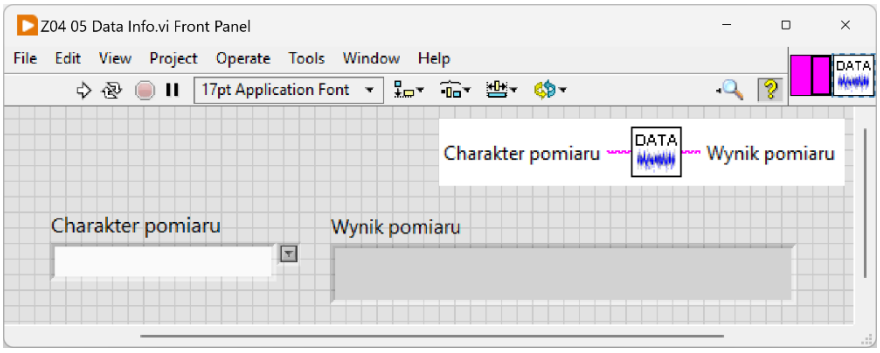
- pole Combo (palety Controls /Modern /String & Path i Combo Box). Zmienić nazwę pola na „Charakter pomiaru". Zadeklarować wartości pola combo jako: Brak, Napięcia, Prądu oraz Rezystancji. Deklarację przeprowadza się przez kartę Edit Items arkusza właściwości obiektu,



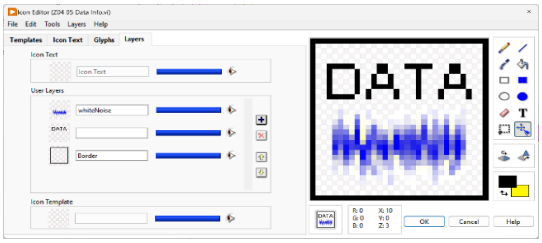
- kontrolka wyjściowa zmiennej łańcuchowej String Indicator (palety Controls / Modern /String & Path i String Indicator). Zmienić nazwę obiektu na „Wynik pomiaru".



2. Uporządkować elementy na panelu czołowym. Dopasować rozmiary tak, aby były zbliżone do zamieszczonych na rysunku.
3. Deklaracja zmiennych wejściowych i wyjściowych. Za pomocą menu kontekstowego panelu konektorów wybrać szablon konektora z jednym wejściem i jednym wyjściem oraz połączyć je z odpowiednimi zmiennymi panelu czołowego.



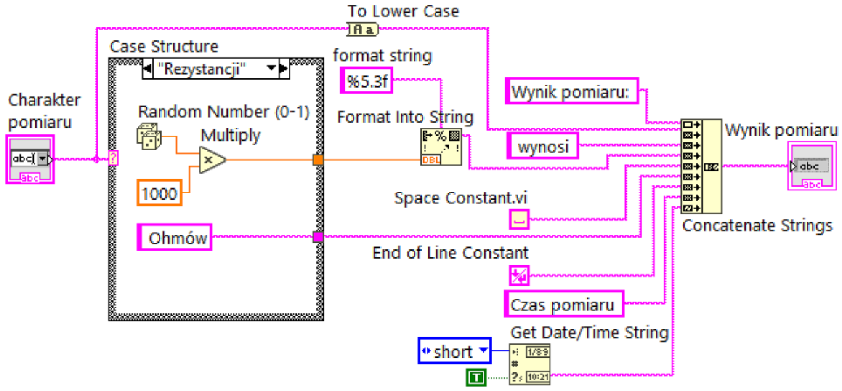
4. Opracowanie ikony. Za pomocą Edytora ikon sporządzić oryginalną ikonę programu optycznie zbliżoną do zaprezentowanej na rysunku.



5. Zapisać plik programu pod nazwą (045_Data_Info.vi) w lokalizacji wskazanej przez prowadzącego.

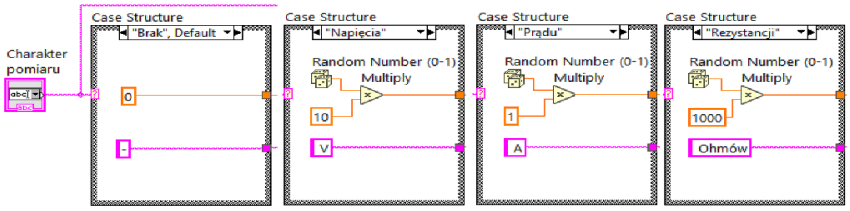
6. **Schemat blokowy programu**. Przejść do okna schematu blokowego. Aktualnie kod zawiera jedynie dwa elementy stanowiące odpowiedniki obiektów umieszczonych na panelu aplikacji.
7. Zamieszczanie funkcji i stałych łańcuchowych. Z głównej palety Programming wybrać subpaletę String, a z niej:

- funkcję łączenia łańcuchów Concatenate Strings,
- funkcję konwersji do małych znaków To Lower Case,
- konwersję zmiennych liczbowych do wymaganego formatu łańcuchowego – Format Into String,
- stałą znaku spacji – Space Constant,
- stałą końca wiersza – End of Line Constant,
- stałą łańcuchową – String Constant. Docelowo będzie występowało osiem stałych łańcuchowych o różnej zawartości. Aktualnie możliwe jest zastosowanie czterech z nich z zawartością: „%5.3.f", „Wynik pomiaru", „wynosi", „Czas pomiaru".



8. Zamieszczanie funkcji pobierania czasu systemowego (Programming /Timing oraz Get Date / Time String).

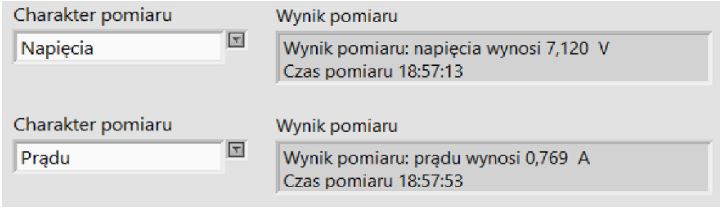
9. Deklarację stałych dla funkcji Get Date / Time String należy przeprowadzić za pomocą menu kontekstowego węzłów numerycznych (menu kontekstowe /Create / Constant /wprowadzenie oczekiwanej wartości).
10. Zamieścić strukturę wyboru (Functions /Programming /Structures /Case Structure). Struktura zawiera cztery ramki zaprezentowane poniżej. Dla działania programu istotne jest, aby zawartość identyfikatora (Selector label) była zgodna z wartościami generowanymi przez Combo Box.



11. Wewnątrz ramek struktury wyboru znajdują się:
- stałe łańcuchowe (Programming /String oraz String Constant) (-, V, A, Ohmów),
- węzły funkcji mnożenia (Programming /Numeric oraz Multiply),
- węzły generowania zmiennej pseudolosowej (Programming /Numeric oraz Random Number (0–1),
- stałe liczbowe zadeklarowane za pomocą menu kontekstowego (menu kontekstowe węzła /Create /Constant /wprowadzenie oczekiwanej wartości).

12. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
13. Zapisać plik programu pod aktualną nazwą (045_Data_Info.vi).

14. **Uruchamianie programu**. Zadeklarować typ prowadzonego pomiaru.



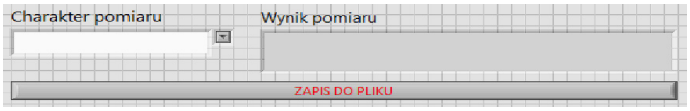
15. Uruchomić program w trybie jednokrotnym. Zaobserwować, czy występuje:
- zgodność generowanych komunikatów z wybraną opcją pomiaru (np. napięcie = napięcie),
- formatowanie wartości liczbowych w łańcuchu zgodnie z podanym formatem,
- zamieszczanie poprawnych informacji o dacie i czasie.

16. Zamknąć plik programu po zakończeniu testowania.

4.3.6. Zapis danych do pliku

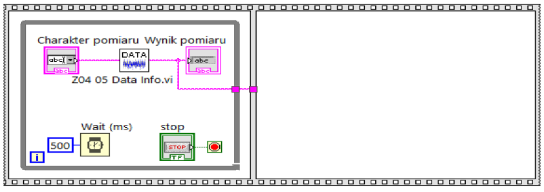
Cel zadania:	zapoznanie z mechanizmem zapisu danych łańcuchowych w pliku oraz typową kolejnością obsługi funkcji plikowych.
Zakres zadania:	opracowywanie programu umożliwiającego zapis w pliku strumienia tekstu z wartościami pomiarowymi wygenerowanymi inną aplikacją. W trakcie ćwiczenia zostanie wykorzystany uprzednio opracowany program (045_Data_Info.vi).

1. **Budowa panelu czołowego**. Uruchomić środowisko LabVIEW z pustym plikiem programu. Utworzyć panel czołowy podobny do zamieszczonego na rysunku. Elementy panelu mogą pochodzić z istniejącego programu (045_Data_Info.vi).



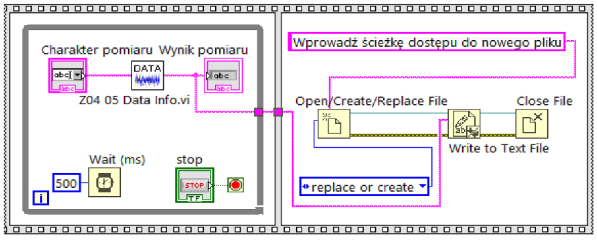
2. Zamieścić przycisk logiczny Stop Button (palety Controls /Modern /Boolean i Stop Button). Za pomocą arkusza właściwości przycisku zmienić przycisk na „Zapis do pliku”.
3. Uporządkować elementy na panelu czołowym. Dopasować rozmiary tak, aby były zbliżone do zamieszczonych na rysunku.
4. Zapisać plik programu pod nazwą (046_Data_zapis.vi) w lokalizacji wskazanej przez prowadzącego.

5. **Schemat blokowy programu**. Przejść do okna schematu blokowego. Aktualnie kod zawiera jedynie trzy elementy stanowiące odpowiedniki obiektów umieszczonych na panelu aplikacji.
6. Zamieścić strukturę sekwencyjną z dwoma ramkami (Functions /Programming /Structures /Flat Sequence Structure). Struktura zawiera dwie ramki. Pierwsza z ramek odpowiedzialna jest za generowanie danych, a druga za zapis danych do pliku.
7. Zamieścić pętlę While (Functions /Programming /Structures /While Loop). Pętla odpowiedzialna jest za generowanie danych o charakterze wybranym przez użytkownika, co 500 ms. Zakończenie działania pętli powoduje automatyczne przejście do drugiej z ramek struktury wyboru.



8. Zamieszczanie funkcji zwłoki czasowej Wait (ms) (Programming /Timing oraz Wait (ms)). Stałą liczbową zadeklarować za pomocą menu kontekstowego (menu kontekstowe węzła wejścia węzła Wait (ms) i dalej /Create /Constant /wprowadzenie oczekiwanej wartości).
9. Zamieszczanie podprogramu (045_Data_Info.vi). Poza przeciągnięciem za pomocą kursora myszy widocznej ikony podprogramu, przeciągnięciem za pomocą kursora myszy pliku programu wyświetlanego w eksploratorze systemu operacyjnego, możliwe jest wykorzystanie polecenia Select a VI (Programming /Select a VI oraz wskazanie zamieszczanego pliku podprogramu).
10. Zamieszczanie funkcji operujących na plikach. Z głównej palety Programming wybrać subpaletę File I/O, a z niej:

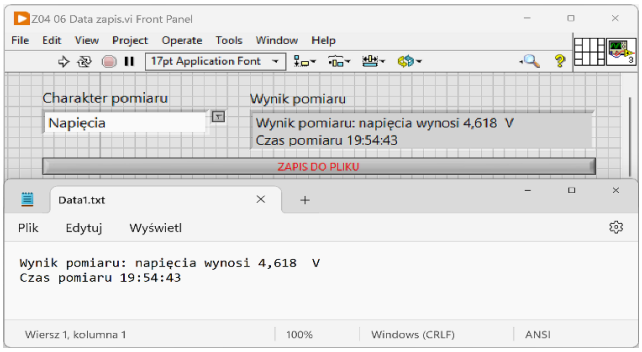
- funkcję otwierania /tworzenia pliku Open /Create /Replace File,
- funkcję zapisu do pliku Write to Text File,
- funkcję zamykania pliku Close File.



11. Deklaracja sposobu funkcjonowania węzła otwierania/tworzenia pliku Open / Create /Replace File. Za pomocą menu kontekstowego wejścia Operation węzła wybrać polecenie /Create /Constant. Z dostępnych opcji wybrać replace or create.
12. Deklaracja komunikatu okna dialogowego węzła Open /Create /Replace File. Za pomocą menu kontekstowego wejścia Prompt węzła wybrać polecenie /Create / Constant. W polu stałej łańcuchowej wprowadzić tekst „Wprowadź ścieżkę dostępu do nowego pliku".
13. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
14. Zapisać plik programu pod aktualną nazwą (046_Data_zapis.vi).

15. **Uruchamianie programu**. Zadeklarować typ prowadzonego pomiaru.
16. Uruchomić program w trybie jednokrotnym.
17. Wybrać jedną z opcji „Charakteru pomiaru" (np. „napięcia").
18. W wybranym przez użytkownika momencie wybrać przycisk „Zapis do pliku".
19. Wskazać miejsce zapisu pliku i nadać plikowi nazwę „Data1.txt".
20. Za pomocą dowolnego edytora tekstu skontrolować zawartość pliku „Data1.txt" z danymi panelu czołowego.

21. Program ponownie uruchomić dla pomiarów „prądu” i „rezystancji”. Pliki zapisywać z kolejnymi liczbami Data2.txt, Data3.txt.



22. Za pomocą edytora tekstu zinstalowanego w systemie opreacyjnhych skontrolować zawartość plików „Data2.txt” i „Data3.txt” z danymi panelu czołowego
23. Zamknąć plik programu po zakończeniu testowania.

4.3.7. Odczyt danych z pliku

Cel zadania: utrwalenie znajomości obsługi danych łańcuchowych magazynowanych w plikach. Poznanie metody obsługi odczytu danych za pomocą funkcji plikowych.

Zakres zadania: opracowywanie programu umożliwiającego odczyt wartości pomiarowych z pliku. W trakcie ćwiczenia zostaną wykorzystane pliki zapisane podczas uprzedniego ćwiczenia (Data1.txt, Data2.txt, Data3.txt) za pomocą programu 046_Data_zapis.vi.

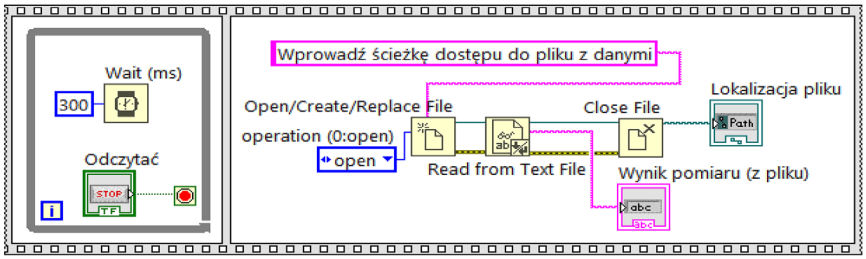
1. **Budowa panelu czołowego**. Uruchomić środowisko LabVIEW z pustym plikiem programu. Utworzyć panel czołowy podobny do zamieszczonego na rysunku. Elementy panelu mogą pochodzić z istniejącego programu (046_Data_zapis.vi).

2. Zamieścić przycisk logiczny Stop Button (palety Controls /Modern /Boolean i Stop Button). Za pomocą arkusza właściwości przycisku zmienić przycisk na „Odczyt pliku – START”.

3. Panel zawiera dodatkowo:
- obiekt zmiennej wyjściowej łańcuchowej (palety Controls /Modern /String & Path i String Indicator),
- obiekt ścieżki dostępu (palety Controls /Modern /String & Path i File Path Indicator).

4. Uporządkować elementy na panelu czołowym. Dopasować rozmiary tak, aby były zbliżone do zamieszczonych na rysunku.
5. Zapisać plik programu pod nazwą (047_Data_odczyt.vi) w lokalizacji wskazanej przez prowadzącego.

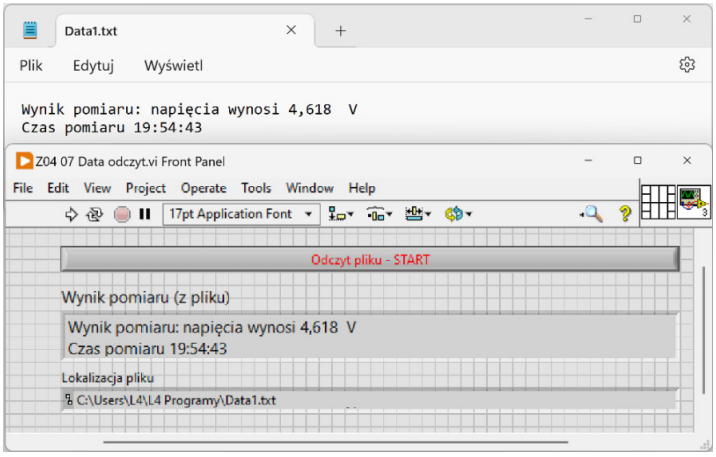
6. **Schemat blokowy programu**. Przejść do okna schematu blokowego. Aktualnie kod zawiera jedynie trzy elementy stanowiące odpowiedniki obiektów umieszczonych na panelu aplikacji.
7. Zamieścić strukturę sekwencyjną z 2 ramkami (Functions /Programming /Structures /Flat Sequence Structure). Pierwsza z ramek odpowiedzialna jest za utworzenie sztucznej struktury oczekiwania na uruchomienie odczytu pliku. Druga z ramek odpowiedzialna będzie za zapis danych do pliku.



8. Zamieścić pętlę While (Functions /Programming /Structures /While Loop). Pętla odpowiedzialna jest za utworzenie sztucznej struktury oczekiwania na uruchomienie odczytu pliku. Zakończenie działania pętli powoduje automatyczne przejście do drugiej z ramek struktury wyboru.
9. Zamieszczanie funkcji zwłoki czasowej Wait (ms) (Programming /Timing oraz Wait (ms)). Stałą liczbową o wartości 300 zadeklarować za pomocą menu kontekstowego (menu kontekstowe węzła wejścia węzła Wait (ms) i dalej /Create / Constant /wprowadzenie oczekiwanej wartości).
10. Zamieszczanie funkcji operujących na plikach. Z głównej palety Programming wybrać subpaletę File I /O, a z niej:
- funkcję otwierania/tworzenia pliku Open /Create /Replace File,
- funkcję odczytu z pliku Read from Text File,
- funkcję zamykania pliku Close File.

11. Deklaracja sposobu funkcjonowania węzła otwierania/tworzenia pliku Open / Create /Replace File. Za pomocą menu kontekstowego wejścia Operation węzła wybrać polecenie /Create /Constant. Z dostępnych opcji wybrać open.
12. Deklaracja komunikatu okna dialogowego węzła Open /Create /Replace File. Za pomocą menu kontekstowego wejścia Prompt węzła wybrać polecenie /Create / Constant. W polu stałej łańcuchowej wprowadzić tekst „Wprowadź ścieżkę dostępu do pliku z danymi".
13. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
14. Zapisać plik programu pod aktualną nazwą (047_Data_odczyt.vi).

15. **Uruchamianie programu.**
16. Za pomocą dowolnego edytora tekstu skontrolować zawartość uprzednio utworzonych plików „Data1.txt", „Data2.txt" i „Data3.txt".
17. Uruchomić program w trybie jednokrotnym. W tle wykonywana jest „pusta" pętla.
18. W dowolnym momencie wybrać przycisk „Odczyt z pliku – START".
19. Wskazać miejsce uprzednio zapisanego pliku „Data1.txt".
20. Jeśli występuje zgodność zawartości pliku skontrolowanego edytorem tekstu i aplikacją 047_Data_odczyt.vi, to podobnie odczytać pliki „Data2.txt" i „Data3.txt".



21. Usunąć pliki „Data1.txt", „Data2.txt" i „Data3.txt" z dysku komputera.
22. Zamknąć plik programu po zakończeniu testowania.

4.4. Pytania kontrolne

1. Co oznacza czarny symbol zmiennej klastrowej?
2. Jaka struktura środowiska LabVIEW zapewnia prostą obsługę błędu?
3. Co w sposobie funkcjonowania struktury sekwencyjnej może stanowić źródło błędów w programie?
4. Jakie jest główne ograniczenie węzła Expression Node?
5. Od czego zależy wybór separatora dziesiętnego w węźle Expression Node?
6. Ile zmiennych wyjściowych węzła Formula Node może pozostać niewykorzystanych w kodzie programu?
7. Jaki problem występuje przy prezentacji danych łańcuchowym zmagazynowanych w postaci tabeli?
8. W jaki sposób należałoby skonfigurować pole tekstowe służące do wpisywania hasła, aby hasło nie było widoczne dla osób postronnych?
9. Jaka jest typowa kolejność postępowania przy obsłudze obiektów magazynujących dane w postaci plików systemu operacyjnego?
10. Jakie typy funkcji i podprogramów plikowych charakteryzowane są w środowisku LabVIEW jako zaawansowane (advanced)?

5. Realizacja aplikacji kontrolno-pomiarowych w środowisku LabVIEW

5.1. Cel zajęć

Rozdział zawiera informacje związane z funkcjonowaniem aplikacji realizujących procesy kontrolno-pomiarowe. Wykonywane zadania umożliwiają poznanie procedur budowania założonej aplikacji przez etapową rozbudowę kolejnych plików VI. W sposób praktyczny przedstawiane są możliwości środowiska LabVIEW w zakresie wykorzystania plików SubVI, struktur programowych, gotowych narządzi i funkcji oraz układów umożliwiających symulowanie sygnałów pomiarowych. W ten sposób przedstawione zostaną praktyczne aspekty funkcjonowania, przedstawionych we wstępie rozdziału, wiadomości dotyczących układów i systemów kontrolno-pomiarowych.

Celem zajęć jest w szczególności:

- utrwalenie umiejętności z zakresu wykorzystania programów SubVI w większych projektach,
- wykorzystanie modułów symulacyjnych do zastępowania rzeczywistych układów kontrolno-pomiarowych na etapie tworzenia aplikacji,
- wykorzystanie struktur programowych do zarządzania pracą aplikacji,
- przygotowanie testów i procedur analitycznych,
- poznanie funkcjonowania aplikacji kontrolno-pomiarowych i ich możliwości operacyjnych,
- poznanie charakterystyki systemów kontrolno-pomiarowych oraz uniwersalności środowiska LabVIEW w tworzeniu takich systemów.

5.2. Wstęp

5.2.1. Charakterystyka systemów kontrolno-pomiarowych

System pomiarowy to zbiór elementów (sprzętowych i programowych) tworzących spójny układ podzespołów przeznaczonych do zrealizowana określonego zadania pomiarowego. W odróżnieniu od układu pomiarowego (samych mierników) system pomiarowy ma wyróżniony element zarządzający procesem pomiarowym (kontroler). Taką jednostką może być odpowiednio zaprogramowany układ mikrokontrolera lub komputer. Rolą jednostki zarządzającej jest sterowanie urządzeniami stanowiącymi system pomiarowy, akwizycja danych pomiarowych oraz ich przetwarzanie w celu otrzymania odpowiednio obrobionej informacji o pomiarze. Środowisko LabVIEW umożliwia budowę spinającej wszystkie procesy pomiarowe aplikacji [17, 19].

Aplikacje zarządzające procesem kontrolno-pomiarowym mogą działać na wielu płaszczyznach i realizować różne procesy, w tym:

- przeprowadzać testy funkcjonowania, sprawdzania ustawień i kalibracji czujników,

- symulować realizowane procesy i wspomagać proces decyzyjny i nadzoru,
- prowadzić procesy akwizycji i archiwizacji pomiarów i zdarzeń,
- przetwarzać i analizować otrzymane dane,
- dokonywać oceny realizacji zadań kontrolno-pomiarowych.

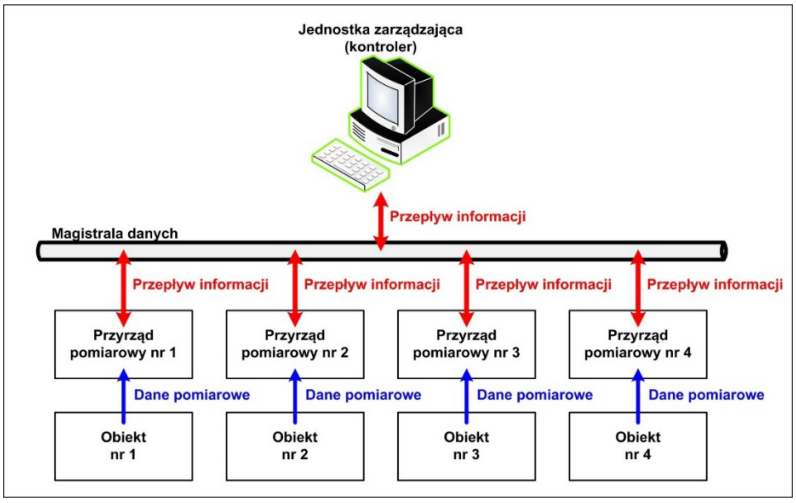
Systemy pomiarowe ze względu na zastosowanie dzielą się na [17]:

- kontrolno-pomiarowe – służą do oceny procesu – wynik pomiaru musi odpowiadać założonym wartościom,
- diagnostyczno-pomiarowe – służą do wykrycia uszkodzeń i nieprawidłowości – wynik pomiaru jest porównywany z wartościami oczekiwanymi,
- badawcze – służą do celów pomiaru i oceny eksperymentu – wynik pomiaru bardzo często jest przypuszczalny, ewentualnie wstępnie oszacowany na podstawie obliczeń symulacyjnych.

W zależności od zastosowania, systemy pomiarowe mogą mieć rozbudowane układy akwizycji danych, procedury obliczeniowe lub mieć charakter oprogramowania otwartego.

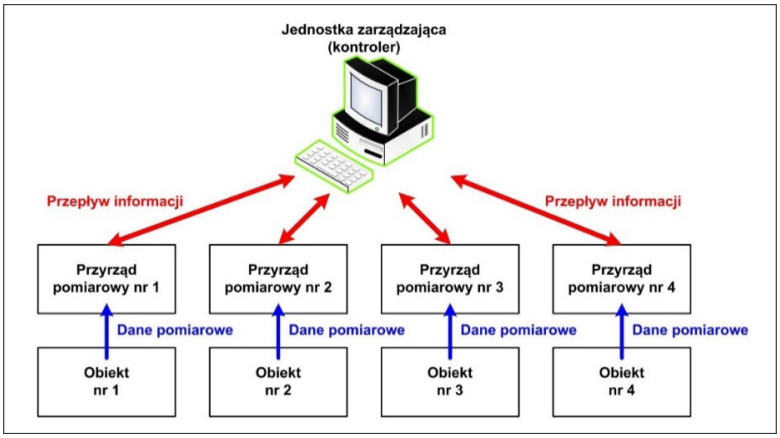
Systemy kontrolno-pomiarowe, ze względu na konfigurację, dzieli się na:

- magistralowe zwane również liniowymi (rys. 5.1),
- gwiazdowe (rys. 5.2),
- posobne zwane również pętlowymi (rys. 5.3).



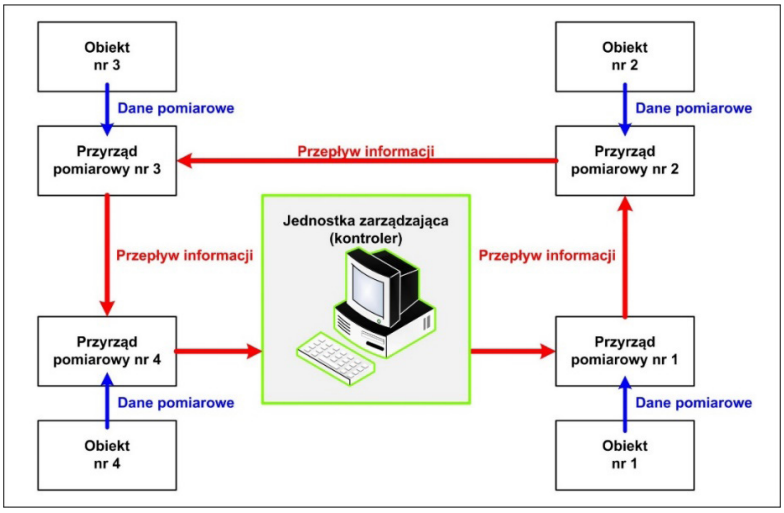
Rys. 5.1. Magistralowy system kontrolno-pomiarowy

Systemy o topologii magistralowej mają zastosowanie szczególnie w układach, w których wymiana informacji może odbywać się bilateralnie między poszczególnymi elementami systemu. Systemy te charakteryzuje możliwość szybkiej rozbudowy z zastosowaniem istniejącej infrastruktury i okablowania. Jednostka zarządzająca stanowi zazwyczaj tylko układ nadzoru, umożliwia kontrolę nad realizowanymi procesami.



Rys. 5.2. Gwiazdowy system kontrolno-pomiarowy

Systemy o topologii gwiazdowej cechuje priorytetowy charakter jednostki zarządzającej oraz duże zasoby sprzętowe. Jednostka zarządzająca nadzoruje wymianę danych pomiędzy elementami systemu i tylko w niej realizowany jest proces analizy danych. Pod względem rozbudowy, każdorazowo wymaga prowadzenia dodatkowego okablowania do kolejnych układów.



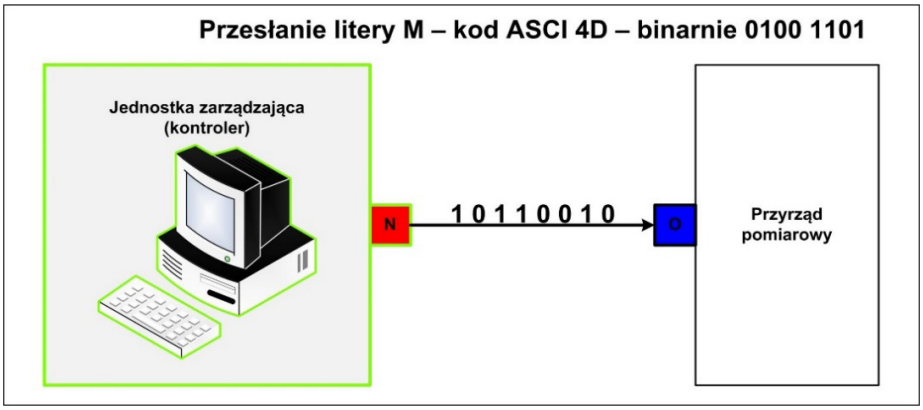
Rys. 5.3. Posobny system kontrolno-pomiarowy

Systemy o topologii posobnej (pętlowej) to specyficzne systemy, które łączą dużą operacyjność poszczególnych komórek i jednokierunkową wymianę tylko niezbędnych informacji między elementami systemu. Systemy te wykorzystują simplexową komunikację danych. Zadanie jednostki zarządzającej ogranicza się w tym przypadku

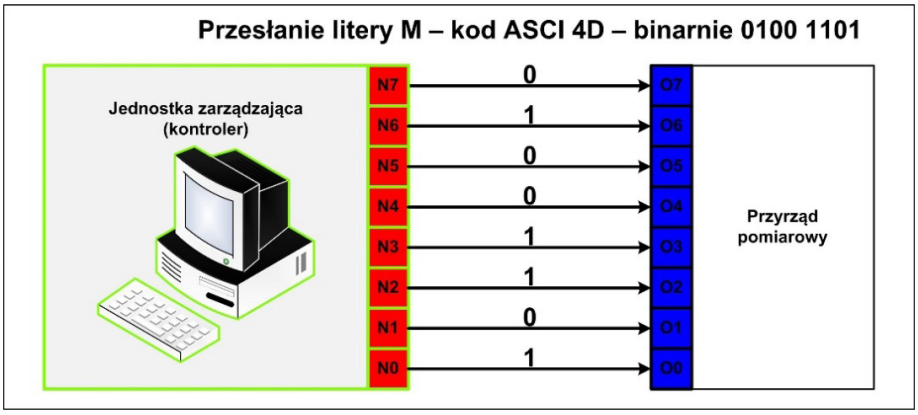
do kontroli procesu oraz zarządzania danymi konfiguracyjnymi. Często taka topologia może być wykorzystywana w systemach zintegrowanych na poziomie transmisji informacji.

Transmisja danych między elementami systemu kontrolno-pomiarowego może odbywać się w następujący sposób:

- szeregowy – szeregowa transmisja danych (rys. 5.4) – jednoliniowa, według zasady kolejności bitów – transmisja wolniejsza, ale zapewniająca mniejsze ryzyko wystąpienia zakłócenia sygnału oraz przesył danych na większe odległości,
- równoległy – równoległa transmisja danych (rys. 5.5) – wieloliniowa – według zasady kolejności słów (wielobitowych) – transmisja szybsza, ale wiążąca się z większym ryzykiem zakłócenia sygnału, jest to transmisja danych na krótkie odległości.



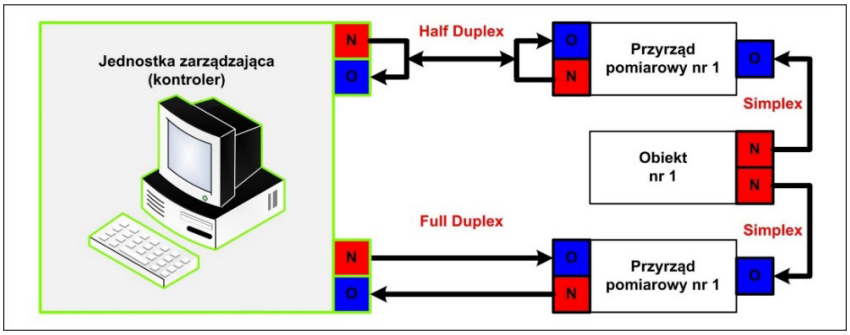
Rys. 5.4. Szeregowa transmisja danych w systemach kontrolno-pomiarowych



Rys. 5.5. Równoległa transmisja danych w systemach kontrolno-pomiarowych

Transmisja danych między elementami systemu kontrolno-pomiarowego może odbywać się w następujących trybach (rys. 5.6):

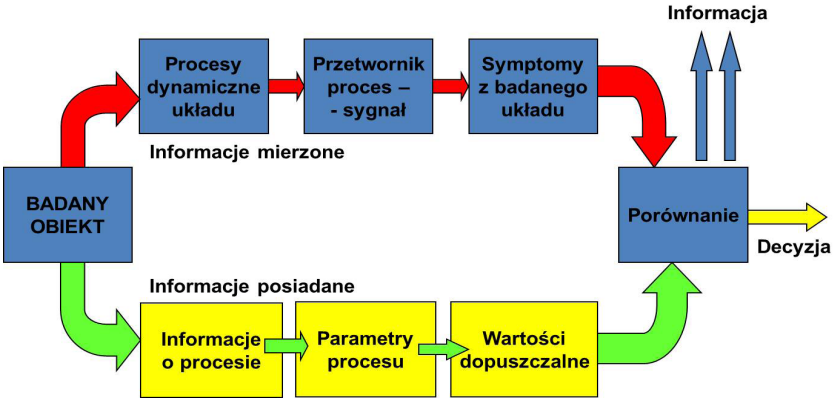
- Simplex – jednokierunkowa transmisja danych – ma zastosowanie w kanałach pomiarowych czujników – układy rozsiewcze typu broadcast,
- Half Duplex – dwukierunkowa naprzemienna transmisja danych – ma zastosowanie w zamkniętych układach korzystających ze standardów RS-485, Modbus RTU,
- Full Duplex – dwukierunkowa niezależna transmisja danych – ma zastosowanie w układach korzystających z sieci Ethernet i standardów TCP/IP.



Rys. 5.6. Przykładowa transmisja danych w systemie kontrolno-pomiarowym z wykorzystaniem trybu: Simplex, Half Duplex i Full Duplex

5.2.2. Funkcjonowanie systemów kontrolno-pomiarowych

Przez systemy kontrolno-pomiarowe (rys. 5.7) należy rozumieć systemy, które – na podstawie informacji o procesie oraz informacji pomiarowych z czujników – umożliwiają, przez układy regulacyjne, wpływanie na sposób działania obiektu [8, 17, 20].

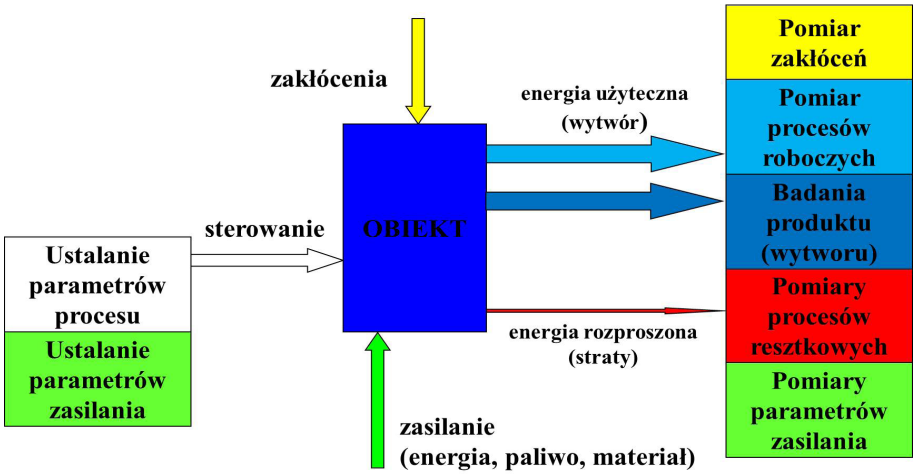


Rys. 5.7. Schemat ideowy powiązań wewnętrznych w systemie kontrolno-pomiarowym z uwzględnieniem modułów diagnostycznych i symulacyjnych

Funkcjonowanie systemów kontrolno-pomiarowych związane jest z realizacją następujących etapów działania:

- określenie realizowanego celu,
- określenie modelu matematyczno-fizycznego obiektu i przebiegu procesu,
- ustalanie bieżących parametrów procesu i parametrów zasilania,
- rejestracja pomiarów,
- proces porównania i przetwarzania danych,
- procesy regulacyjne.

Schemat odziaływania poszczególnych czynników wpływających na obiekt oraz czynniki umożliwiające określenie przebiegu realizowanego procesu [8, 18] zaprezentowano na rysunku 5.8.



Rys. 5.8. Funkcjonowanie rzeczywistych układów kontrolno-pomiarowych – schemat oddziaływania czynników wpływających na przebieg procesu roboczego

Ze względu na możliwości analityczne systemów kontrolno-pomiarowych systemy te mogą realizować następujące cele:

- określanie na podstawie mierzonych i zarchiwizowanych danych aktualnych informacji o procesie,
- określanie na podstawie mierzonych i zarchiwizowanych danych aktualnych danych o procesie oraz przedstawianie stanów wcześniejszych,
- określanie na podstawie mierzonych i zarchiwizowanych danych aktualnych danych o procesie, przedstawianie stanów wcześniejszych oraz prognozowanie stanów przyszłych.

Realizacja przedstawionych zadań wiąże się z realizacją funkcji genezowania, diagnozowania oraz prognozowania.

Głównym celem systemów kontrolno-pomiarowych jest nadzór nad prawidłowym przebiegiem procesu roboczego związanego z otrzymywaniem założonego produktu. W działaniach kontrolno-pomiarowych rozróżnia się następujące fazy badania oceno-wego:

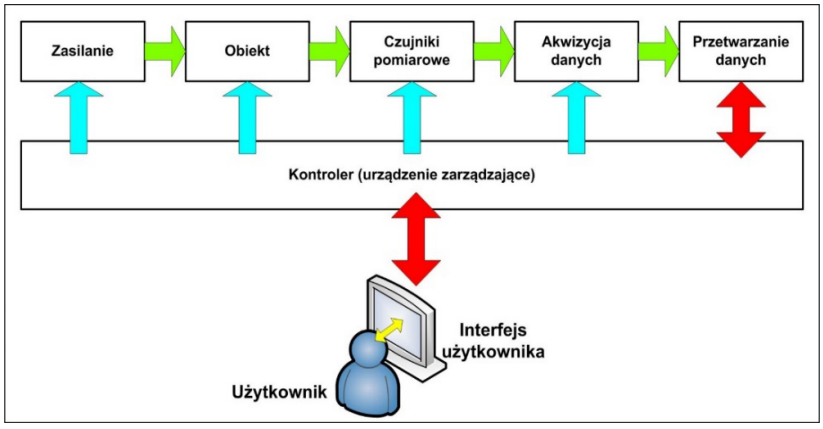
- kontrolę stanu obiektu,
- ocenę aktualnego stanu i konsekwencje tego stanu dla produktu końcowego,
- lokalizację i separację uszkodzeń powstałych w obiekcie,
- wnioskowanie o przyszłych stanach obiektu.

Czynności związane z funkcjonowaniem układów kontrolno-pomiarowych można podzielić na funkcje:

- monitorowania,
- nadzór,
- zabezpieczenie.

W systemach kontrolno-pomiarowych można rozróżnić systemy powiązane z mo-nitorowaniem i nadzorem nad procesami związanymi z przebiegiem procesu oraz ze stanem obiektu.

Struktura systemów kontrolno-pomiarowych określona jest oddziaływaniem po-szczególnych czynników wpływających na proces roboczy oraz danymi, jakie moż-na uzyskać od elementów procesu roboczego. Schemat przebiegu procesu uzyskiwa-nia informacji oraz rola kontrolera procesu (sterownika, urządzenia zarządzającego lub komputera), a także możliwości regulacyjnych systemu kontrolno-pomiarowego zostały zaprezentowane na rysunku 5.9. Przedstawiona rola użytkownika to zarządza-nie. Dane, jakie otrzymuje użytkownik, mogą być związane z jego uprawieniami sys-temowymi. W kontrolerze można przypisać poszczególnym grupom użytkowników odpowiednie uprawnienia, dające możliwości wpływania na proces kontroli, konfigu-racji lub nadzoru [3, 6].



Rys. 5.9. Struktura systemu kontrolno-pomiarowego

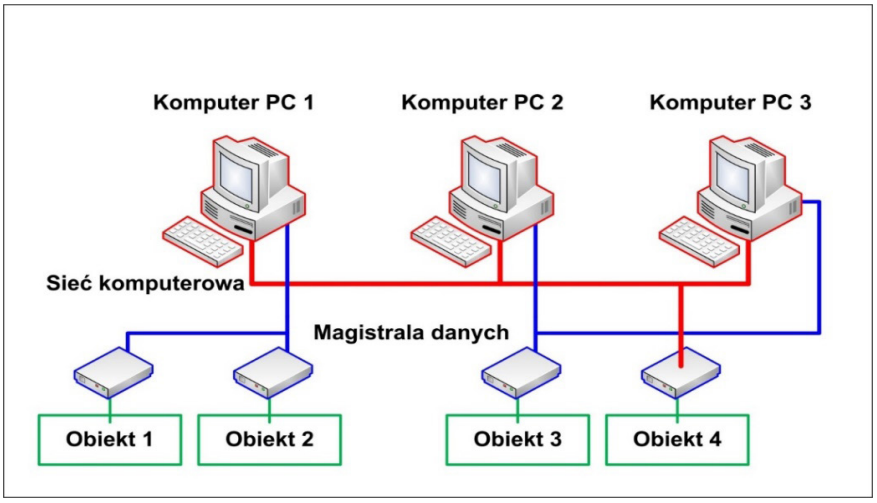
Ze względu na sposób użycia jednostki zarządzającej (komputera) systemy kontrolno-pomiarowe można podzielić na następujące grupy:

- Dozorowe systemy kontrolno-pomiarowe z ręcznym sterowaniem.
  Komputer służy tylko do zbierania i późniejszego przetwarzania danych z układów pomiarowych, proces kontroli ograniczony jest do sterowania ręcznego.
- Komputerowe systemy kontrolno-pomiarowe.
  Komputer służy do zarządzania procesem sterowania, kontroli oraz do zbierania i późniejszego przetwarzania danych z układów pomiarowych, jednak proces kontrolno-pomiarowy ograniczony jest do dostarczonego oprogramowania.
- Programowalne systemy kontrolno-pomiarowe.
  Komputer służy do zarządzania procesem sterowania, kontroli oraz do zbierania i późniejszego przetwarzania danych z układów pomiarowych, a użytkownik dzięki oprogramowaniu może swobodnie dostosować interfejs i zakres prowadzonych badań (pomiarów) i symulacji.

O konfiguracjach systemu komputerowego z układami pomiarowymi mogą decydować następujące czynniki:

- Wykorzystanie standardowych portów jednostki zarządzającej lokalnie. Wymiana danych przez porty RS-232 między dwoma komputerami. Nadzór nad przebiegiem procesów transmisji danych.
- Wykorzystanie kart pomiarowych zainstalowanych w komputerze. Aplikacja (oprogramowanie) jest zainstalowana na komputerze. Jednostka zarządzająca (komputer) obsługuje procesy sterowania i pomiaru przez urządzenie podłączone do płyty głównej komputera lub bezpośrednio przez porty/interfejsy komunikacyjne, będące na wyposażeniu komputera.
- Wykorzystanie zewnętrznych kart pomiarowych podłączonych do jednostki zarządzającej (komputera) za pomocą gniazda RS, USB, WiFi. Aplikacja (oprogramowanie) jest zainstalowana na komputerze. Jednostka zarządzająca (komputer) obsługuje procesy sterowania i pomiaru przez urządzenia zewnętrzne przyłączone do komputera.
- Wykorzystanie rozbudowanych i rozległych układów pomiarowych zarządzanych w sposób zdalny. Aplikacja jest zainstalowana na komputerze. Jednostka zarządzająca (komputer) obsługuje procesy sterowania i pomiaru przez urządzenia zewnętrzne przyłączone do sieci komputerowej.

Poszczególne konfiguracje systemów zarządzających z systemami pomiarowymi przedstawiono na rysunku 5.10.



Rys. 5.10. Powiązania między systemami zarządzającymi a układami pomiarowymi

5.2.3. Proces tworzenia aplikacji do systemów kontrolno-pomiarowych

Proces tworzenia aplikacji zarządzającej systemem kontrolno-pomiarowym związany jest z wieloma etapami, główne z nich to:

- wybór platformy programowej,
- wybór osprzętu pomiarowego,
- określenie sposobu funkcjonowania programu,
- określenie sposobu komunikacji wewnętrznej i zewnętrznej,
- określenie struktury systemu pomiarowego (zwarta czy rozproszona),
- wykonanie projektu/schematu funkcjonowania całego systemu pomiarowego,
- zdefiniowanie węzłów programu,
- budowa szkieletu aplikacji,
- określenie algorytmów przepływu danych,
- wybór narzędzi i struktur programowych,
- tworzenie podprogramów,
- implementacja podprogramów w aplikacji głównej,
- testowanie programu,
- wdrożenie poprawek i gotowego produktu.

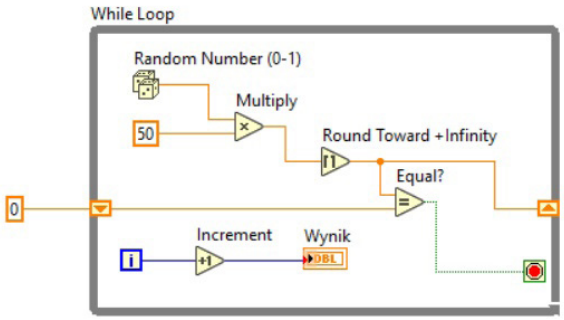
Ważnym, ze względu na realizowane przez system kontrolno-pomiarowy cele, jest etap wyboru odpowiedniej, umożliwiającej spełnienie wymagań, architektury programu zarządzającego. Architektura programu, czyli modele szkieletu programu, związana jest z funkcjonowaniem systemu pomiarowego oraz zadaniami, jakie on realizuje.

Typowe architektury programu to:

- prosta,
- podstawowa (architektura prosta ujęta w pętlę pozwalającą na nieprzerwane wykonywanie instrukcji kodu),

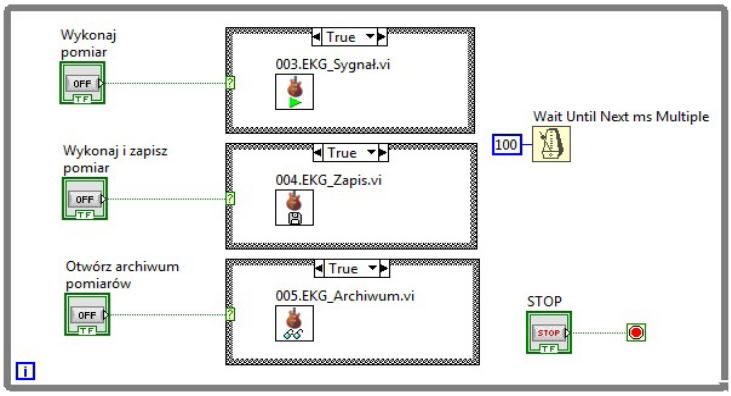
- wielokrotnych struktur wyboru,
- pętli równoległych,
- maszyny stanu.

Architektura prosta programu – program nie zawiera skomplikowanych struktur programistycznych, jego działanie polega na realizowaniu prostego kodu zawartego w oknie schematu blokowego. Zamknięcie kodu w pętli powoduje zmodyfikowanie architektury do postaci architektury podstawowej (rys. 5.11).



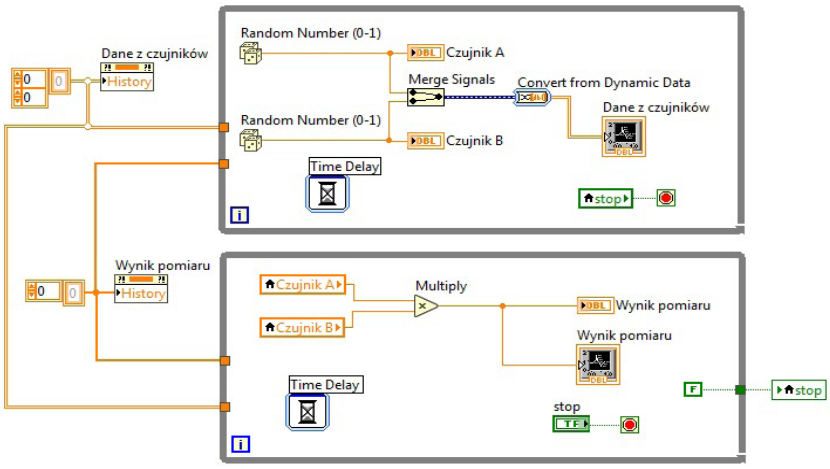
Rys. 5. 11. Przykład aplikacji o architekturze ogólnej

Architektura wielokrotnych struktur wyboru (rys. 5.12) związana jest z wykorzystaniem odpowiednich ustawień programu VI Properties. Program główny działa w pętli While, a w strukturach warunku (Case) umieszczone są podprogramy, które mogą być wywołane.



Rys. 5.12. Architektura wielokrotnych struktur wyboru

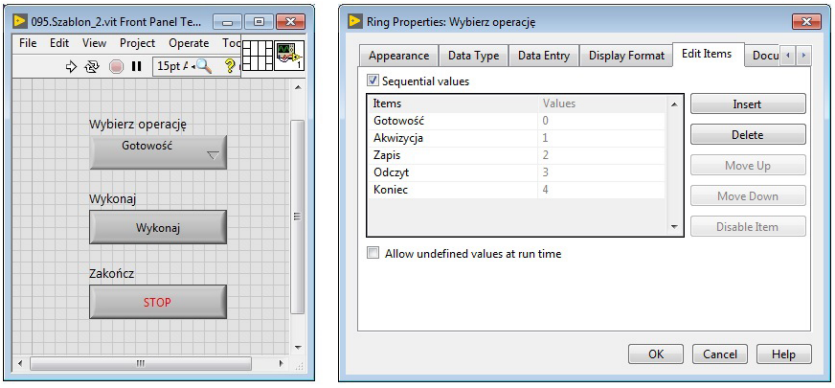
Architektura pętli równoległych (rys. 5.13) wykorzystywana jest w systemach pomiarowych, w których istotna jest kwestia ciągłości pomiarów, a architektura prosta (przez zastosowanie sekwencji – pomiar – analiza – zapis) nie daje takiej możliwości.



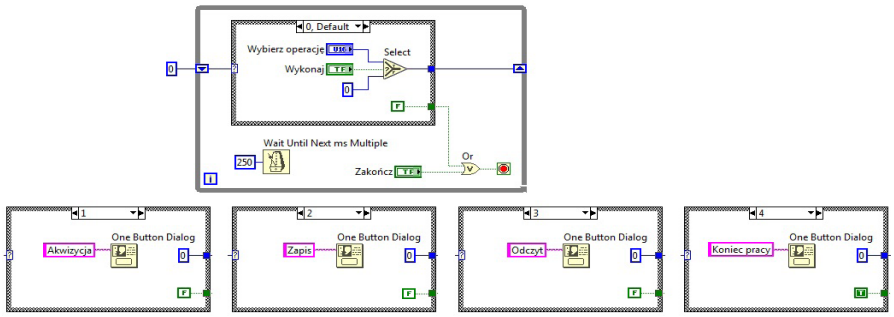
Rys. 5.13. Architektura pętli równoległych

Architektura maszyny stanu to określenie funkcjonowania programu przez zdefiniowanie szeregu stanów, jakie są wykonywane w kolejnych iteracjach. Pokazuje możliwe stany obiektu, od stanu początkowego do stanu końcowego oraz przejścia, które powodują zmianę tego stanu.

Na rysunku 5.14 przedstawiono interfejs programu oraz okno dialogowe służące do generowania przypadków. Na rysunku 5.15 jest schemat blokowy, prezentujący funkcjonowanie programów z wykorzystaniem architektury maszyny stanu z opcją wywoływania operacji ze zdefiniowanej listy. Sterowanie odbywa się przez wybór operacji oraz potwierdzenie wykonania. Dzięki temu program dostaje informację, który zdefiniowany przypadek struktury wyboru ma być wybrany w kolejnej iteracji programu. Podczas wykonywania zawartych w oknie danego przypadku instrukcji inne opcje programu nie są aktywne.



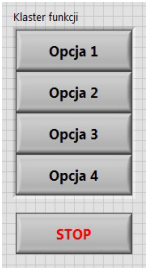
Rys. 5.14. Interfejs z listą stanów oraz okno dialogowe do tworzenia stanów programu



Rys. 5.15. Przykładowa aplikacja oparta o architekturę maszyny stanu

W przypadku wykorzystania architektury maszyny stanu ze sterowaniem za pomocą klastra nie jest wybierany przypadek ze zdefiniowanej listy tylko analizowane są stany zdefiniowanych przycisków.

Na rysunku 5.16 przedstawiono interfejs programu z umieszczonym klastrem sterującym. Na rysunku 5.15 znajduje się schemat blokowy prezentujący funkcjonowanie programów z wykorzystaniem architektury maszyny stanu sterowanego klastrem funkcyjnym.



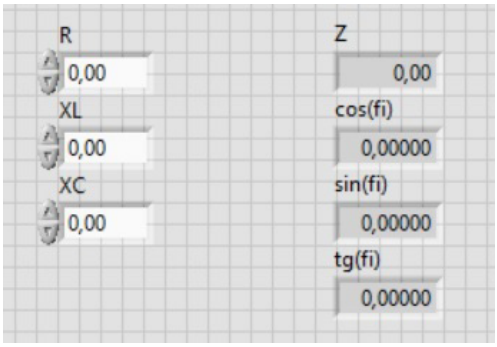
Rys. 5.16. Interfejs z klastrem funkcyjnym

5.3. Zadania

5.3.1. Impedancja – opracowanie procedury obliczeniowej symulatora

Cel zadania:	utworzenie kompletnego przyrządu wirtualnego, służącego do wyznaczania impedancji Z oraz współczynników cosφ, sinφ oraz tgφ obwodu, na podstawie danych o rezystancji R, reaktancji XL i reaktancji XC w szeregowej gałęzi RLC obwodu elektrycznego.
Zakres zadania:	zadania: zastosowanie podstawowych elementów i modułów obliczeniowych do przygotowania procedury pomiarowej. Dla programu zostaną utworzone ikona identyfikacyjna oraz terminal wejść i wyjść.

1. **Panel czołowy programu.** Uruchomić program LabVIEW i otworzyć nowy plik programu w celu utworzenia panelu czołowego o wyglądzie zbliżonym do zaprezentowanego na rysunku.



2. Po wyświetleniu nowego czystego arkusza programu, należy z menu Window wybrać polecenie Tile Left and Right. Daje ono możliwość jednoczesnego wyświetlenia obok siebie dwóch podstawowych okien projektu (okna panelu czołowego i okna schematu blokowego). Umożliwia to sprawne przemieszczanie się między tymi oknami.
3. Utworzyć na panelu czołowym trzy kontrolki numeryczne. W tym celu należy wybrać trzykrotnie z palety Controls obiekt Numeric Control, znajdujący się w grupie Numeric Controls. Wygenerowanym kontrolkom numerycznym nadać nazwy „R", „XL" i „XC" (jeśli etykieta obiektu jest niewidoczna, należy wybrać Properties z menu kontekstowego i wpisać nazwę obiektu w pole Label, zaznaczając opcję Visible).
4. W podobny sposób umieścić na panelu czołowym cztery wskaźniki numeryczne, znajdujące się w palecie Controls, w grupie Numeric Indicators, pod nazwą Numeric Indicator. Wstawionym wskaźnikom numerycznym nadać nazwy „Z", „cos(fi)", „sin(fi)" oraz „tg(fi)".

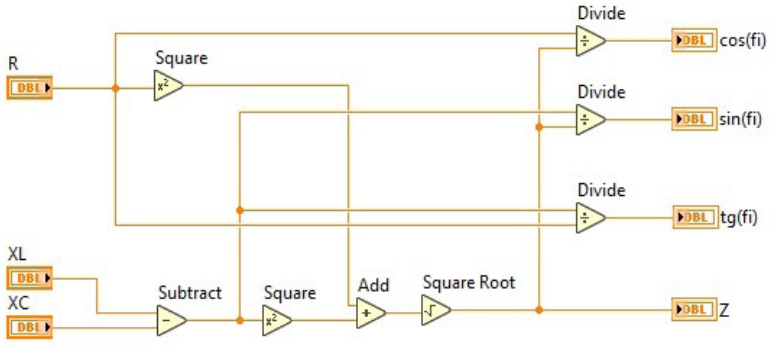
5. **Schemat blokowy programu**. Przejść do okna schematu blokowego i uaktywnić paletę Function. Następnie uaktywnić grupę obiektów Express Numeric znajdującą się w zakładce Express → Arithmetic & Comparison.
6. Umieścić w obszarze okna schematu blokowego następujące obiekty:

- trzy moduły dzielenia (Divide),
- moduł odejmowania (Subtract),
- moduł dodawania (Add),
- dwa moduły podnoszenia do kwadratu (Square),
- moduł pierwiastka kwadratowego (Square root).

7. Po wstawieniu wymaganych obiektów należy przystąpić do tworzenia połączeń. Połączenia mają być tak wykonane, aby umożliwiały realizację następujących funkcji:

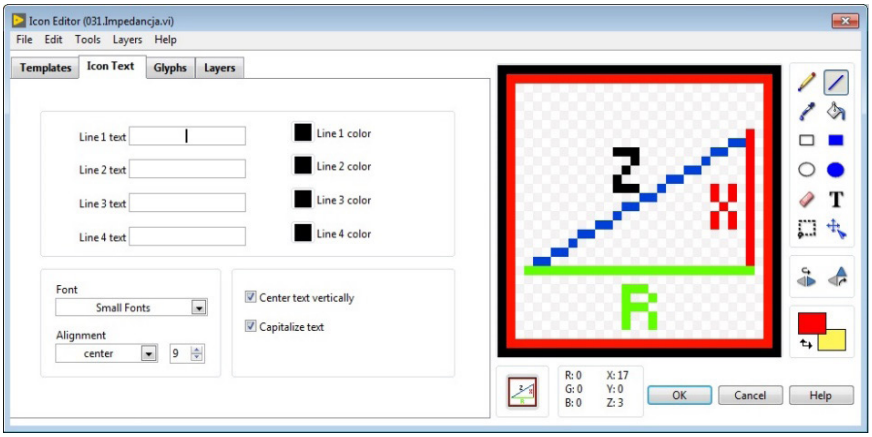
$$Z = \sqrt{R^2 + (XL - XC)^2},$$
$$\cos(fi) = \frac{R}{Z},$$
$$\sin(fi) = \frac{XL - XC}{Z},$$
$$\mathrm{tg}(fi) = \frac{XL - XC}{R}.$$

8. Po wykonaniu powyższych instrukcji dokonać porządkowania obiektów w oknie schematu blokowego tak, żeby wszystkie połączenia były widoczne i wykonane w przejrzysty sposób. Na rysunku przedstawione zostało okno schematu blokowego po uporządkowaniu znajdujących się na nim obiektów i poprowadzeniu niezbędnych połączeń.

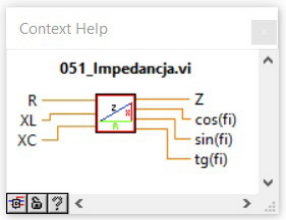


9. **Uruchomienie programu**. Uaktywnić okno panelu czołowego i wprowadzić liczby w pola kontrolek/regulatorów liczbowych.
10. W celu sprawdzenia poprawności działania stworzonego projektu, uruchomić go, wykorzystując funkcję Run Continously (działanie ciągłe).
11. Zapisać zbudowaną aplikację na dysku pod nazwą 051.Impedancja.vi.

12. **Tworzenie ikony** dla przyrządu wirtualnego. W celu zmiany standardowej ikony w ikonę charakterystyczną dla danego przyrządu wirtualnego należy uruchomić edytor ikon (Icon Editor). Do edycji projektowanego obszaru ikony służą narzędzia znajdujące się w belce po prawej stronie okna edytora. Za ich pomocą można stworzyć dowolny obraz ikony. Po zakończeniu edycji zmianę wyglądu ikony zatwierdza się, wciskając przycisk [OK]. Rezultat przeprowadzonych czynności kończy się zdefiniowaniem ikonki (np. podobnej do przedstawionej na rysunku).



13. **Tworzenie terminali wejścia i wyjścia** dla przyrządu wirtualnego. W celu dopasowania szablonu do potrzeb tworzonej aplikacji, należy wybrać odpowiadający potrzebom użytkownika wzorzec (funkcja Patterns w menu skrótów edytora połączeń). W przypadku bieżącego programu należy użyć wzorca z trzema wejściami i czterema wyjściami. Jeżeli tło szablonu nie jest białe, oznacza to, że wcześniej zostały już przypisane typy sygnałów wejściowych lub wyjściowych. Przed przystąpieniem do tworzenia nowego układu połączeń wejścia i wyjścia dla przyrządu wirtualnego należy w menu kontekstowym wybrać funkcję Disconnect All Terminals.
14. W tworzonym przyrządzie wirtualnym obiektami wejściowymi są kontrolki R, XL i XC (kontrolki numeryczne), a obiektami wyjściowymi wskaźniki Z, cos(fi), sin(fi) oaz tg(fi) (wskaźniki numeryczne). Aby dokonać stosownych połączeń i zdefiniować terminale I/O dla tego programu, należy w palecie narzędzi uruchomić funkcję tworzenia połączeń Connect Wire i przesunąć kursor myszki nad obszar obiektu R, a potem wcisnąć lewy przycisk myszki (obiekt powinien zostać zaznaczony obwiednią), następnie skierować kursor nad lewą część kwadratu i ponownie kliknąć lewy przycisk. Kolor pola powinien się zmienić z białego na kolor sygnału przypisanego danemu rodzajowi obiektu, z którym dokonane zostało połączenia (w tym przypadku na pomarańczowy). Analogicznie należy wykonać połączenia między pozostałymi elementami, przypisując je do konkretnych terminali.
15. Po utworzeniu panelu terminali i przejściu do okna panelu czołowego, wyświetlić okno pomocy kontekstowej (Help /Show Context Help lub CTRL + H). W trakcie obserwowania okna pomocy kontekstowej, umieścić wskaźnik myszy nad obszarem konektora/ikony przyrządu wirtualnego. Informacja prezentowana w oknie Context Help powinna być analogiczna do przedstawionej na rysunku.



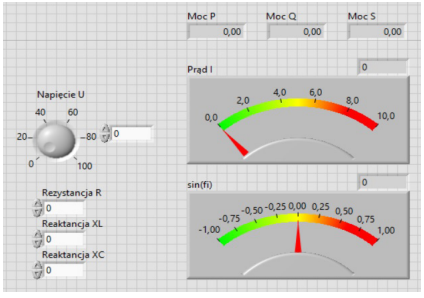
16. Zapisać dokonane zmiany w programie pod istniejącą nazwą (051_Impedancja.vi). Zamknąć plik programu.

5.3.2. Moce – opracowanie symulatora układu do pomiaru wybranych wielkości elektrycznych

Cel zadania: budowa przyrządu wirtualnego umożliwiającego symulowanie pomiarów prądu I, mocy czynnej P, mocy biernej Q i mocy pozornej S oraz wyznaczania współczynnika mocy sinφ. Jako moduł budowanego programu będzie wykorzystany wcześniej zbudowany program (program 051_Impedancja.vi).

Zakres zadania: zastosowanie podstawowych elementów i modułów obliczeniowych do przygotowania procedury pomiarowej. Wykorzystanie przygotowanych wcześniej programów jako elementów programu głównego.

1. **Panel czołowy programu.** Uruchomić program LabVIEW i otworzyć nowy plik programu.
2. Po wyświetleniu nowego czystego arkusza programu należy z menu Window wybrać polecenie Tile Left and Right. Daje ono możliwość jednoczesnego wyświetlenia obok siebie dwóch podstawowych okien projektu (okna panelu czołowego i okna schematu blokowego). Pozwala to na sprawne przemieszczanie się między tymi oknami.
3. Korzystając z obiektów dostępnych w palecie Controls, utworzyć panel czołowy o wyglądzie zbliżonym do zaprezentowanego na rysunku.



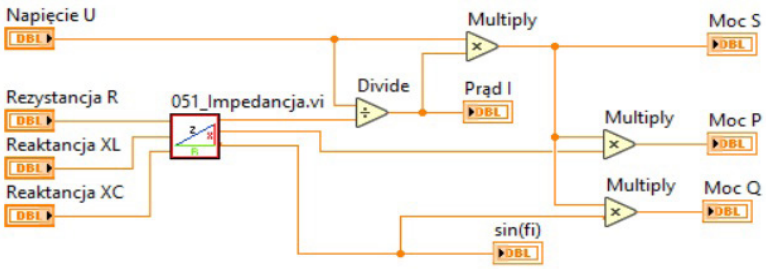
4. W tym celu umieścić następujące kontrolki zdefinjowane w palecie Controls w zakładce Modern → Numeric:
- kontrolkę numeryczną typu Knob (gałka), nadać jej nazwę „Napięcie U”,
- trzy kontrolki numeryczne typu Numeric Control, nadać im nazwy: „Rezystancja R”, „Reaktancja XL” oraz „Reaktancja XC”.

5. Do kontrolki „Napięcie U” dodać wyświetlacz cyfrowy. W celu dodania wyświetlacza cyfrowego należy rozwinąć menu kontekstowe tego obiektu i wybrać polecenie Digital Display, występujące w menu Visible Items.
6. Na front panelu programu umieścić następujące wskaźniki występujące w palecie Controls w zakładce Modern → Numeric:
- trzy wskaźniki numeryczne typu Numeric Indicator, nadać im nazwy: „Moc P”, „Moc Q” i „Moc S”,
- dwa wskaźniki numeryczne typu Meter, nadać i nazwy: „Prąd I” i „sin(fi)”.

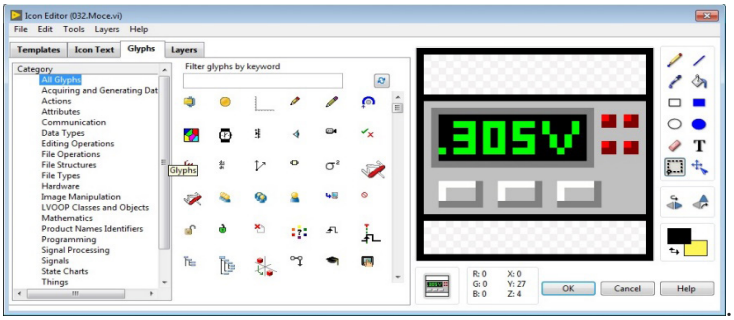
7. Do wskaźników „Prąd I” oraz „sin(fi)” dodać wyświetlacz cyfrowy. W celu dodania wyświetlacza cyfrowego należy rozwinąć menu kontekstowe tego obiektu i wybrać polecenie Digital Display znajdujące się w menu Visible Items.

8. **Schemat blokowy programu**. Uaktywnić okno schematu blokowego i umieścić w nim następujące obiekty występujące w palecie Functions w zakładce Programming → Numeric:
- moduł dzielenia (Divide),
- cztery moduły mnożenia (Multiply).

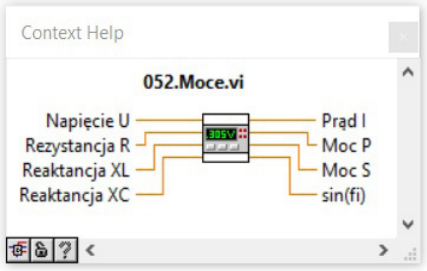
9. Następnie w palecie Function należy wejść do zakładki „Select a VI…”. W uaktywnionym oknie Select the VI to Open należy znaleźć następujący plik 051_Impedancja.vi. Jest to plik z gotowym programem do wyznaczania parametrów elektrycznych Z i współczynników mocy elementu RLC wykonany w poprzednim ćwiczeniu. Prawidłowe przeprowadzenie tego polecenia skutkuje wstawieniem obiektu 051_Impedancja.vi do okna schematu blokowego.
10. Wykonać ścieżki połączeń tak, by powstały schemat połączeń i rozmieszczenie elementów było analogiczne jak na rysunku.



11. **Tworzenie ikony i deklarowanie terminali we/wy**
    Uaktywnić okno panelu czołowego, wskazać kursorem myszy na ikonę projektu (prawy górny róg okna) i wybrać z menu funkcję Edit Icon. Następnie, za pomocą dostępnych w uaktywnionym oknie narzędzi, opracować ikonę, która będzie charakteryzowała budowany przyrząd. Okno edytora ikon oraz obraz przykładowej ikony przedstawiono na rysunku.



12. Następnie wykonać układ konektorów terminali wejść i wyjść dla projektowanego przyrządu wirtualnego. W tworzonym przyrządzie wirtualnym obiektami wejściowymi są kontrolki „Napięcie U", „Rezystancja R", „Reaktancja XL" i „Reaktancja XC" (kontrolki numeryczne), a obiektami wyjściowymi wskaźniki „Prąd I", „Moc P", „Moc S" oaz „sin(fi)".
13. Po wykonaniu czynności związanych z przygotowaniem ikony programu oraz terminali wejścia i wyjścia, informacja prezentowana w oknie Context Help dla tego programu powinna być analogiczna do przedstawionej na rysunku.



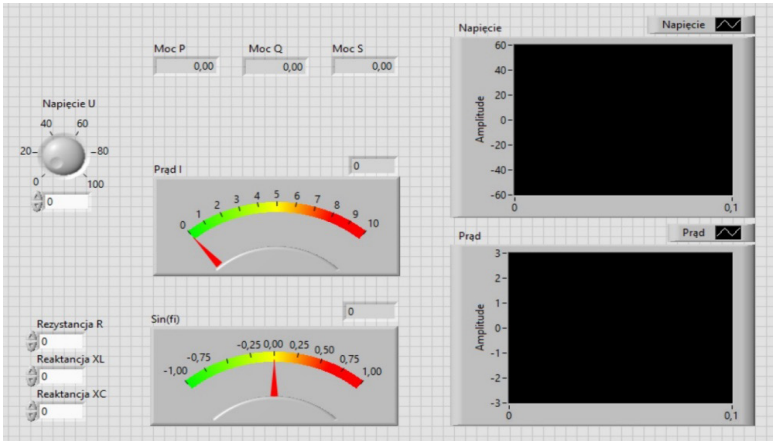
14. **Uruchomienie aplikacji.** Uaktywnić okno panelu czołowego.
15. Wprowadzić liczby w pola kontrolek/regulatorów liczbowych. W celu sprawdzenia poprawności działania stworzonego projektu uruchomić go, wykorzystując funkcję Run Continuously (ciągłe działanie).
16. Obserwować informacje prezentowane na poszczególnych obiektach front panelu.
17. Dokonać zapisu programu na dysku pod nazwą 052_Moce.vi. Zamknąć plik programu.

5.3.3. Moce – modyfikacja symulatora układu do pomiaru wybranych wielkości elektrycznych

Cel zadania: rozbudowa przyrządu wirtualnego o nowe elementy umożliwiające wygenerowanie przebiegów czasowych na podstawie zadanych parametrów obwodu elektrycznego. Jako baza dla budowanego programu będzie wykorzystany wcześniej wykonany program (052_Moce.vi).

Zakres zadania: zastosowanie dodatkowych modułów funkcyjnych do realizacji rozszerzonych zadań w układzie symulatora obwodu elektrycznego. Wykorzystanie wcześniej wykonanych programów i modułów funkcji zaawansowanych jako elementów schematu blokowego.

1. **Panel czołowy programu.** Uruchomić program LabVIEW i otworzyć plik z programu 052_Moce.vi. Program ten był wykonany w ramach zadania 5.3.2.
2. 052_Moce.vi jest w pełni funkcjonującym programem, umożliwiającym wyznaczanie podstawowych parametrów założonego obwodu elektrycznego RLC.
3. Przed przystąpieniem do wykonywania kolejnych czynności należy zapisać program pod nową nazwą 053_Oscylogram_RLC.vi.
4. Uzupełnić panel czołowy programu o dwa wyświetlacze graficzne typu Waveform Chart. Nadać im nazwy „Napięcie" i „Prąd". Ustawić wartość maksymalną osi x na 0,1.
5. Rozmieścić elementy panelu czołowego w sposób analogiczny do przedstawionego na rysunku.

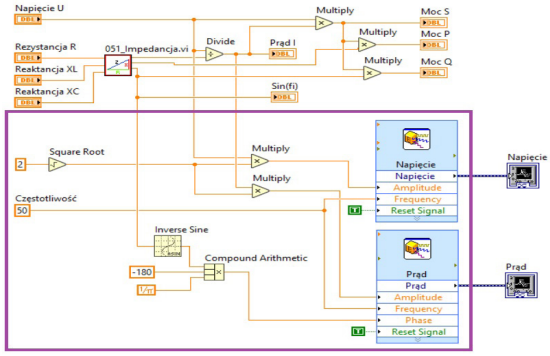


6. **Uzupełnianie schematu blokowego programu.** Do istniejącego schematu blokowego programu należy dodać następujące elementy znajdujące się na palecie Functions:

- dwa moduły mnożenia Multiply (Programming → Numeric),
- moduł pierwiastka Square Root (Programming → Numeric),

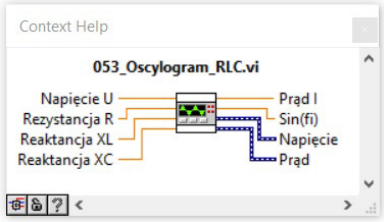
- moduł Compound Arithmetic (Programming → Numeric),
- moduł Inverse Sine (Mathematics → Elementary → Trigonometric),
- dwa moduły Simulate Signal (Signal Processing → Waveform Generations),
- trzy stałe liczbowe DBC Numeric Constant (Programming → Numeric). Stałym liczbowym nadać wartości 2, 50 oraz -180,
- stałą liczbową 1/π (Programming → Numeric → Math Constants),
- dwie stałe logiczne True Constant (Programming → Boolean).

7. W oknach konfiguracyjnych modułów Simulate Signal w polu Signal Name ustawić nowe nazwy sygnałów. W jednym wprowadzić nazwę „Napięcie", w drugim „Prąd".
8. Rozmieścić wstawione do schematu blokowego elementy oraz wykonać stosowne połączenia w sposób analogiczny do przedstawionego na rysunku. W fioletowej ramce zaznaczono nowy tworzony fragment programu.



9. **Uruchomienie aplikacji.** Uaktywnić okno panelu czołowego.
10. Wprowadzić liczby w pola kontrolek/regulatorów liczbowych. W celu sprawdzenia poprawności działania stworzonego projektu uruchomić go, wykorzystując funkcję Run Continuously (ciągłe działanie).
11. Obserwować informacje prezentowane na poszczególnych obiektach front panelu.
12. Zatrzymać funkcjonowanie programu i dokonać zmian w ustawieniach parametrów wyświetlaczy graficznych. W obu obiektach odznaczyć aktywną opcję auto skalowania osi Y. W tym celu uaktywnić menu kontekstowe obiektu i zakładki Y Scale, a następnie odznaczyć opcję AutoScale Y.
13. Dla obu wyświetlaczy ustawić sztywne wartości minimum i maksimum osi Y. Dla wyświetlacza „Napięcie" ustawić wartość min. na -80, a max. na 80. Dla wyświetlacza „Prąd" ustawić wartość min. na -20, a max. na 20.
14. Uruchomić ponownie program i zwrócić uwagę na zmiany w sposobie wyświetlania obu sygnałów.
15. Dokonać zapisu programu na dysku pod aktualną nazwą 053_Oscylogram_RLC.vi.

16. **Tworzenie ikony i deklarowanie terminali we/wy programu.** Uaktywnić okno panelu czołowego i otworzyć narzędzie edytora ikon. Za pomocą dostępnych w uaktywnionym oknie narzędzi stworzyć ikonę, która będzie charakteryzowała budowany przyrząd.
17. Następnie opracować układ konektorów terminali wejść i wyjść dla projektowanego przyrządu wirtualnego. W tworzonym przyrządzie wirtualnym obiektami wejściowymi są kontrolki „Napięcie U", „Rezystancja R", „Reaktancja XL" i „Reaktancja XC" (kontrolki numeryczne), a obiektami wyjściowymi wskaźniki „Prąd I", i „sin(fi)" oraz wyświetlacze graficzne „Napięcie" i „Prąd".
18. Po wykonaniu czynności związanych z przygotowaniem ikony programu oraz terminali wejścia i wyjścia, informacja prezentowana w oknie Context Help dla tego programu powinna być analogiczna do przedstawionej na rysunku.



19. Dokonać zapisu programu na dysku pod obecną nazwą 053_Oscylogam_RLC.vi. Zamknąć plik programu.

5.3.4. Moce – rozbudowanie symulatora układu do pomiaru wybranych wielkości elektrycznych

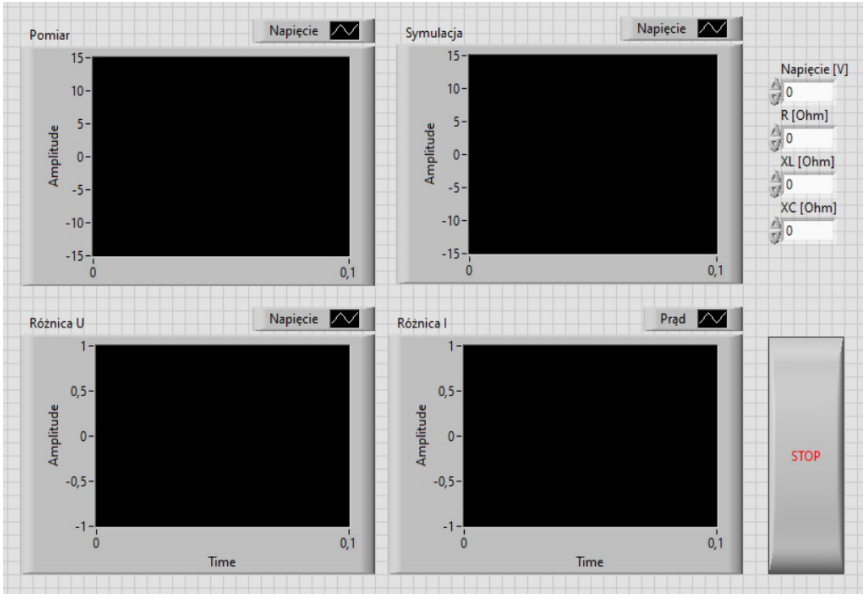
Cel zadania: ważnym czynnikiem przy realizacji zadań diagnostycznych jest ocena rejestrowanych parametrów technicznych przez układ pomiarowy. Często ocena diagnostyczna polega na określeniu parametrów stanu układu na podstawie sygnałów wynikowych. W tym celu powszechnie w układach diagnostycznych stosuje się – jako układ porównawczy – moduł symulatora lub modele matematyczne. Ćwiczenie polega na zbudowaniu aplikacji, umożliwiającej określenie podstawowych parametrów obwodu elektrycznego RLC i napięcia zasilania układu na podstawie sygnałów wyjściowych rejestrowanych przez układ pomiarowy.

Zakres zadania: zastosowanie procedur porównawczych symulatora układu pomiarowego oraz modułu z modelem matematycznym układu.

1. **Panel czołowy programu.** Uruchomić środowisko LabVIEW i otworzyć nowy plik programu. Zbudowany front panel będzie umożliwiał przedstawienie zarejestrowanego sygnału diagnostycznego oraz – za pomocą kontrolek numerycznych – wprowadzenie do symulatora procesu (modelu matematycznego) przypuszczalnych danych pierwotnych układu. Na podstawie tych danych wygenerowany zostanie sygnał teoretyczny przebiegu czasowego funkcji napięcia i prądu oraz różnica w przebiegach sygnału napięcia U i prądu I między układem pomiarowym a symulatorem. Zadanie polega na wstawieniu takich danych do kontrolek „Napięcie U", „R", „XL" i „XC", żeby różnica między sygnałem z układu pomiarowego i symulatora wynosiła 0.
2. Korzystając z elementów dostępnych w palecie Controls w oknie front panelu umieścić:

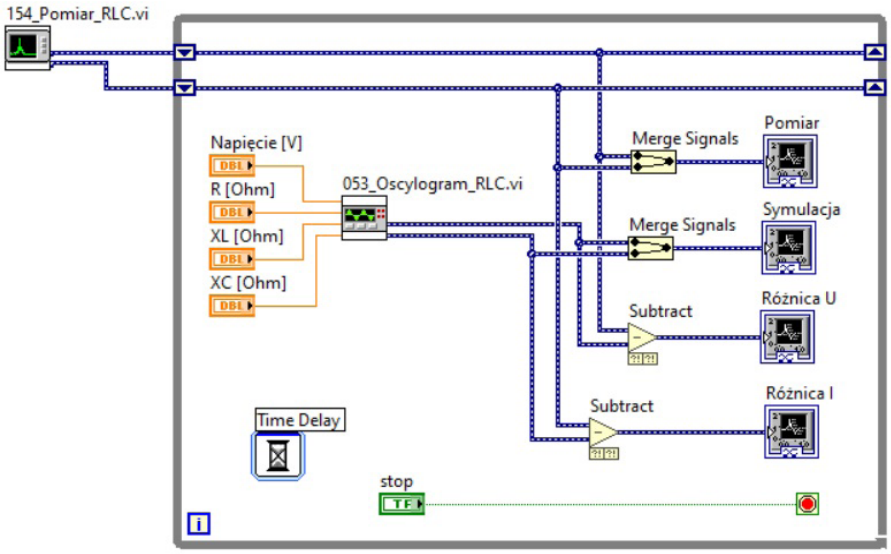
- cztery wyświetlacze graficzne typu Waveform Chart (Modern → Graph). Wyświetlaczom graficznym nadać nazwy: „Pomiar", „Symulacja", „Różnica U" i „Różnica I",
- cztery kontrolki numeryczne typu Numeric Control (Modern → Numeric). Kontrolkom numerycznym nadać nazwy: „Napięcie U", „R", „XL" i „XC",
- przycisk typu Stop Button (Modern → Boolean).

3. Rozmieścić elementy panelu czołowego w sposób analogiczny do przedstawionego na rysunku.



4. Zapisać program pod nazwą 054_Analizator_RLC.vi

5. **Tworzenie schematu blokowego programu.** Korzystając z obiektów dostępnych w palecie Functions, utworzyć schemat blokowy o wyglądzie zbliżonym do zaprezentowanego na rysunku.

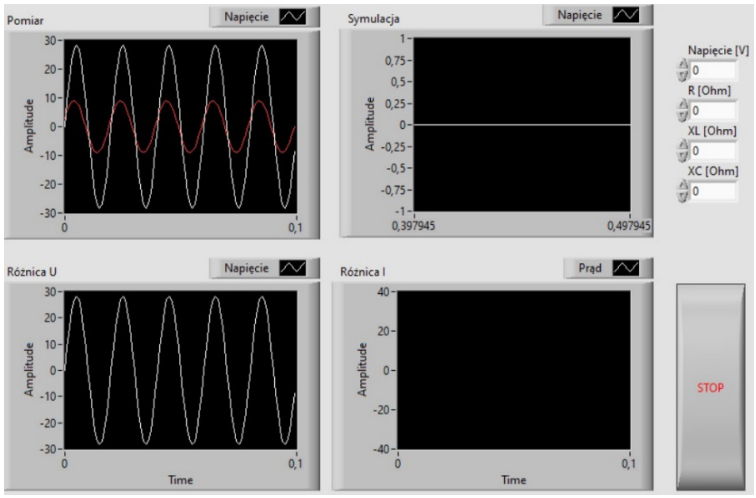


6. W tym celu umieścić w oknie schematu blokowego następujące elementy:
- strukturę programową typu While Loop (Programming → Structures),
- w strukturze programowej While Loop zdefiniować dwa rejestry przesuwne. Użyć funkcji Add Shift Register dostępnej w menu kontekstowym pętli While,
- moduł opóźnienia czasowego Time Delay (Programming → Timing). Ustawić wartość opóźnienia na 0,1 sek,
- dwa moduły Merge Signals (Express → Signal Manipulation),
- dwa moduły odejmowania Subtrack (Programming → Numeric).

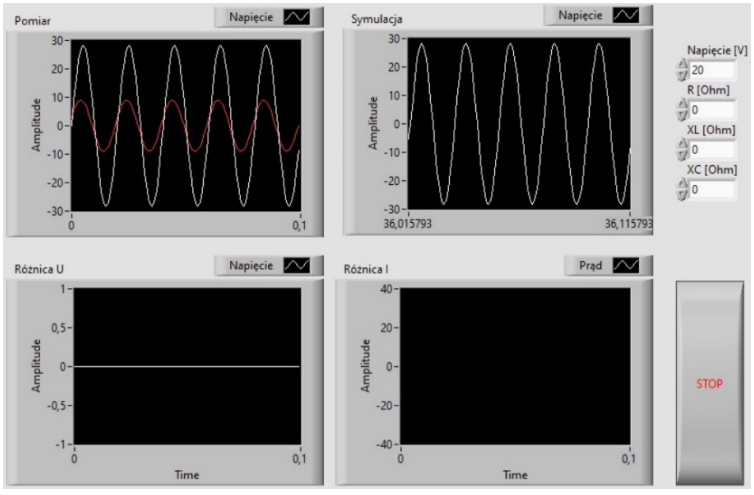
7. Następnie w palecie Function należy wejść do zakładki „Select a VI…". W uaktywnionym oknie Select the VI to Open należy odnaleźć i wstawić do schematu blokowego następujące pliki: 053_Oscylogram_RLC.vi, 154_Pomiar_RLC.vi.
8. Uporządkować okno schematu blokowego i dokonać niezbędnych połączeń.
9. Zapisać dokonane zmiany w aplikacji w pliku programu pod bieżącą nazwą (054_Analizator_RLC.vi).

10. **Uruchomienie aplikacji.** Uaktywnić okno panelu czołowego.
11. W celu sprawdzenia poprawności działania zbudowanego programu, uruchomić go, wykorzystując funkcję Run. Program zawiera pętlę While, służącą do podtrzymywania działania programu. Zakończenie działania programu odbywa się poprzez naciśnięcie przycisku [STOP].

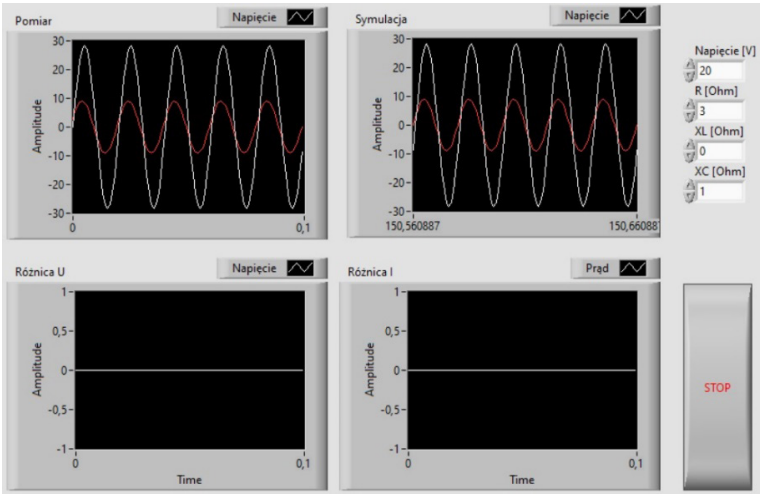
12. Po uruchomieniu programu automatycznie zostaną wygenerowane sygnały z symulatora układu pomiarowego (wyświetlacz „Pomiar”). Sygnały te są generowane na podstawie wybranych z pewnego zakresu ustawień losowych (np. analogiczny do przedstawionego na rysunku).



13. Wygenerowanie sygnału na wyświetlaczu „Symulacja” odbywa się po wprowadzeniu wartości do kontrolek „Napięcie U”, „R”, „XL” i „XC”.
14. Pierwsze zadanie to ustalenie napięcia zasilania za pomocą kontrolki „Napięcie U”. Należy zwiększać wartość napięcia o jeden aż do momentu wygenerowania sygnału o amplitudzie identyczej jak w układzie pomiarowym. Rezultatem prawidłowo wykonanego działania jest otrzymanie sygnału 0 na wyświetlaczu „Różnica U”.



15. Następnie, w celu określenia nieznanych jeszcze parametrów obwodu elektrycznego R, XL i XC, należy zmieniać wartości na kontrolkach numerycznych „R”, „XL” i „XC”. Dla ograniczenia możliwych, generowanych losowo, ustawień parametrów R, XL i XC szukane wartości są ograniczone do następujących ustawień R = 0, 1, 2, 3, 4 lub 5, a XL i XC = 0, 1, 2, 3 lub 4.
16. Wstępne ustawienia R, XL i XC można oszacować na postawie wykresu prądu na wyświetlaczu „Pomiar”, patrząc na amplitudę przebiegu czasowego prądu i przesunięcie fazowe między przebiegiem prądu i napięcia oraz na przebieg sygnału czasowego na wyświetlaczu „Różnica I”.
17. Zmieniać wartości na kontrolach „R”, „XL” i „XC” aż do momentu, kiedy przebieg czasowy prądu na wyświetlaczu „Różnica I” będzie wynosił 0.
18. Rozwiązaniem zadania inżynierskiego jest otrzymanie sygnałów czasowych na wyświetlaczach „Różnica U” i „Różnica I” analogicznych do przedstawionych na kolejnym rysunku.



19. Operację wyszukiwania ustawień parametrów obwodu elektrycznego należy kilkukrotnie powtórzyć.
20. Dokonać zapisu programu na dysku pod obecną nazwą 054_Analizator_RLC.vi. Zamknąć plik programu.

5.4. Pytania kontrolne

1. Jakie systemy pomiarowe pod kątem topologii oraz towarzyszących im wad i zalet?
2. Jakie zadania są stawiane aplikacjom w systemach kontrolno-pomiarowych?
3. Jakie są zalety i wady szeregowej i równoległej transmisji danych w systemach kontrolno-pomiarowych?
4. Jakie metody transmisji danych opisywane są pojęciami simplex, half duplex i full duplex?
5. Do jakich zastosowań można wykorzystać aplikacje oparte na architekturze prostej oraz ogólnej i dlaczego?
6. Na czym polega główna zaleta aplikacji opartej na architekturze maszyny stanu w stosunku do aplikacji opartej na architekturze wielokrotnych struktur wyboru?

6. Tworzenie programów z zadaniami diagnostyki stanu urządzeń i wspomagających decyzję

6.1. Cel zajęć

Rozdział szósty zawiera informacje związane z zawansowanymi sposobami zarządzania funkcjonowaniem aplikacji. Obejmuje zagadnienia takie jak: przedstawienie możliwości środowiska programistycznego LabVIEW w zakresie budowy wirtualnych aplikacji diagnostycznych dla zastosowań inżynierskich, wykorzystanie poznanych wcześniej procedur zapisu danych do pliku w celu realizacji funkcji magazynowania danych, opracowanie archiwizatora danych z wykazem wybranych zdarzeń, poznanie funkcjonowania aplikacji kontrolno-pomiarowych i ich możliwości operacyjnych w zakresie prowadzenia testów diagnostycznych, aktywne zarządzanie funkcjonalnością i właściwościami elementów umieszczonych na front panelu podczas działania programu, poznanie zasad wykorzystania węzłów właściwości w programach.

Celem zajęć jest w szczególności poznanie:

- rozbudowanej techniki zapisu danych do pliku, z zastosowaniem procedury automatycznego generatora nazwy pliku,
- sposobu kontroli zadeklarowanych wartości pomiarowych oraz sygnalizacja przekroczeń ustawionych limitów,
- obsługi klastra błędu oraz możliwości jego wykorzystania w programach VI,
- metody konfigurowania właściwości programów oraz dostosowanie opcji funkcjonowania programu zgodnie z potrzebami użytkownika,
- możliwości wykorzystania węzłów właściwości do sterowanie wybranymi opcjami i ustawieniami elementów front panelu podczas działania programu.

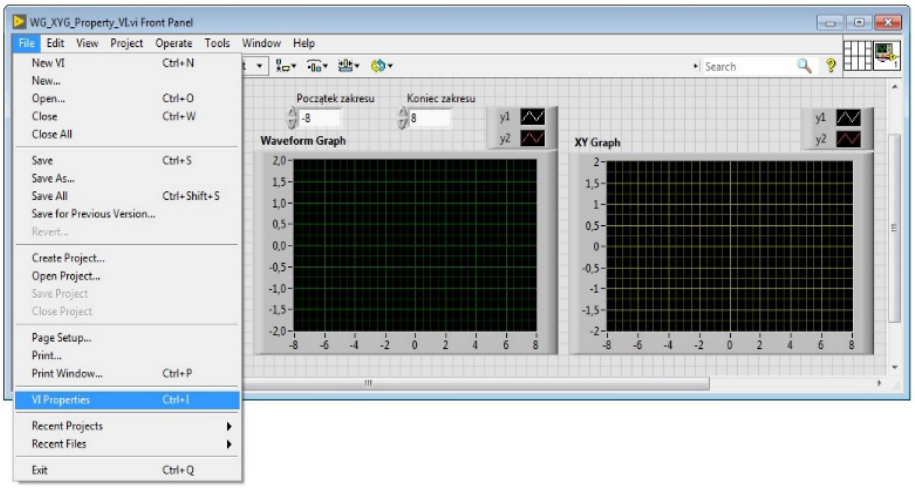
6.2. Wstęp

6.2.1. Narzędzie VI Properties – modyfikacja właściwości programów

Funkcjonowanie programu często wiąże się z koniecznością ustawień specyficznych wymagań całej aplikacji. Jest to spowodowane dążeniem do odpowiedniego przedstawiania danych użytkownikowi, w tym sposobów wywoływania poszczególnych procedur, dopasowania się okna interfejsu do konkretnych wymagań, przygotowania dokumentacji budowanego narzędzia, przygotowania dodatkowych danych pomocniczych związanych z obsługą aplikacji, a także zabezpieczenia programu i kodu źródłowego przed zmianą lub poznaniem [2, 14, 20].

Narzędzia systemowe umożliwiające dopasowanie funkcjonalności programu dostępne są w każdym programie VI. Umożliwiają zarządzanie ustawieniami i właściwościami całego programu (VI). Dzięki temu programista może zadecydować o sposobie wyświetlania programu, określić, czy program ma działać jako procedura (w tle),

czy jako wywoływany element systemu pomiarowego. Odpowiednie ustawienia VI wzbogacają możliwości systemu kontrolno-pomiarowego. Ustawienia wybranych opcji dostępne są po wywołaniu narzędzia VI Properties. Sposób wywołania narzędzia VI Properties został przedstawiony na rysunku 6.1.

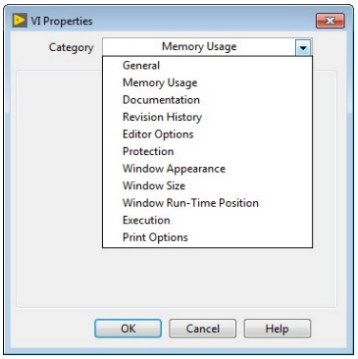


Rys. 6.1. Wywołanie narzędzia VI Properties dla programu w środowisku LabVIEW

Narzędzie VI Properties umożliwia określenie modyfikacji właściwości danego programu. Po wyświetleniu okna dialogowego VI Properties możliwe jest wprowadzenie indywidualnych dla danej aplikacji rozwiązań programowych, ułatwiających obsługę systemu pomiarowego [14].

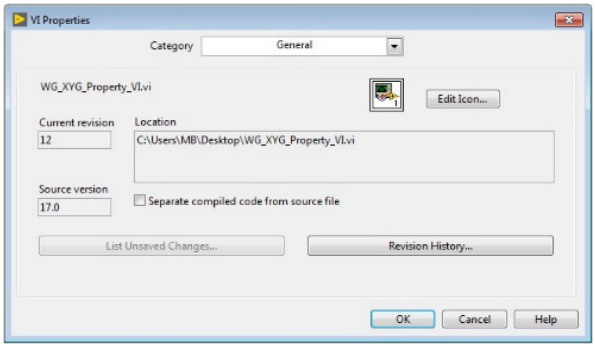
Funkcje ustawień właściwości programu VI (VI Properties) są pogrupowane według kategorii (rys. 6.2):

- General – ustawienia właściwości ogólnych programu,
- Memory Usage – informacje związane z użyciem pamięci przez aplikację VI,
- Documentation – edytor dokumentacji programu i funkcji pomocy,
- Revision History – przedstawianie historii zmian w programie VI,
- Editor Option – opcje edytora programu,
- Protection – funkcje zabezpieczania programu i jego ochrony,
- Windows Appearance – zarządzanie wyglądem okna programu,
- Window Size – ustawianie rozmiaru okna i sposobu wyświetlania,
- Windows Run-Time Position – deklarowanie pozycji okna programu podczas uruchamiania,
- Execution – wykonywanie programu,
- Print Option – opcje drukowania.



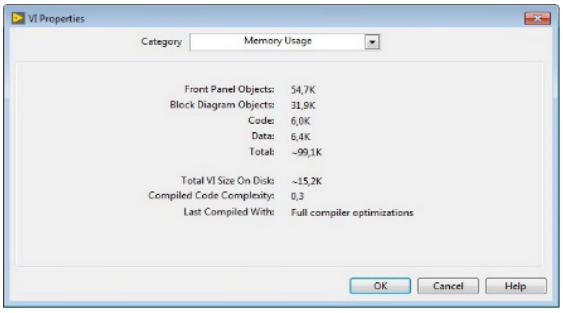
Rys. 6.2. Kategorie ustawień właściwości programu VI

W oknie kategorii General (rys. 6.3) użytkownik może wejść do edytora ikon i wykonać indywidualną reprezentację graficzną programu VI oraz sprawdzić aktualną wersję programu i aktualną lokalizację pliku VI.



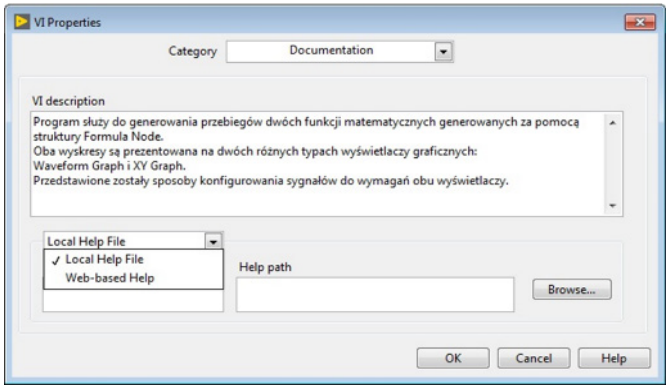
Rys. 6.3. Okno kategorii General w narzędziu VI Properties

W oknie kategorii Memory Usage (rys. 6.4) znajdują się informacje o użyciu pamięci przez bieżącą aplikację VI.



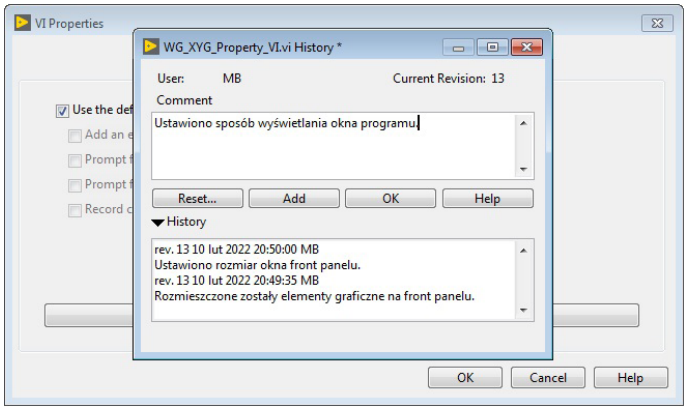
Rys. 6.4. Okno kategorii Memory Usage w narzędziu VI Properties

W oknie kategorii Documentation (rys. 6.5) użytkownik może wygenerować informacje o programie, które będą widoczne po nakierowaniu kursorem myszy na ikony programu (np. Context Help). Dzięki temu, na podstawie prezentowanych informacji, można określić prawdopodobny sposób funkcjonowania programu, a w razie konieczności użyć odnośników do informacji dodatkowych.



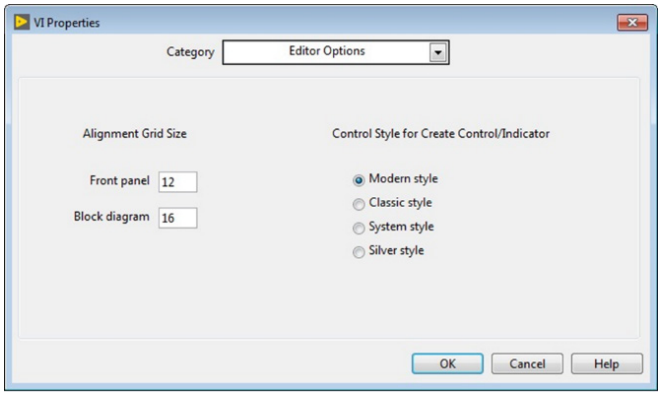
Rys. 6.5. Okno kategorii Documentation w narzędziu VI Properties

W oknie kategorii Revision History (rys. 6.6) użytkownik może budować archiwum zmian dokonywanych w programie. W oknie Revision History można również zobaczyć listę niezapisanych, a wprowadzonych w programie zmian związanych z front panelem lub schematem blokowym.



Rys. 6.3. Okno Revision History w narzędziu VI Properties

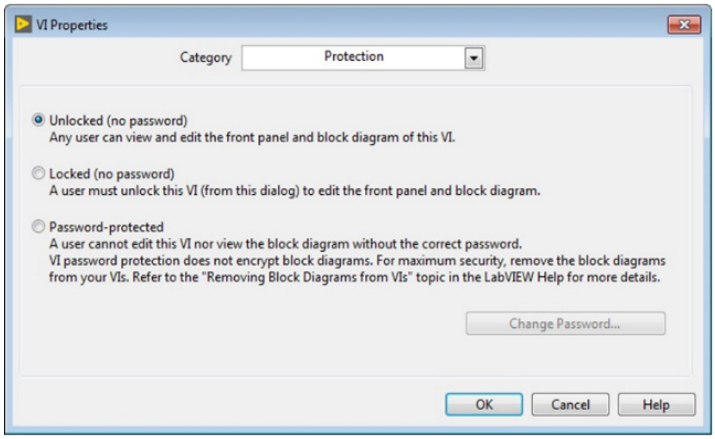
W oknie kategorii Editor Options (rys. 6.7) użytkownik może określić wielkość siatki wyrównywania programu, a także styl kontrolek i wskaźników w budowanym programie VI.



Rys. 6.7. Okno kategorii General

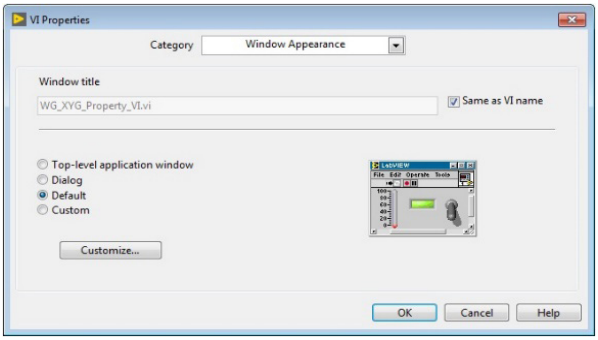
Ważną, ze względu na ochronę tworzonej aplikacji oraz zabezpieczenie programu przed nieautoryzowanymi zmianami, jest kategoria Protection (rys. 6.8). W oknie kategorii Protection możliwe jest ustawienie trzech poziomów dostępu do aplikacji (front panelu i schematu blokowego):

- Unlocked (no password) – użytkownik może przeglądać i edytować front panel i schemat blokowy programu bez żadnych przeszkód,
- Locked (no password) – przed edycją front panelu i schematu blokowego użytkownik musi odblokować zabezpieczenie z poziomu okna dialogowego VI Properties.
- Password-protected – przed wprowadzeniem jakichkolwiek zmian użytkownik musi podać hasło. Hasłem blokowany jest również podgląd schematu blokowego.



Rys. 6.8. Okno kategorii Protection

Ze względu na sposób wyświetlania okna front panelu przez użytkownika, istotne są ustawienia dostępne w kategorii Windows Appearance (rys. 6.9).

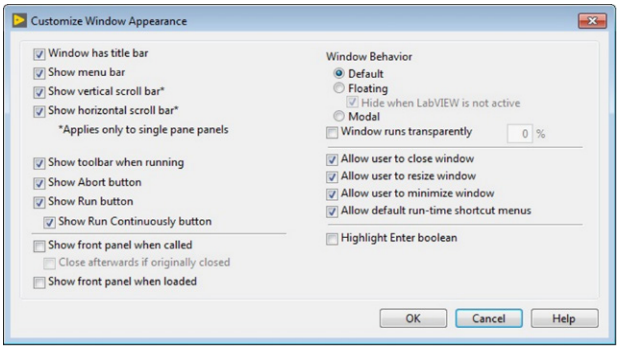


Rys. 6.9. Widok okna kategorii Windows Appearance

W oknie kategorii Windows Appearance dostępne są trzy opcje ze zdefiniowanymi ustawianiami dotyczącymi wyglądu okna programu oraz jedna z ustawieniami wybranymi przez użytkownika:

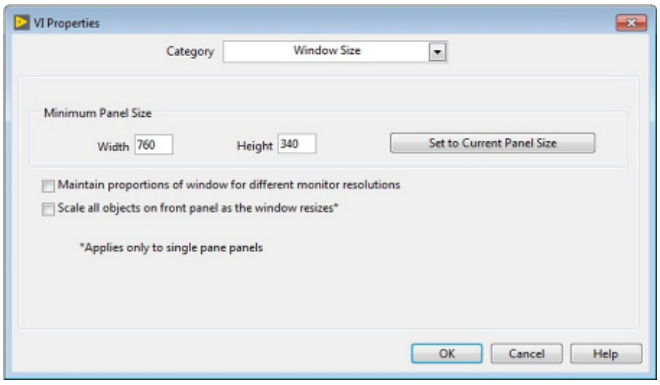
- Top level application – widoczne: pasek nazwy, menu, wywoływane są podprogramy, dostępne skróty klawiaturowe, niedostępne: paski przewijania, narzędzi, zmiana rozmiaru.
- Dialog – jedna aplikacja LabVIEW jest pierwszoplanowa dla innych aplikacji LabVIEW, kolory okna programu będą systemowe, niedostępne są: paski menu, przewijania, narzędzi, zmiana rozmiaru.
- Default – program będzie działał z ustawieniami identycznymi jak przy tworzeniu aplikacji (możliwość wpływania na pracę programu, wygląd panelu sterowania, diagram blokowy).
- Customize – umożliwiają użytkownikowi dowolną konfigurację elementów, jakie mają być wyświetlanie razem z oknem front panelu programu.

Po wywołaniu opcji Customize (rys. 6.10) możliwe jest również zadeklarowanie sposobu wyświetlania programu użytego jako podprogram – czy ma być wykonany w tle, czy uruchomiony jako aplikacja pierwszoplanowa.



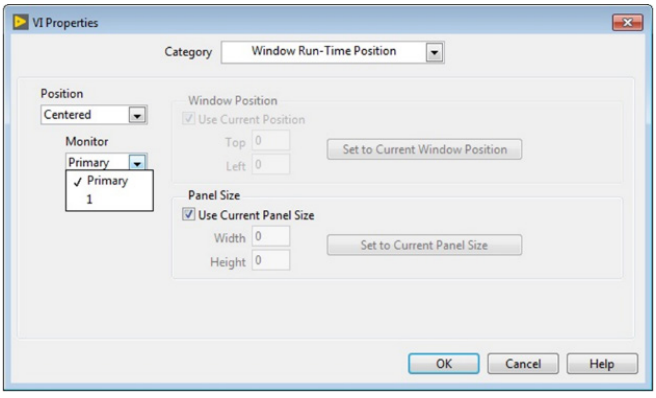
Rys. 6.10. Widok okna ustawień w opcji Customize

W oknie kategorii Window Size (rys. 6.11), związanym z zarządzaniem ustawianiami okna front panelu programu, użytkownik ma możliwość określenia minimalnych wartości wysokości i szerokości okna interfejsu (pola w obszarze Minimum Panel Size). Dodatkowo można określić sposób skalowania elementów panelu czołowego programu przy zmianach wielkości okna front panelu.



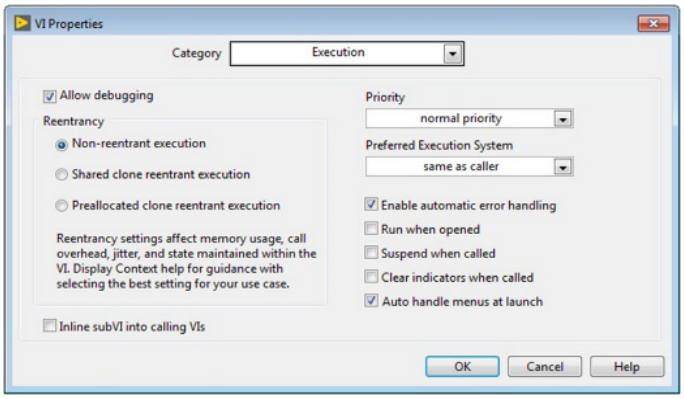
Rys. 6.11. Okno kategorii Windows Size

W oknie kategorii Window Run-Time Position (rys. 6.12), związanego z zarządzaniem ustawianiami okna front panelu, użytkownik ma możliwość określenia ustawień okna programu podczas jego uruchamiania (pozycję na ekranie, wybrać ekran, na którym ma być wyświetlany program i wielkość panelu).



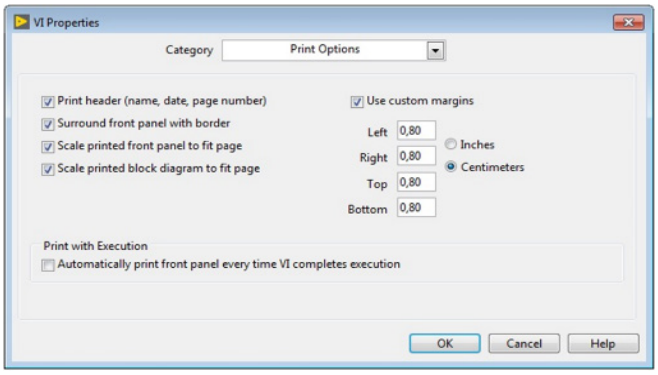
Rys. 6.12. Okno kategorii Windows Run-Time Position

W oknie kategorii Execution (rys. 6.13), związanymi z zarządzaniem wybranymi opcjami wykonywania programu, użytkownik może zezwolić na debugowanie programu, określić priorytety programu oraz preferowany system realizacji.



Rys. 6.13. Okno kategorii Execution

W oknie kategorii Print Options użytkownik może określić opcje drukowania



Rys. 6.14. Okno kategorii Print Options

6.2.2. Projektowanie panelu czołowego

W przypadku systemów kontrolno-pomiarowych, gdzie następuje bieżąca interakcja między użytkownikiem a systemem, programem i obiektem, głównym elementem danego programu dla obsługi jest panel czołowy (front panel) aplikacji [3, 7, 9].

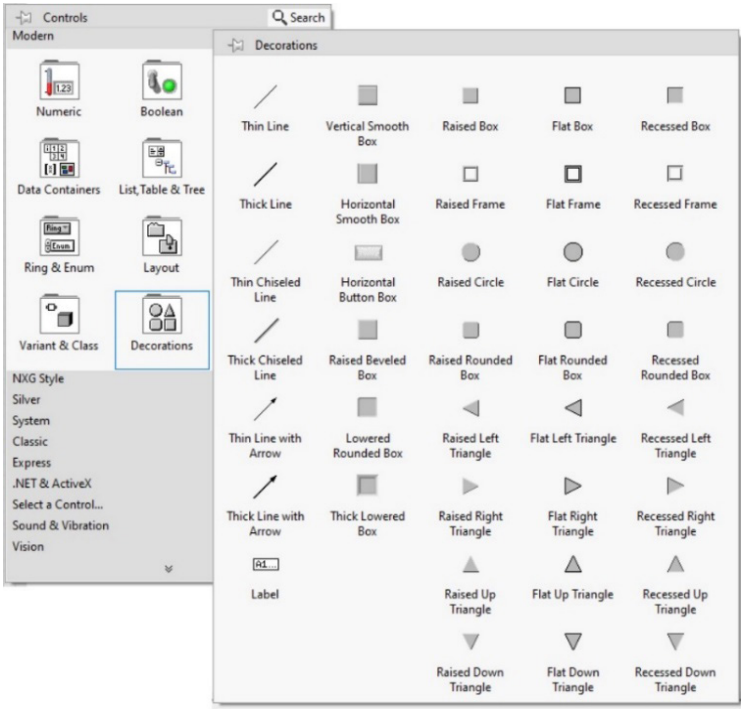
Panel czołowy aplikacji stanowi interfejs komunikacyjny miedzy człowiekiem a systemem. Pełni rolę HMI (Human Machine Interface), który umożliwia zarządzanie programem i kontrolę nad procesem pomiarowym, procesem sterującym lub diagnostycznym.

Ważne cechy interfejsu komunikacyjnego HMI są związane z zapewnieniem użytkownikowi pełnej funkcjonalności programu, czyli zapewnienia dużej przejrzystości, intuicyjności i ergonomiczności obsługi oraz dostępu do zastosowanych rozwiązań.

W celu zapewnienia odpowiedniej funkcjonalności front panelu należy zwrócić uwagę na użycie następujących zasad:

- ograniczenie wyświetlanej, w danym momencie, liczby elementów na front panelu do elementów związanych z realizacją zadania,
- podział elementów znajdujących się na ekranie na grupy powiązane tematycznie,
- wprowadzenie klasyfikacji użytkowników i dopasowanie dostępnych funkcji do kategorii użytkownika (nadawanie priorytetów obsługi programu – admin, user, serwis),
- dopasowanie kolorystyczne elementów, wprowadzenie sygnalizacji graficznej,
- skalowanie front panelu przy zmianach rozdzielczości ekranu i wielkości okna.

Do poprawy czytelności i przejrzystości prezentowanych danych na panelu HMI wykorzystuje się elementy zawarte w zakładce Decorations palety Controls. W tym miejscu znajdują się elementy, które nie są związane z procesem funkcjonowania aplikacji (kontrolki, wskaźniki), tylko umożliwiają realizację celów „ozdobnych" podnoszących czytelność, intuicyjność obsługi i ergonomię panelu czołowego. W zakładce tej (rys. 6.15) są dostępne różne linie i strzałki, etykiety i pola tekstowe oraz ramki, obwódki i obwiednie.

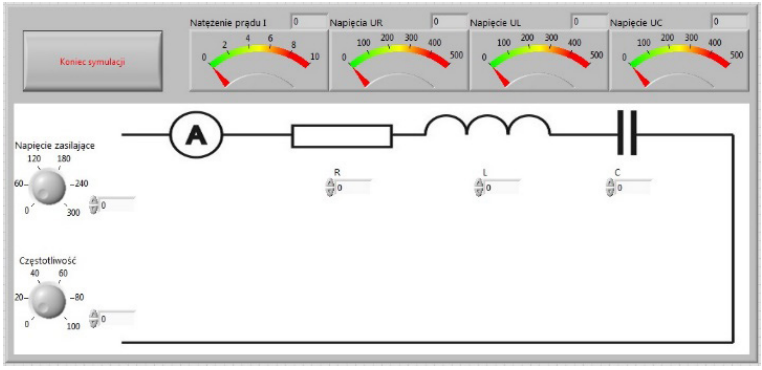


Rys. 6.15. Widok elementów dekoracyjnych dostępnych na karcie Decorations

Do poprawy czytelności i funkcjonalności paneli czołowych (HMI) można wykorzystać importowane z zewnątrz pliki z grafiką lub zdjęcia. W tym celu należy:

- użyć funkcji „Import Picture to Clipboard...” (z zakładki Edit), wybrać w oknie dialogowym dany plik graficzny, a następnie użyć polecenia Paste (z menu Edit) lub [Ctrl] + [V],
- wstawić plik bezpośredniego eksploratora plików do okna front panelu metodą przeciągnij i puść.

Front panel aplikacji z zaimportowanym schematem z pliku graficznego (.jpeg) został przestawiony na rysunku 6.16.



Rys. 6.16. Widok front panelu z wstawionym zewnętrznym elementem graficznym

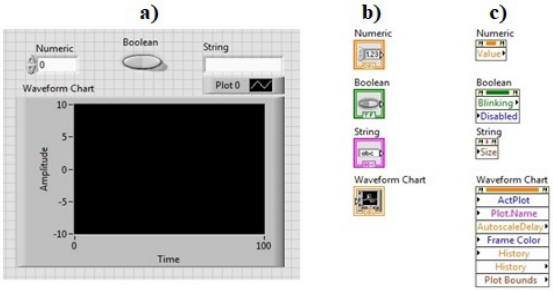
6.2.3. Węzły właściwości

Zastosowanie metod zwiększania funkcjonalności i czytelności front paneli programu, w postaci elementów dekoracyjnych, zaimportowanych grafik, a także zmiany kolorystyki i indywidualizacji kontrolek i wskaźników, umożliwia tylko wprowadzenie statycznych zmian w wizualizacji elementów. Nie daje jednak możliwości automatycznej zmiany właściwości elementów podczas funkcjonowania programu [1, 20].

Możliwość dynamicznych, dostosowanych do aktualnie realizowanych procedur w programie, zmian wyglądu poszczególnych cech elementów wyświetlanych na front panelu dają węzły właściwości (Porperty Node). Charakteryzują się poniższymi cechami:

- umożliwiają odwołanie się, pobranie i zmianę dowolnej właściwości obiektu podczas działania programu,
- pobierają (odczytują) i zmieniają (zapisują) właściwości obiektu,
- węzły z wybranymi cechami umieszcza się w schemacie blokowym i są „wykonywane” razem z kodem głównym programu.

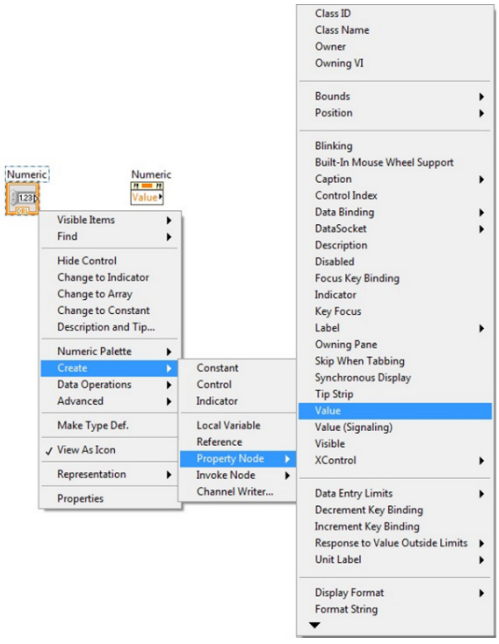
Węzły właściwości (Property Nodes) dla przykładowych elementów front panelu zostały przedstawione na rysunku 6.17.



Rys. 6.17. Elementy front panelu i schematu blokowego programu: a) kontrolki i wskaźniki, b) terminale kontrolek i wskaźników, c) węzły właściwości kontrolek i wskaźników

Każda właściwość obiektu przypisana (zdefiniowana) jest do ściśle określonego typu danych (zachowanie typów danych programowych: typ numeryczny, logiczny, klastrowy, łańcuchowy itp.). Dzięki temu możliwe jest wkomponowanie węzłów właściwości w kod źródłowy programu, wykorzystanie danych z programu do zmian właściwości, ale także pobieranie danych z obiektów przez węzły właściwości i użycie ich w kodzie programu.

Węzły właściwości definiuje się (patrz rys. 6.18) dla każdego obiektu (kontrolki lub wskaźnika) oddzielnie za pomocą polecenia Property Node w grupie Create menu kontekstowego.

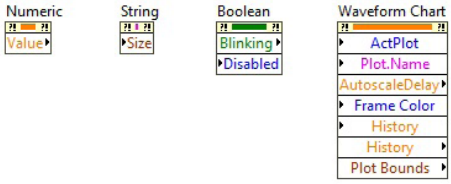


Rys. 6.18. Procedura zdefiniowania właściwości obiektu w węźle właściwości

Najczęściej węzły właściwości używane są do:

- odczytu i zmiany położenia obiektu na front panelu,
- odczytu i zmiany wielkości obiektu na front panelu,
- ukrywania i pokazywania obiektów na front panelu,
- migania obiektów na front panelu,
- zarządzania kolorami elementów obiektów,
- zarządzania skalami i zakresami danych w obiektach,
- zaznaczaniu i wywoływaniu obiektów,
- pobierania, zmiany i kasowania danych w obiektach.

Węzły właściwości (Property Node) mogą być jednoelementowe lub wieloelementowe, mogą służyć do zapisu (typ write) i do odczytu (typ read) danej właściwości. Węzły właściwości o różnym typie oraz jedno-, dwu- i wieloelementowe przedstawiono na rysunku 6.19.

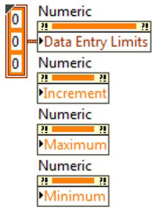


Rys. 6.19. Węzły właściwości o różnym charakterze

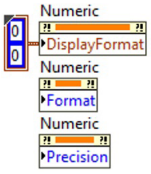
Na rysunku 6.20 przedstawiono oznaczenie graficzne węzłów właściwości oraz ich opis dla podstawowych i najczęściej wykorzystywanych funkcjonalności kontrolek i wskaźników.

Rys. 6.20. Węzły właściwości Blinking, Disable, Key Focus oraz Value dla kontrolek i wskaźników

Inne węzły właściwości, stosowane przy zarządzaniu właściwościami wskaźników i kontrolek, mogą być związane z formatowaniem i sposobem wyświetlania danych (rys. 6.21). Za pomocą takiego węzła właściwości można w odpowiedni sposób przedstawiać prezentowane dane.



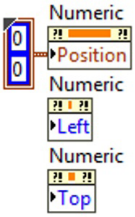
Data Entry Limits (limity danych)
Może być klastrem lub pojedynczą wartością
Increment – skok
Maximum – wartość maksymalna
Minimum – wartość minimalna



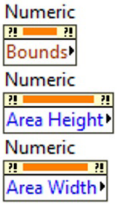
Display Format (format wyświetlania danych)
Może być klastrem lub pojedynczą wartością
Format – **0** – dziesiętny, **1** – naukowy, **2** – inżynierski,
3 – binarny, **4** – ósemkowy, **5** – szesnastkowy, **6** – czas względny,
7 – godzina i data, **8** – oznaczenie SI lub **9** – niestandardowy
Precision – liczba liczb po przecinku

Rys. 6.21. Właściwości zarządzające sposobem i formatowaniem wyświetlanych informacji

Do sterowania ustawieniami, umiejscowieniem i wielkością konkretnej kontrolki lub wskaźnika, można stosować węzły właściwości przedstawione na rysunku 6.22. Za pomocą takich węzłów właściwości można przestawiać umiejscowienie poszczególnych elementów front panelu i zmieniać ich rozmiar w zależności od wagi danego komunikatu.



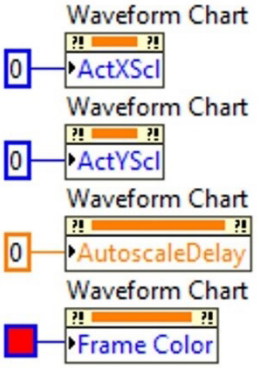
Position (pozycja położenie na ekranie)
– określa lewy górny róg elementy
Może być klastrem lub pojedynczą wartością
Left – umiejscowienie od lewej lewego górnego narożnika
Top – umiejscowienie od góry lewego górnego narożnika



Bound Format (wysokość i szerokość danych)
– obszar obwiedni obiektu
Może być klastrem lub pojedynczą wartością
Area Height – wysokość obiektu
Area Width – szerokość obiektu

Rys. 6.22. Właściwości kontrolek i wskaźników zarządzające ustawieniami i rozmiarem elementów panelu czołowego

W analogiczny sposób można sterować właściwościami innych elementów na front panelu. Do szczególnych elementów, występujących na front panelu programu, zaliczane są wyświetlacze graficzne. W tym przypadku liczba właściwości jest większa i obejmuje również formatowanie pola graficznego. Na rysunku 6.23 przedstawiono właściwości związane z ustawieniami osi i kolorów obiektu graficznego, na rysunku 6.24 przedstawiono opcje określające umiejscowienie wyświetlacza na front panelu programu oraz zakresy danych na osiach wykresu, a na rysunku 6.25 właściwości historii, opisu wykresów.



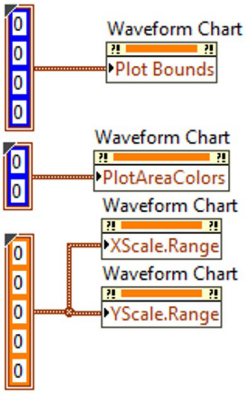
Active X Scale (aktywność skali osi X)
Pobiera lub ustawia aktywną lub bieżącą skalę wykresu X

Active Y Scale (aktywność skali osi Y)
Pobiera lub ustawia aktywną lub bieżącą skalę wykresu X

Autoscale Delay (opóźnienie autoskalowania)
Liczba sekund opóźnienia między dwiema kolejnymi operacjami autoskalowania wykresu, gdy skala musi się zmniejszyć, aby dopasować dane

Frame Color (kolor ramki wykresu
Ustawienie koloru ramki wykresu
Color Box – wybór koloru z palety
Kontrolka / wskaźnik numeryczny – oznaczenie koloru

Rys. 6. 23. Węzły właściwości dla ustawień Activ X/Y Scale, Autoscale Delay oraz Frame Color dla wyświetlaczy graficznych

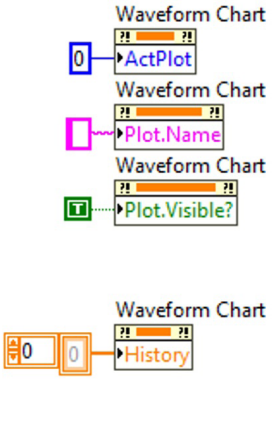


Plots Bounds (bounds – pozycja, położenie na ekranie)
– określa obszaru pola wykresu względem obiektu wykresu
Left – lewa strona
Top – góra
Right – prawa strona
Botton – dół

Plot Area Colors (kolor pola wykresu)
Foreground – kolor wierchu
Background – kolor tła

X Scale Range/YScale Range (ustawienia limitów danych)
Minimum – wartość minimalna
Maximum – wartość maksymalna
Increment – wartość głównego skoku
Minor Increment – wartość pomocniczego skoku
Start – wartość początkowa

Rys. 6. 24. Węzły właściwości dla ustawień Plots Bounds, Plot Area Colors oraz X/Y Scale Range dla wyświetlaczy graficznych



Active Plot (określenie aktywnego wykresu)
– każdy sygnał ma swój numer porządkowy

Plot Name (nazwa wykresu)
– wprowadza lub odczytuję nazwę aktywnego wykresu
Współpracuje i jest powiązany z Active Plot

Plot Visible (widoczność wykresu)
Współpracuje i jest powiązany z Active Plot
True – przebieg wykres jest widoczny
False – przebieg wykresu jest ukryty (niewidoczny)

History (dane historyczne)
Odczytuje punktu wyświetlanych na wykresie danych
Wprowadza dane historyczne (ustawione 0 kasuje dane
i wykres zaczyna się od początku)

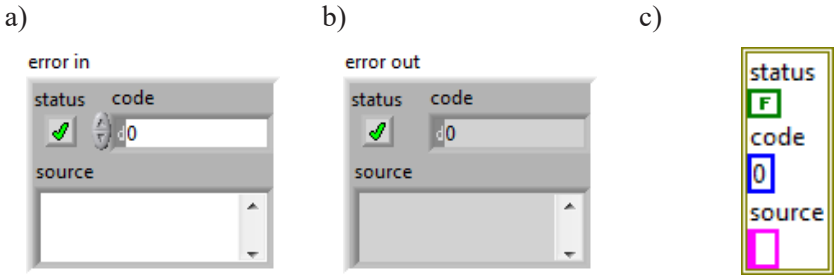
Rys. 6.25. Węzły właściwości dla ustawień Active Plot, Plot Name, Plot Visible oraz History dla wyświetlaczy graficznych

6.2.4. Obsługa klastra błędu

Klaster błędów (Error Cluster) jest predefiniowanym klastrem LabVIEW, który jest używany do przechowywania, obsługi informacji o stanie błędu w programie. Klaster błędu (Error Cluster) zawsze zawiera trzy komponenty:

- status – dana typu logicznego – zawiera informację o wykryciu lub braku błędu,
- code – dana typu liczbowego I32 – zawiera informację o numerze zdiagnozowanego błędu,
- source – dana typu łańcuchowego – zawiera informację o opisie i źródle błędu.

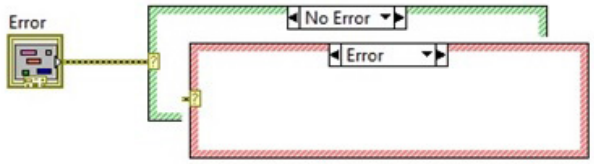
Za pomocą klastra błędu możliwy jest nadzór nad wykonywaniem programu i ingerowanie w jego funkcjonowanie w przypadku wykrycia błędu.



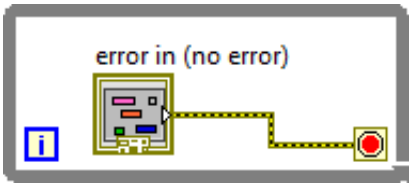
Rys. 6.26. Klaster błędu: a) kontrolka klastra błędu, b) wskaźnik klastra, c) stała klastra błędu

Klaster błędów (Error Cluster) jest specyficznym sygnałem diagnostycznym. Za jego pomocą można sterować przypadkami struktury wyboru (Error / No Error) oraz

funkcjonowaniem pętli (While Loop). Na rysunku 6.27 przedstawiono strukturę wyboru Case sterowaną sygnałem z klastra błędu, a na rysunku 6.28 przedstawiono sposób funkcjonowania pętli warunkowej While, sterowanej za pomocą informacji z klastra błędu.



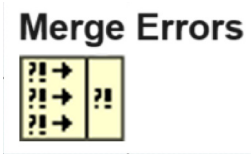
Rys. 6.27. Struktura wyboru Case sterowana klastrem błędu Error Cluster



Rys. 6.28. Struktura pętli While sterowana klastrem błędu Error Cluster

Terminale wejściowe error in i wyjściowe error out, związane są z przekazywaniem informacji o błędzie, posiada je wiele modułów i funkcji. W przypadku posobnego (szeregowego) połączenia elementów przetwarzających dane, możliwe jest komunikowanie się o błędzie między wieloma węzłami. Wykrycie błędu przesyłane jest między modułami i może uruchomić założoną reakcję programu zapisaną w procedurach obsługi błędów i wyjątków.

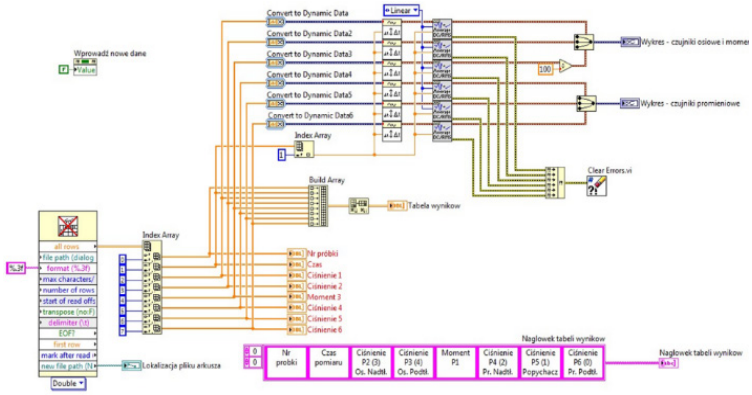
W przypadku powiązania niezależnie przebiegających wcześniej równoległych procesów w algorytmie aplikacji, stosuje się węzeł umożliwiający łączenie sygnałów błędu Merge Error VI (rys. 6.29). W takim przypadku połączony sygnał zawiera informacje o błędzie ze wszystkich procesów i nie ma konieczności wielokrotnej obsługi sygnału o błędzie w procesach połączonych.



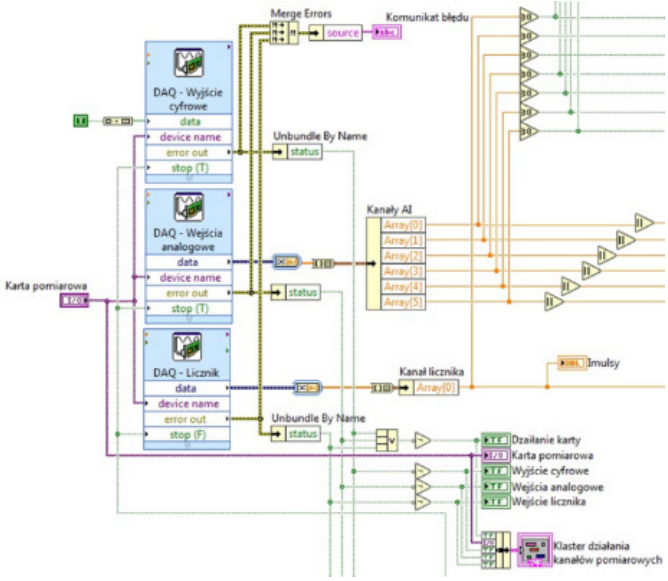
Rys. 6.29. Ikona węzła Merge Errors

Obsługa sygnału klastra błędu w programie do obsługi systemów kontrolno-pomiarowych jest istotna z powodu możliwości przeciwdziałania wykrytym nieprawidłowościom. Można opracować algorytmy umożliwiające uwzględnienie błędu i dalszą pracę programu lub bezpośrednie poinformowanie użytkownika o jego wystą-

pieniu bez ryzyka zawieszenia się działającego programu lub utraty danych z realizowanego procesu pomiarowego. W ten sposób można również przeprowadzać zadania związane z autodiagnostyką programu. Za pomocą dostępnych funkcji można kasować wykryte wszystkie lub wybrane błędy (rys. 6.30), tworzyć procedury ochronne w programie w przypadku jego pojawienia się albo generować komunikaty o wystąpieniu błędu na front panelu użytkownika (rys. 6.31).



Rys. 6.30. Procedura resetowania wykrytych błędów za pomocą modułu Clear Errors.vi



Rys. 6.31. Wyświetlania komunikatu o błędzie z 3 kanałów DAQ na jednym wskaźniku

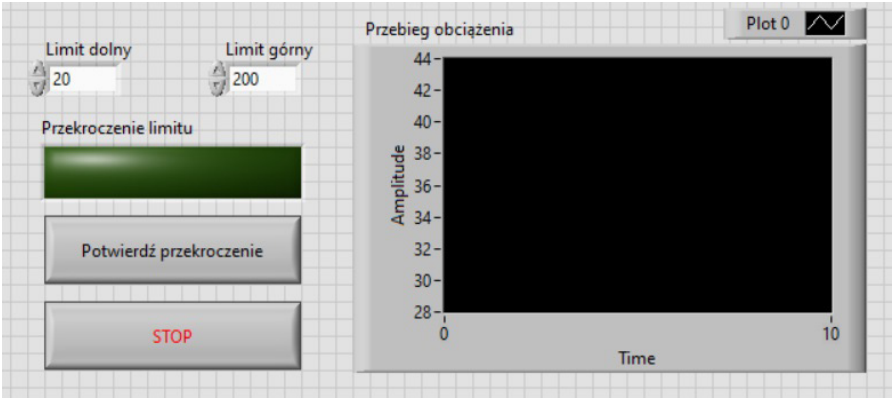
6.3. Zadania

6.3.1. Monitorowanie parametrów technicznych – przebieg i analiza krzywej obciążenia

Cel zadania: utworzenie kompletnego przyrządu wirtualnego do odczytu i wyświetlania przebiegu zmienności obciążenia układu zasilania. Określenie parametrów granicznych, po przekroczeniu których generowany będzie sygnał alarmowy. Wyświetlanie i kasowanie stanu alertu.

Zakres zadania: zastosowanie elementów kontrolno-pomiarowych i procedur do wykrywania stanów nieprawidłowych w monitorowanym układzie. Obsługa funkcji zapisu danych do pliku oraz struktur wyboru.

1. **Panel czołowy programu.** Uruchomić program LabVIEW i otworzyć nowy plik programu w celu utworzenia panelu czołowego o wyglądzie zbliżonym do zaprezentowanego na rysunku.



2. W celu wykonania front panelu zgodnego z założeniami programu należy wstawić następujące elementy dostępne w palecie Controls:

- dwie kontrolki numeryczne typu Numeric Control. Kontrolkom numerycznym nadać nazwy: „Limit dolny” i „Limit górny”,
- przycisk typu Stop Button. Ukryć jego etykietę (Label),
- przycisk typu OK Button. Ukryć jego etykietę (Label),
- wskaźnik logiczny typu Square LED. Nadać mu nazwę „Przekroczenie limitu”,
- wykres typu Waveform Chart. Nadać mu nazwę „Przebieg obciążenia”.

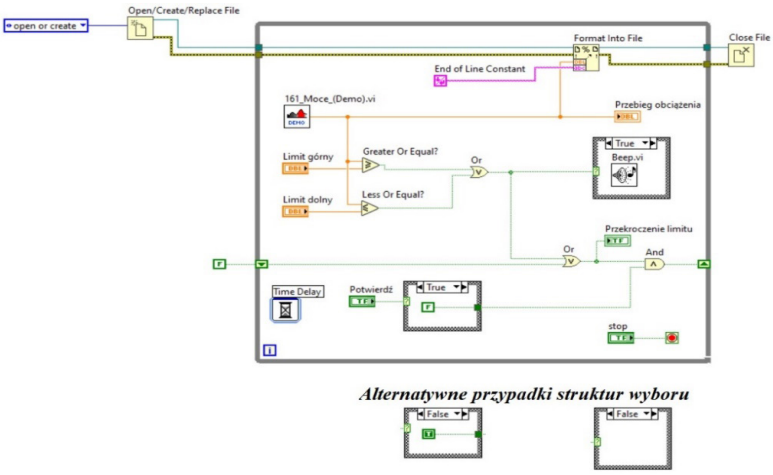
3. Wprowadzić do kontrolek numerycznych wartości 20 i 200, odpowiadające wymaganiom ćwiczenia (analogicznie do przedstawionego na rysunku).

4. Następnie należy ustawić ich wartości domyślne. Do ustawienia wartości domyślnych wykorzystuje się funkcję Make Current Value Default (polecenie dostępne w menu kontekstowym danego elementu w zakładce Data Operations) lub funkcję Make Selected Value Default (polecenie znajdujące się w pasku menu w zakładce Edit po zaznaczeniu danego elementu lub grupy elementów na front panelu).
5. Zapisać bieżącą postać programu w pliku pod nazwą 061_Przebieg_mocy.vi.

6. **Budowa schematu blokowego.** Przejść do okna schematu blokowego i uaktywnić paletę Functions.
7. Z palety Functions wybrać i wstawić do okna schematu blokowego następujące elementy:

- strukturę pętli While (Programming → Structures). W strukturze pętli While zdefiniować rejestr przesuwny Shift Register,
- dwie struktury wyboru Case (Programming → Structures),
- moduł funkcyjny Open /Create /Replace File (Programming → File I/O). Do wejścia operation (0:open) należy wygenerować stałą i zadeklarować opcję open or create,
- moduł funkcyjny Format Into File (Programming → File I/O). Należy zwiększyć liczbę wejść sygnałowych modułu z jednego do dwóch,
- moduł funkcyjny Close File (Programming → File I/O),
- moduł komparatora większe lub równe Greater Or Equal? (Programming → Comparison),
- moduł komparatora mniejsze lub równe Less Or Equal? (Programming → Comparison),
- dwa moduły logiczne Or (Programming → Boolean),
- moduł logiczny And (Programming → Boolean),
- moduł opóźnienia czasowego Time Delay (Programming → Timing),
- moduł generatora sygnału dźwiękowego Beep.vi (Programming → Graphics and Sound),
- dwie stałe logiczne typu False (Programming → Boolean),
- stałą logiczną typu True (Programming → Boolean),
- stałą łańcuchową typu End of Line Constant (Programming → String).

8. Następnie w palecie Function należy wejść do zakładki „Select a VI…". W uaktywnionym oknie Select the VI to Open odszukać i wstawić do schematu blokowego następujący plik: 161_Moce_(Demo).vi. Jest to plik z gotowym programem do generowania przebiegu zmienności obciążenia.
9. Dopasować do potrzeb zadania wielkość powłok struktur pętli While i struktur wyboru Case oraz rozmieścić poszczególne elementy schematu blokowego w sposób analogiczny do przedstawionego na rysunku. Takie rozmieszczenie ułatwi przeprowadzanie kolejnych zadań. Wykonać stosowne połączenia.



10. Zapisać aktualną postać programu w pliku pod bieżącą nazwą (061_Przebieg_mocy.vi).

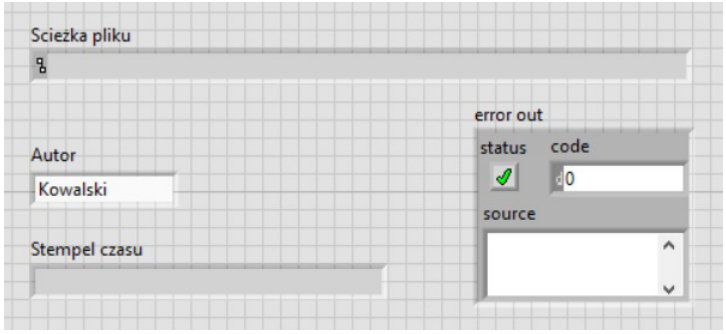
11. **Uruchomienie aplikacji i testowanie procedur.** Upewnić się, że program nie zawiera błędów formalnych i że jest aktywna ikonka przycisku Run. Przejść do okna front panelu i uruchomić program.
12. W momencie uruchomienia programu wyświetli się okno dialogowe, w którym należy wprowadzić nazwę pliku dla zapisywanych danych pomiarowych (np. demo.txt). Pod taką nazwą będą dostępne zarchiwizowane dane pomiarowe.
13. Prawidłowe funkcjonowanie programu umożliwia wyświetlanie kolejnych danych pomiarowych na wykresie w zaprogramowanych odstępach czasowych.
14. W przypadku przekroczenia ustawionego limitu dolnego lub górnego, zostanie wygenerowany sygnał dźwiękowy, a wskaźnik „Przekroczenie limitu" zmieni kolor. Skasowanie sygnalizacji alertu odbywa się za pomocą przycisku „Potwierdź przekroczenie".
15. Takie rozwiązanie stosowane jest w systemach monitorowania, gdzie użytkownik jest automatycznie powiadamiany o stanach nieprawidłowych, a zapoznanie się z taką informacją potwierdza odpowiednim przyciskiem. Dzięki temu następuje interakcja między programem a użytkownikiem. Użytkownik jest odciążany od żmudnego procesu bieżącej kontroli wielu parametrów, a reaguje tylko na założone stany.
16. Zakończyć funkcjonowanie programu, naciskając przycisk [STOP].
17. Zamknąć wszystkie aktywne okna programu LabVIEW.
18. Wybrać plik z zapisanymi informacjami (np. demo.txt). Otworzyć plik, wykorzystując aplikację Notatnik. W pliku powinny być zamieszczone wszystkie punkty pomiarowe.
19. Zamknąć okno programu Notatnik.

6.3.2. Generator nazwy pliku

Cel zadania: poznanie mechanizmu działania przykładowego automatycznego generatora nazwy pliku. Wykorzystanie narzędzi umożliwiających usprawnienie pracy operatora systemu do tworzenia nazw plików i informacji o określonej strukturze. Budowa SubVI ze zdefiniowanymi terminalami wejścia i wyjścia oraz opracowanie ikony programu.

Zakres zadania: użycie modułu File Dialog oraz funkcji związanych z przetwarzaniem danych łańcuchowych do zbudowania (zgodnej z wymaganiami użytkownika) procedury automatycznego tworzenia nazwy pliku.

1. **Panel czołowy programu.** Uruchomić program LabVIEW i otworzyć nowy plik programu w celu utworzenia panelu czołowego o wyglądzie zbliżonym do zaprezentowanego na rysunku.



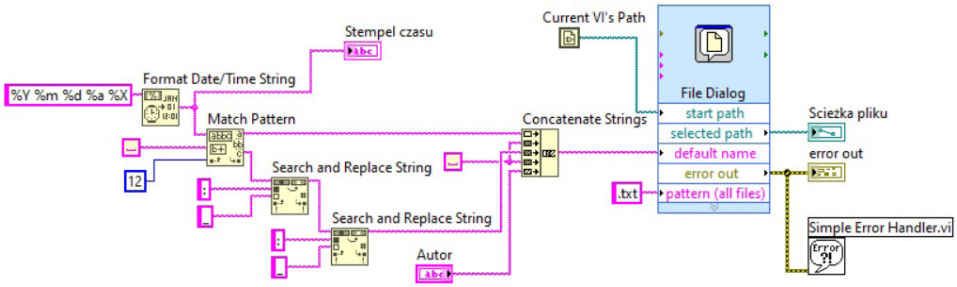
2. W celu wykonania front panelu zgodnego z założeniami programu należy wstawić następujące elementy dostępne w palecie Controls:
- wskaźnik typu File Path Indicator (Modern → String & Path). Nadać mu nazwę „Ścieżka pliku”,
- wskaźnik typu String Indicator (Modern → String & Path). Nadać mu nazwę „Stempel czasu”,
- kontrolkę typu String Control (Modern → String & Path). Kontrolce nadać nazwę „Autor”,
- wskaźnik klastra błędu Error Out 3D.ctl (Modern → Data Containers).

3. Do kontrolki tekstowej wprowadzić tekst „Kowalski”. Następnie ustawić ten tekst jako wartość domyślną kontrolki. Do ustawienia wartości domyślnych wykorzystuje się funkcję Make Current Value Default (polecenie dostępne w menu kontekstowym w grupie Data Operations).
4. Zapisać aktualny stan programu w pliku o nazwie 062_Generator_nazwy.

5. **Tworzenie schematu blokowego programu.** Przejść do okna schematu blokowego i uaktywnić paletę Functions.
6. Z palety Functions wybrać i wstawić do okna schematu blokowego następujące elementy:

- obiekt File Dialog (Programming → File I/O → Advanced File Functions). Skonfigurować jego strukturę wejść i wyjść tak, żeby uzyskać kolejność: „start path", „selected path", „default name", „error out" i „pattern (all files)",
- obiekt Simple Error Handler.vi (Programming → Dialog & User Interface),
- obiekt Format Date / Time String (Programming → Timing). Do wejścia time format string (%c) wygenerować stałą tekstową i wprowadzić do niej następującą wartość „%W %m %d %a %X". Tak zdefiniowane wejście umożliwi prezentację Stempla czasowego w postaci: Rok Miesiąc Dzień tygodnia Godzina:Minuta:Sekunda.
- obiekt Match Pattern (Programming → String),
- dwa obiekty Search and Replace String (Programming → String),
- obiekt Concatenate Strings (Programming → String),
- stałą numeryczną Numeric Constant (Programming → Numeric). Stałej nadać wartość 12.
- obiekt Current VI's Path (Programming → File I/O),
- dwie stałe tekstowe Space Constant (Programming → String),
- pięć stałych tekstowych String Constant (Programming → String). Do dwóch wprowadzić wartość "_", do dwóch: ":", zaś do pozostałej ".txt".

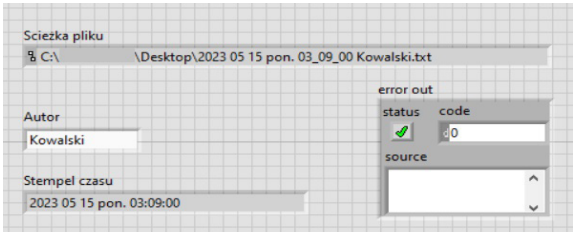
7. Rozmieścić poszczególne elementy schematu blokowego w sposób analogiczny do przedstawionego na rysunku. Dokonać stosownych połączeń miedzy elementami schematu blokowego.



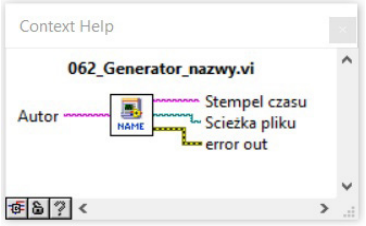
8. Zapisać wprowadzone zmiany w programie w pliku pod bieżącą nazwą (062_Generator_nazwy.vi).

9. **Uruchomienie aplikacji i testowanie procedur.** Po uruchomieniu programu automatycznie zostanie wyświetlone okno dialogowe o nazwie Choose or Enter Path or Folder or File z eksploratorem plików. W oknie dialogowym Nazwa pliku zostanie przedstawiona wygenerowana nazwa pliku.

10. Sprawdzić, czy wygenerowana nazwa pliku spełnia warunki zadania (zawiera aktualną datę, dzień tygodnia, czas oraz określenie Autora). Jeżeli wszystkie dane są zgodne, zamknąć okno dialogowe przyciskiem Save.
11. Po zrealizowaniu procedury generowania nazwy pliku, panel czołowy programu powinien być analogiczny do przedstawionego na rysunku.



12. **Tworzenie ikony i deklarowanie terminali we/wy programu.** Uaktywnić okno panelu czołowego i otworzyć narzędzie edytora ikon. Za pomocą dostępnych w uaktywnionym oknie narzędzi opracować ikonę, która będzie charakteryzowała budowany przyrząd.
13. Następnie wykonać układ konektorów terminali wejść i wyjść dla projektowanego przyrządu wirtualnego. W tworzonym przyrządzie wirtualnym, wejściem jest kontrolka tekstowa „Autor", a wyjściami są: „Stempel czasu", „Ścieżka pliku" i „error out".
14. Po wykonaniu czynności związanych z przygotowaniem ikony programu oraz terminali wejścia i wyjścia informacja prezentowana w oknie Context Help dla tego programu powinna być analogiczna do przedstawionej na rysunku.



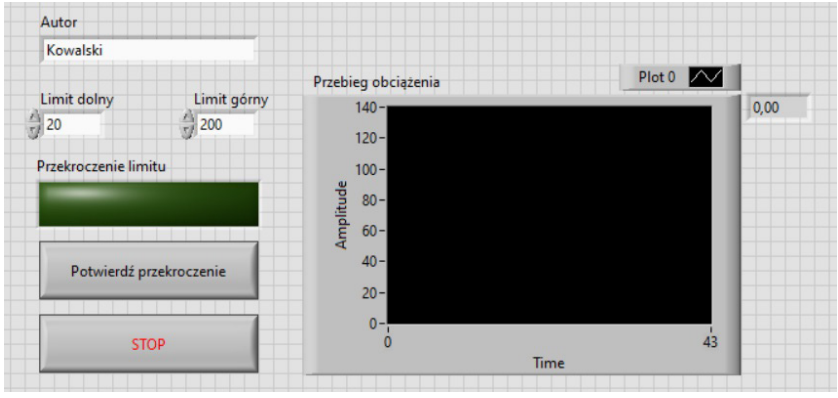
15. Zapisać wprowadzone zmiany w programie w pliku pod bieżącą nazwą (062_Generator_nazwy.vi).
16. Sprawdzić funkcjonowanie stałej formatującej sposób wyświetlania danych stempla czasowego. W tym celu skorzystać z informacji zawartych w Context Help dla obiektu Format Date / Time String. Zmienić dane konfiguracyjne i zaobserwować, jak zostanie przedstawiony nowy stempel czasu. Nie wykonywać zapisu zmian w programie.
17. Zamknąć wszystkie aktywne okna środowiska LabVIEW.

6.3.3. Monitorowanie parametrów technicznych – zastosowanie węzłów właściwości

Cel zadania: poznanie mechanizmów tworzenia i zastosowania w programach węzłów właściwości jako elementów służących do bieżącej (zgodnej z wymaganiami programu) zmiany ustawień wybranych właściwości elementów umieszczonych na front panelu. Przedstawienie technik lepszej komunikacji w układzie HMI – użytkownik.

Zakres zadania: zamieszczanie w kodzie programu węzłów właściwości. Wykorzystanie pliku z procedurą automatu do generowania nazw pliku o określonej strukturze.

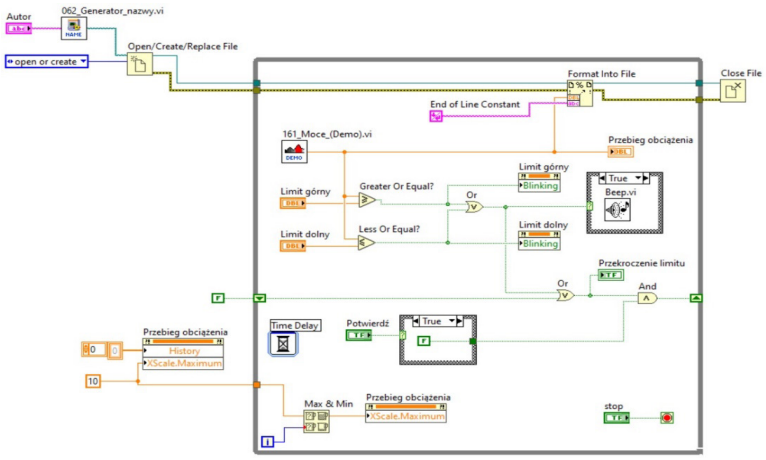
1. **Modyfikacja panelu czołowego programu.** Uruchomić program LabVIEW i otworzyć znajdujący się na dysku plik 061_Przebieg_mocy.vi. Bieżące zadanie będzie bazowało na rozszerzeniu istniejącego programu.
2. Zmodyfikować panel czołowy. Wstawić kontrolkę tekstową String Control i zadeklarować jej wartość domyślną na „Kowalski". Dodać wskaźnik liczbowy (Digital Display) do wyświetlacza „Przebieg czasowy" (patrz rysunek).



3. Zapisać program pod nową nazwą 063_Rejestrator_mocy.vi.

4. **Wprowadzanie modyfikacji do schematu blokowego.** Kolejne czynności będą prowadzone na istniejącym już schemacie blokowym. Przed przystąpieniem do następnych działań należy powiększyć odpowiednio obszar pętli While. Wprowadzone modyfikacje umożliwią automatyczną generację nazwy pliku do zapisu danych pomiarowych oraz dostosują właściwości wyświetlacza do warunków zadania (zmiana zakresu osi x) i migania kontrolek limitów po przekroczeniu zadanych wartości.

5. Należy odszukać i wstawić do schematu blokowego następujący plik: 062_Generator_nazwy.vi (polecenie „Select a VI…” w palecie Functions). Jest to plik z gotowym programem do generowania nazwy programu.
6. Z zakładki Comparison palety Functions wybrać moduł Min & Max.
7. Wygenerować węzeł właściwości obiektu wykresu „Przebieg obciążenia”. Aby go utworzyć, należy wskazać kursorem na obszar ikony wykresu „Przebieg obciążenia” i prawym przyciskiem myszy wywołać menu kontekstowe obiektu. Z menu kontekstowego wybrać polecenie Create /Property Node i kolejno wybrać z listy opcję History. Następnie należy rozszerzyć wygenerowany węzeł właściwości o kolejne pole i zdefiniować je jako XScale/Range/Maximum. Wybierając z menu kontekstowego węzła właściwości wykresu „Przebieg obciążenia” funkcję Change All To Write, zmienić sposób funkcjonowania węzła.
8. Wygenerować węzeł właściwości obiektu wykresu „Przebieg czasowy” z właściwością XScale/Range/Maximum. Wybierając z menu kontekstowego węzła właściwości wykresu „Przebieg obciążenia” funkcję Change All To Write, zmienić sposób funkcjonowania węzła.
9. Wygenerować węzeł właściwości obiektu kontrolki liczbowej „Limit dolny” z właściwością Blinking. Wybierając z menu kontekstowego węzła właściwości kontrolki numerycznej „Limit dolny” funkcję Change All To Write, zmienić sposób funkcjonowania węzła.
10. Wygenerować węzeł właściwości obiektu kontrolki liczbowej „Limit górny” z właściwością Blinking. Wybierając z menu kontekstowego węzła właściwości kontrolki numerycznej „Limit górny” funkcję Change All To Write, zmienić sposób funkcjonowania węzła.
11. Do terminala History w podwójnym węźle właściwości obiektu wykresu „Przebieg czasowy” wygenerować stałą z ustawieniami zerowymi. Najlepiej wykorzystać funkcję Create z menu kontekstowego terminala węzła właściwości.
12. Wstawić stałą liczbową Numeric Constant o wartości 10.
13. Rozmieścić zamieszczone elementy w sposób analogiczny do zaprezentowanego na rysunku i wykonać wymagane połączenia w schemacie blokowym.



14. Przed przystąpieniem do kolejnych czynności zapisać aktualną wersję programu pod bieżącą nazwą (063_Rejestrator_mocy.vi).

15. **Uruchomienie aplikacji i testowanie procedur.** Upewnić się, że program nie zawiera błędów i że jest aktywna ikona przycisku Run. Przejść do okna front panelu.
16. W kontrolce „Limit dolny" wprowadzić wartość 35, a w kontrolce „Limit górny" ustawić wartość 100.
17. Uruchomić program. W momencie uruchomienia programu wyświetli się okno dialogowe, w którym pojawi się drzewo katalogowe oraz przedstawiona wygenerowana automatycznie propozycja nazwy pliku do zapisu danych. Po zaakceptowaniu pliku i lokalizacji będą tam dostępne zarchiwizowane dane pomiarowe.
18. Zwrócić uwagę na funkcjonowanie bieżącego programu. Wstawione do algorytmu programu węzły właściwości umożliwiają:

- automatyczny reset historii przedstawianych na wyświetlaczu „Przebieg obciążenia" danych,
- wyświetlanie danych na wyświetlaczu od 0 i dopasowanie zakresu prezentowanych na wykresie danych do wszystkich punktów pomiarowych (automatyczne rozszerzanie maksimum wartości osi X),
- sygnalizację przekroczeń ustawionych limitów przez miganie pola danych w kontrolkach numerycznych.

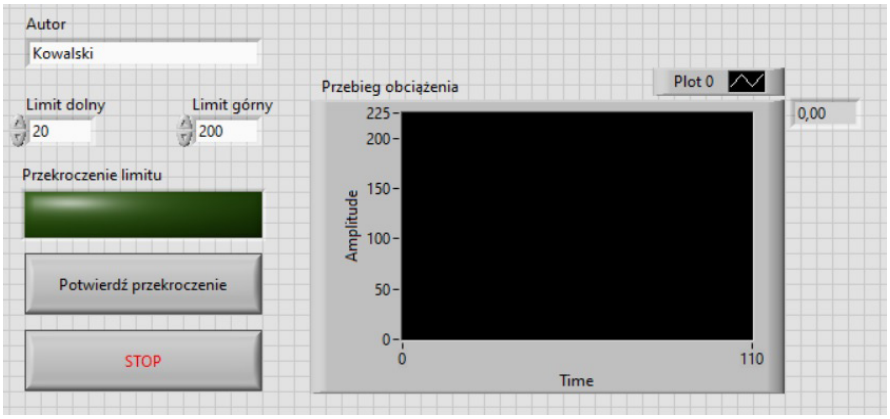
19. Zatrzymać funkcjonowanie programu przyciskiem [STOP].
20. Odszukać plik z zapisanymi informacjami. Otworzyć plik, wykorzystując aplikację Notatnik. W pliku powinny być zamieszczone wszystkie punkty pomiarowe.
21. Uruchomić kilkukrotnie program. Przed każdym uruchomieniem wprowadzać nowe nazwy do kontrolki tekstowej Autor. Po zakończeniu testowania programu należy zwrócić uwagę na nazwy wygenerowanych plików z danymi.
22. Zamknąć wszystkie aktywne okna środowiska LabVIEW i okno programu Notatnik.

6.3.4. Monitorowanie parametrów technicznych – rejestrator zdarzeń

Cel zadania: wprowadzenie modyfikacji do istniejącego programu. Program będzie umożliwiał zapis do pliku danych tylko tych pomiarów, dla których będą występowały przekroczenia limitów. Dodatkowo wraz z wartością danych będzie zapisana data, godzina oraz informacja o aktualnych limitach. Tak przygotowane procedury wykorzystuje się do tworzenia raportów w procesach technologicznych i systemach zarządzania.

Zakres zadania: użycie struktur wyboru do ograniczenia liczb zapisywanych danych do pliku. Przetwarzanie danych plikowych i tworzenie modelu rejestratora wybranych zdarzeń.

1. **Modyfikacja panelu czołowego programu.** Uruchomić program LabVIEW i otworzyć znajdujący się na dysku program 063_Rejestrator_mocy.vi. Bieżące zadanie będzie polegało na dokonaniu modyfikacji istniejącego programu.
2. Panel czołowy programu nie będzie podlegał zmianom i powinien mieć postać zbliżoną do przedstawionego na rysunku.

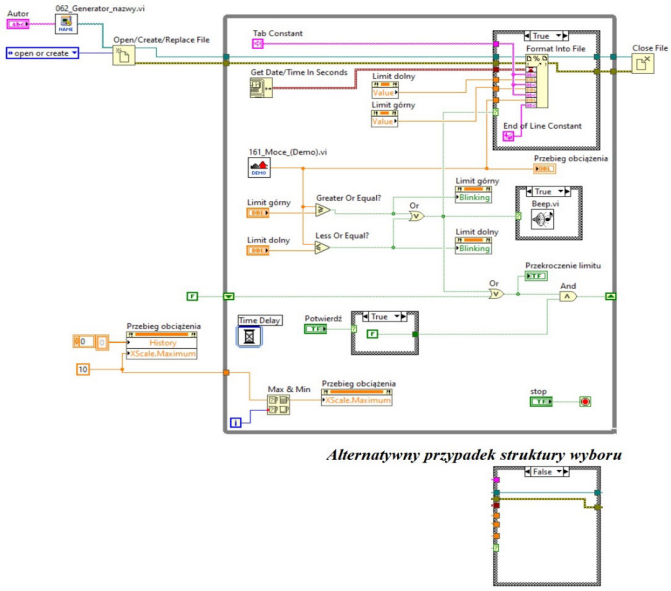


3. Zapisać program pod nową nazwą 064_Archiwum_przekroczeń.vi.
4. Kolejne czynności będą prowadzone na istniejącym już schemacie blokowym. Przed przystąpieniem do następnych działań powiększyć odpowiednio obszar pętli While.
5. Wstawić do wnętrza obszaru pętli trzecią strukturę wyboru i umieścić w niej element Format Into File.
6. Rozszerzyć liczbę terminali wejścia danych modułu Format Into File do 8. Do terminali tych będą podłączone kolejne sygnały z danymi do zapisu lub formatowania danych w pliku.

7. Uzupełnić schemat blokowy o wymienione poniżej elementy i dokonać stosownych połączeń. Po wykonaniu założonych modyfikacji schemat blokowy powinien być analogiczny do przedstawionego na kolejnym rysunku.
8. Używając elementów dostępnych w palecie Functions, wstawić do schematu blokowego:

- moduł Get Date / Time in Seconds (Programming → Timing),
- stałą tekstową Tab Constant (Programing → String).

9. Wygenerować węzeł właściwości obiektu kontrolki liczbowej „Limit dolny” z właściwością Value. Wybierając z menu kontekstowego węzła właściwości kontrolki numerycznej „Limit dolny” funkcję Change All To Write, zmienić sposób funkcjonowania węzła.



10. Wygenerować węzeł właściwości obiektu kontrolki liczbowej „Limit górny” z właściwością Value. Wybierając z menu kontekstowego węzła właściwości kontrolki numerycznej „Limit górny” funkcję Change All To Write, zmienić sposób funkcjonowania węzła.
11. Sprawdzić poprawność wykonanych połączeń. Zapisać dokonane zmiany w programie w pliku pod bieżącą nazwą (064_Archiwum_przekroczeń.vi).

12. **Uruchomienie aplikacji i testowanie.** Upewnić się, że program nie zawiera błędów i że jest aktywna ikona przycisku Run. Przejść do okna front panelu.
13. W kontrolce „Limit dolny” ustawić wartość 30, a w kontrolce „Limit górny” wartość 120.

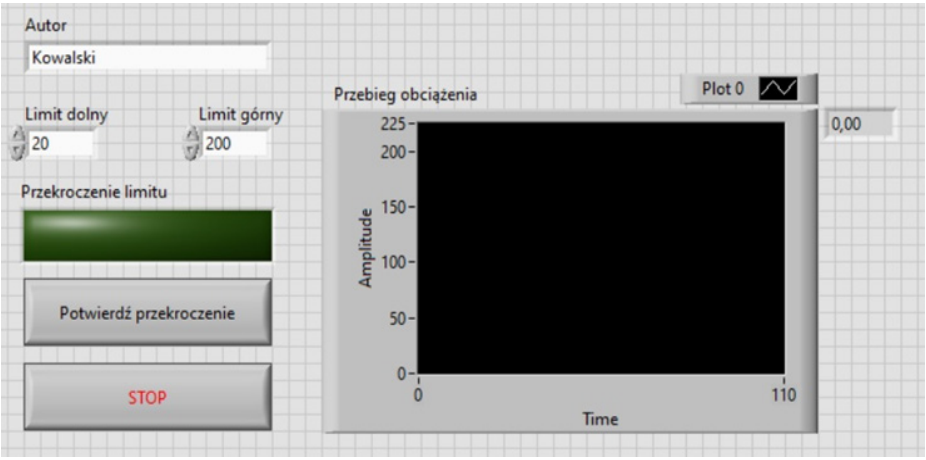
14. Uruchomić program. W momencie uruchomienia programu wyświetli się okno dialogowe, w którym pojawi się drzewo katalogowe oraz przedstawiona wygenerowana automatycznie propozycja nazwy pliku do zapisu danych. Po zaakceptowaniu w takim pliku i w takiej lokalizacji będą dostępne zarchiwizowane dane pomiarowe.
15. Odczekać pewien czas, obserwując dane wyświetlane na wykresie oraz zachowanie się poszczególnych elementów front panelu. W przypadku sygnalizacji przekroczenia limitu na wskaźniku LED, użyć przycisku „Potwierdzenie przekroczenia".
16. Zatrzymać funkcjonowanie programu przyciskiem [STOP].
17. Odszukać plik z zapisanymi informacjami. Otworzyć plik, wykorzystując aplikację Notatnik. W pliku powinny być zamieszczone dane związane z odczytami na temat wystąpienia przekroczenia (godzina, data, ustawienia limitów oraz odczyt).
18. Uruchomić kilkukrotnie program. Przed uruchomieniem wprowadzić nowe dane do kontrolki tekstowej Autor oraz kontrolek limitów. Podczas działania programu zmieniać wartości limitów. Po zakończeniu testowania programu należy zwrócić uwagę na informacje zapisane w plikach z danymi.
19. Zamknąć wszystkie aktywne okna środowiska LabVIEW i okno programu Notatnik.

6.3.5. Monitorowanie parametrów technicznych – dokumentacja programu

Cel zadania: nabycie umiejętności dodawania informacji o programie (dokumentacja funkcjonowania programu, odsyłacze do strony www, np. z dodatkowymi informacjami od producenta). Dodany zostanie opis informacyjny o funkcjonowaniu VI. Dzięki temu użytkownik, wstawiając program do nowego projektu, będzie miał możliwość zapoznania się z jego właściwościami.

Zakres zadania: praktyczne wykorzystanie funkcji związanych z tworzeniem dokumentacji VI dostępnych w VI Properties. Konfigurowanie i wprowadzanie danych w oknie kategorii Documentation. Prezentacja informacji o VI w oknie pomocy kontekstowej (Context Help).

1. **Czynności wstępne.** Uruchomić program LabVIEW i otworzyć znajdujący się na dysku program 064_Archiwum_przekroczeń.vi. Bieżące zadanie będzie polegało na dokonaniu zmian z użyciem funkcji wspomagających w programie.
2. Panel czołowy programu nie będzie podlegał zmianom i powinien mieć postać zbliżoną do przedstawionego na rysunku.

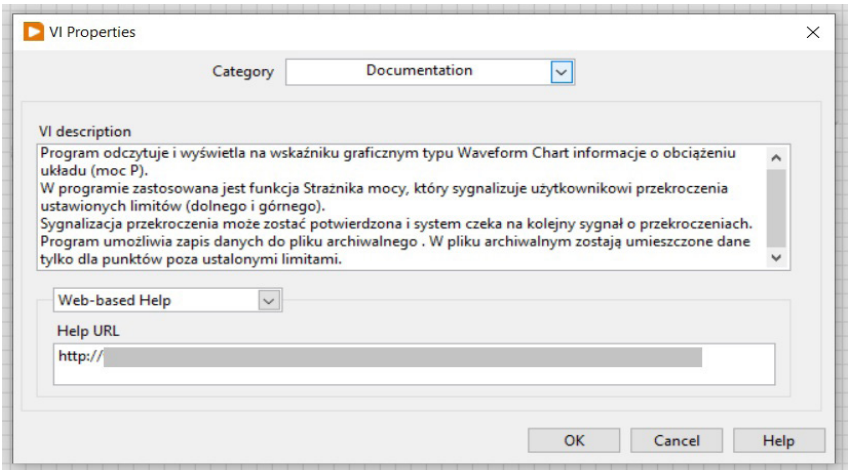


3. Zapisać program pod nową nazwą 065_Dokumentacja.vi.

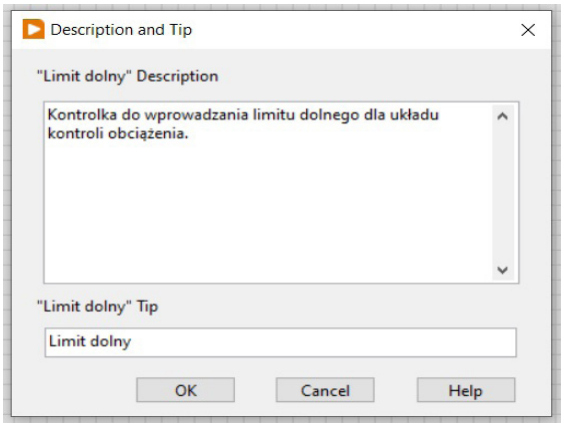
4. **Wprowadzanie dokumentacji użytkownika.** Z menu File wybrać polecenie VI Properties (taki sam efekt można uzyskać, naciskając kombinację klawiszy [Ctrl] + [I]). Uaktywnione zostanie okno VI Properties.
5. Uzupełnieniu będzie podlegało okno: VI description, lista rozwijana z funkcjami o dostępności informacji oraz okno Help URL.
6. W pole VI description wprowadzić poniższy tekst:

- „Program odczytuje i wyświetla na wskaźniku graficznym typu Waveform Chart informacje o obciążeniu układu (moc P).
- W programie zastosowana jest funkcja Strażnika mocy, który sygnalizuje użytkownikowi przekroczenia ustawionych limitów (dolnego i górnego).
- Sygnalizacja przekroczenia może zostać potwierdzona i system czeka na kolejny sygnał o przekroczeniach.
- Program umożliwia zapis danych do pliku archiwalnego. W pliku archiwalnym zostają umieszczone dane tylko dla punktów poza ustalonymi limitami".

7. Z rozwijanej listy źródła wybrać opcję Web-based Help.
8. Do okna Help URL wpisać link do źródła pomocy. W tym celu w zasobach Internetu należy znaleźć dowolną stronę o przebiegach zmienności obciążenia i skopiować jej adres.
9. Rezultat wprowadzonych zmian powinien być zgodny z rysunkiem. Wprowadzone zmiany w oknie VI Properties i kategorii Documentation zatwierdzić, wybierając przycisk [OK].



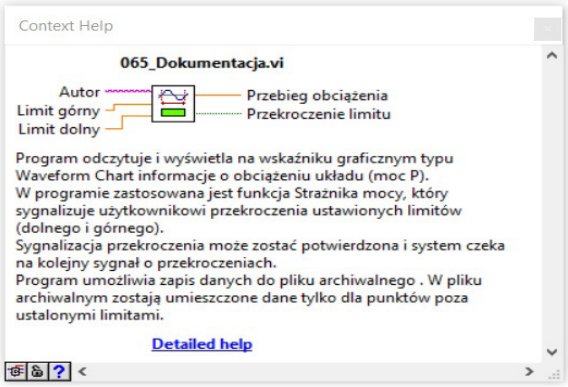
10. Następnie na front panelu odszukać kontrolkę numeryczną „Limit dolny” i uaktywnić jej menu kontekstowe. W menu kontekstowym wybrać polecenie Description and Tip. Otworzy się okno dialogowe Description and Tip.
11. W oknie „Limit dolny” Description wpisać tekst:
„Kontrolka do wprowadzania limitu dolnego dla układu kontroli obciążenia”.
12. W oknie „Limit dolny” Tip wprowadzić tekst „Limit dolny”.
13. Wynik wykonanych czynności powinien być tożsamy z przedstawionym na rysunku.



14. W podobny sposób uzupełnić dane dotyczące pozostałych kontrolek i wskaźników zamieszczonych na front panelu programu.
15. Wprowadzone zmiany właściwości programu w VI Properties zapisać w pliku pod bieżącą nazwą (065_Dokumentacja.vi).

16. **Tworzenie ikony i deklarowanie terminali we/wy.** Uaktywnić okno panelu czołowego i otworzyć narzędzie edytora ikon. Za pomocą dostępnych w uaktywnionym oknie narzędzi należy stworzyć ikonę, która będzie charakteryzowała budowany przyrząd.
17. Następnie wykonać układ konektorów terminali wejść i wyjść dla projektowanego przyrządu wirtualnego. W tworzonym przyrządzie wirtualnym wejściem jest kontrolka tekstowa „Autor", kontrolki liczbowe „Limit dolny" i „Limit górny", a wyjściami są wykres „Przebieg obciążenia" oraz dioda „Przekroczenie limitu".
18. Zamknąć aktywne okna programu.

19. **Kontrola wprowadzonych ustawień.** Otworzyć nowy czysty plik (Blank VI) środowiska LabVIEW i przejść do edycji schematu blokowego.
20. Do okna schematu blokowego wstawić plik 065_Dokumentacja.vi (paleta Functions, zakładka „Select a VI...").
21. Uruchomić pomoc kontekstową (Context Text z menu Help lub [Ctrl] + [H]). Wskazując myszą wstawiony SubVI zapoznać się z informacjami wyświetlanymi w oknie pomocy kontekstowej. W oknie pomocy kontekstowej powinny zostać wyświetlone dane o VI (patrz rysunek).



22. W oknie pomocy kontekstowej kliknąć hiperłącze Detailed help. Wynikiem tego działania jest uruchomienie przeglądarki internetowej ze stroną wpisaną w dokumentacji VI.
23. Wstawić do okna schematu blokowego inne (dowolne) obiekty z palet Functions i zbadać możliwości menu kontekstowego dotyczące informacji o terminalach we/wy, zastosowaniu obiektu, odsyłaczy do pliku pomocy oraz z zamieszczonymi przykładowymi zastosowaniami w postaci plików Exercise.
24. Zamknąć wszystkie okna programu LabVIEW, nie dokonując zapisu w żadnym z użytych programów.

6.3.6. Monitorowanie parametrów technicznych – zabezpieczanie programów przed edycją i przed dostępem do kodu źródłowego

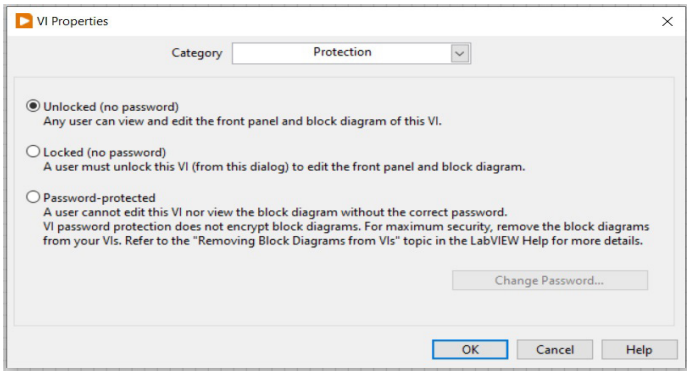
Cel zadania: testowanie możliwości edycji programów zabezpieczonych przed przypadkowym lub celowym dostępem.

Zakres zadania: użycie funkcji związanych z tworzeniem zabezpieczeń programu dostępnych w VI Properties. Konfigurowanie i wprowadzanie danych w oknie kategorii Protection.

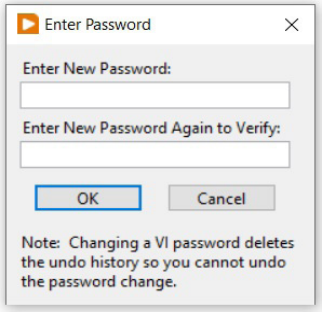
1. **Czynności wstępne.** Uruchomić środowisko LabVIEW i otworzyć znajdujący się na dysku program 065_Dokumentacja.vi. Zapisać plik pod nową nazwą 066_Protection_1.vi.
2. Program nie będzie podlegał żadnym zmianom pod względem edycji front panelu i schematu blokowego.

3. **Zabezpieczanie programu przed dostępem i modyfikacjami.** W trakcie ćwiczenia zostaną utworzone programy, w których zaimplementowane będą pewne środki ochrony przed przypadkowymi lub celowymi zmianami. Takie zabezpieczenia mogą być stosowane w przypadku przekazywania programów do testowania współpracownikom lub prezentowania gotowych rozwiązań zewnętrznym kontrahentom. W zależności od ustawionej opcji możliwe jest ograniczenie opcji edycyjnych lub zabezpieczenie kodu źródłowego przed inwigilacją.
4. Z menu File wybrać polecenie VI Properties lub wcisnąć kombinację klawiszy [Ctrl] + [I]. Uaktywnione zostanie okno VI Properties. W rozwijanym menu Category wybrać opcję Protection, wyświetli się wtedy okno analogiczne z przedstawionym na rysunku. W oknie tym użytkownik ma trzy możliwości do wyboru:

- Unlocked (no password) – Any user can view and edit the front panel and block diagram of this VI, czyli każdy użytkownik może przeglądać i edytować front panel i schemat blokowy tego VI bez żadnych przeszkód,
- Locked (no password) – A user must unlock this VI (from this dialog) to edit the front panel and block diagram, czyli przed edycją front panelu i schematu blokowego użytkownik musi zdjąć zabezpieczenie tego VI z poziomu tego okna dialogowego. Odblokowanie polega na zmianie ustawionej opcji z Locked na Unlocked,
- Password-protected – A user cannot edit this VI nor view the block diagram without the correct password, czyli przed wprowadzeniem jakichkolwiek zmian użytkownik musi podać hasło, hasłem blokowany jest również podgląd schematu blokowego. Odblokowanie dostępu wymaga wpisania prawidłowego hasła. Jego podanie jest konieczne także przy zmianach poziomu zabezpieczenia programu.



5. Wybrać opcję Locked (no password), a wprowadzoną zmianę zatwierdzić przyciskiem [OK]. Należy zwrócić uwagę, że wygląd okna front panelu się zmienił.
6. Zamknąć wszystkie aktywne okna programu LabVIEW.
7. Odnaleźć na dysku program 066_Protection_01 i otworzyć go.
8. Zaobserwować zmiany w możliwościach edycyjnych programu po ustawieniu opcji Locked (no password). Program obecnie nie podlega edycji i nie można wprowadzić żadnych zmian w schemacie blokowym, a na front panelu można tylko zmieniać ustawienia kontrolek i innych elementów sterujących.
9. Z menu File wybrać polecenie VI Properties lub kombinację klawiszy [Ctrl] + [I]. Uaktywnione zostanie okno VI Properties. W rozwijanym menu Category wskazać opcję Protection, następnie wybrać domyślną wartość ustawień Unlocked (no password), a wprowadzoną zmianę zatwierdzić przyciskiem [OK]. Zwrócić uwagę, że po wykonaniu tej instrukcji program znowu jest w pełni edytowalny.
10. Bieżącą postać programu zapisać pod nową nazwą 066_Protection_2.vi.
11. Z menu File wybrać polecenie VI Properties lub kombinację klawiszy [Ctrl] + [I]. Uaktywnione zostanie okno VI Properties. W rozwijanym menu Category wskazać opcję Protection. Po zadeklarowaniu tej kategorii działania, wybrać Password-protected. Po zaznaczeniu tej opcji wygenerowane zostanie okno dialogowe (patrz rysunek), w którym należy wprowadzić hasło ochronne do programu.



12. Wprowadzić dowolne hasło dla programu i nacisnąć przycisk [OK]. Wybrać przycisk [OK] również w oknie dialogowym VI Properties.
13. Zamknąć wszystkie aktywne okna programu LabVIEW.
14. Odszukać na dysku program 066_Protection_02 i otworzyć go.
15. Zaobserwować zmiany w możliwościach edycyjnych programu po ustawieniu opcji Password-protected.
16. Program obecnie nie podlega edycji i na front panelu można zmieniać tylko ustawienia kontrolek i innych elementów sterujących. Otwarcie okna schematu blokowego możliwe jest tylko po wprowadzeniu zadeklarowanego wcześniej hasła.
17. Po wprowadzeniu hasła, użytkownik otrzymuje ponownie pełen dostęp do narzędzi edycyjnych.
18. Zamknąć wszystkie aktywne okna programu, bez dokonywania zmian w programie.

6.4. Pytania kontrolne

1. W jaki sposób konfiguruje się wprowadzenie ochrony panelu czołowego przed przypadkowym i umyślnym dostępem?
2. Jak opisać strukturę klastra błędu i znaczenie jego stosowania w programach?
3. Jaki efekt będzie miało zadeklarowanie funkcjonowania panelu czołowego w każdym z trybów, tj. default, top level application, dialog oraz custom?
4. Jak scharakteryzować obiekt węzła właściwości pod kątem sposobu wprowadzania do kodu programu i potencjalnych efektów jego działania?
5. Jak opisać rolę struktur wyboru w tworzeniu reguł progowych?
6. Jakie są znane sposoby reakcji na wykryte „nieprawidłowości" w zakresie funkcjonowania programu i otrzymywanych wyników?

7. Organizacja panelu systemu diagnostycznego i obsługa za pomocą klawiatury

7.1. Cel zajęć

Ta część podręcznika skupia się na dwóch zagadnieniach. Pierwszym z nich jest przedstawienie problemów występujących na ekranach aplikacji systemów diagnostycznych. Oryginalnym, specjalistycznym, dedykowanym programom towarzyszą niedoskonałości wyglądu i obsługi ekranu użytkownika, które zazwyczaj nie występują w popularnych programach komercyjnych tworzonych przez zespoły programistów. Ten rozdział zawiera wskazówki i praktyczne rady pozwalające na uniknięcie problemów przy obsłudze interfejsu użytkownika. Drugim opisywanym zagadnieniem jest wspomaganie obsługi aplikacji poprzez wykorzystanie klawiatury. Mimo że zarówno środowisko LabVIEW, jak i opracowane w nim systemy mają charakter graficzny, to wprowadzenie obsługi systemu za pomocą klawiatury może znacznie usprawnić pracę, a nawet wprowadzić możliwość obsługi systemu w przypadku awarii urządzenia wskazującego (myszy).

Celem zajęć jest praktyczne zapoznanie z:

- czynnikami wpływającymi na komfort obsługi panelu aplikacji pomiarowej,
- implementacją rozwiązań prowadzących do efektywnego korzystania z ekranu systemu pomiarowego, takimi jak: sterownie rozmiarem okna systemu, organizacja obiektów na ekranie,
- menu opartym na bazie klastra,
- dynamicznym sterowaniem zachowaniem i wyglądem obiektów za pomocą węzłów właściwości,
- poruszaniem się pomiędzy obiektami panelu oraz uruchamianiem funkcji za pomocą klawiatury,
- zwiększaniem efektywności pracy ze środowiskiem LabVIEW dzięki wykorzystaniu skrótów klawiaturowych.

7.2. Wstęp

7.2.1. Planowanie panelu – wprowadzenie

Systemy komputerowe opracowywane w środowisku LabVIEW są zarówno opracowywane, jak i użytkowane w środowisku graficznym. Łącznikiem pomiędzy użytkownikiem a systemem pomiarowym jest graficzny interfejs użytkownika (GUI – graphical user interface). W środowisku LabVIEW interfejs graficzny użytkownika dostarcza panel czołowy (front panel).

W trakcie opracowywania interfejsu użytkownika należy pamiętać o wielu czynnikach, między innymi o takich jak:

- wygląd interfejsu może decydować o wyborze aplikacji przez użytkownika,
- nowy wygląd i rodzaj interfejsu będzie wymagał od użytkownika nauki jego zasad obsługi,
- interfejs wpływa na łatwość użytkowania programu i szybkość jego obsługi, co oddziałuje na oszczędność czasu przy pracy z całą aplikacją. W krytycznym przypadku użytkownicy będą unikać lub odmawiać korzystania z aplikacji,
- źle zaprojektowany interfejs będzie komplikował obsługę programu, wymagał dodatkowego czasu na zapoznanie się z nim, a w granicznym przypadku uniemożliwiał przeprowadzenie działań, do których został przeznaczony łącznie z generowaniem błędów,
- biegłość w obsłudze środowiska programistycznego nie jest jednoznaczna z biegłością w opracowaniu interfejsu.

W celu zapewnienia wysokiej funkcjonalności interfejsu z punktu widzenia użytkownika, zarówno projektant, jak i programista powinni uwzględniać, że użytkownicy [3, 7]:

- mogą być w różnym wieku (w tym mogą być to dzieci i osoby starsze),
- posiadają różny poziom wykształcenia i kwalifikacji w obszarach związanych z użytkowanym systemem,
- mogą realizować różne zadania za pomocą tego samego interfejsu,
- posiadają zróżnicowane preferencje i uprzedzenia w odniesieniu do poznanych interfejsów.

Istnieje możliwość zdefiniowania ogólnych zasad projektowania interfejsu graficznego, które można zgrupować w poniższe kategorie [3, 6, 8]:

- wygląd interfejsu – powinien występować wyraźny podział na obszary dedykowane wykonywaniu zróżnicowanych funkcji,
- świadoma eksploracja interfejsu – interfejs powinien dostarczać użytkownikowi informacji na temat tego, w którym miejscu znajdują się elementy panelu i co one oznaczają,
- wysoka estetyka – powinna zostać zachowana równowaga pomiędzy czytelnością i funkcjonalnością,
- uwzględnienie doświadczenia – powinna być zapewniona możliwość obsługi interfejsu tak przez początkującego, jak i zaawansowanego użytkownika,
- spójność obsługi – użytkownik powinien mieć możliwość przewidywania i oszacowania efektów działania wybranych przez siebie funkcji (w zakresie: sposobu obsługi interfejsu, barwy, czcionki, menu, przycisków, skrótów, rozmieszczenia elementów na ekranie, wyrównywania, informacji dla użytkownika),
- zminimalizowanie wysiłku – ograniczanie liczby operacji prowadzących do osiągnięcia celu,
- zasada zbliżenia do użytkownika – użytkownik nie powinien mieć trudności z przystosowaniem do interfejsu,

- potwierdzanie działań destrukcyjnych – operacje skutkujące np. usuwaniem obiektów (np. plików, kont) powinny być potwierdzane przez użytkownika,
- zapewnienie odwracalności działań – użytkownik powinien mieć możliwość przywrócenia stanu sprzed wykonywanej operacji.

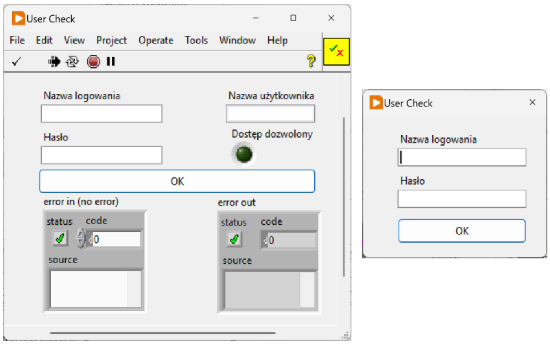
Mimo dużego nakładu pracy i pozytywnego podejścia do podstawowych zasad projektowania interfejsu graficznego, możliwe jest opracowanie nieefektywnego interfejsu, co wynika najczęściej z tego, że:

- główny nacisk został położony na aspekty techniczne programu, przy nieuwzględnieniu w dostatecznym stopniu wagi interfejsu,
- zastosowano bezpłatne lub niskonakładowe oprogramowanie, co skutkowało opracowaniem trudnego w użyciu interfejsu,
- zrozumienie systemu przez projektanta i programistę było inne niż rozumienie systemu przez użytkownika końcowego,
- niektóre technologie mogły ograniczać funkcjonalność interfejsu,
- przy opracowywaniu funkcjonalności interfejsu użytkownika kierowano się intuicją, a jednocześnie zrezygnowano z fazy badań i testów,
- projektanci i programiści podejmowali arbitralne decyzje dotyczące interfejsu użytkownika, mimo braku doświadczenia w opracowywaniu podobnych systemów.

7.2.2. Elementy panelu czołowego – ograniczanie widoku

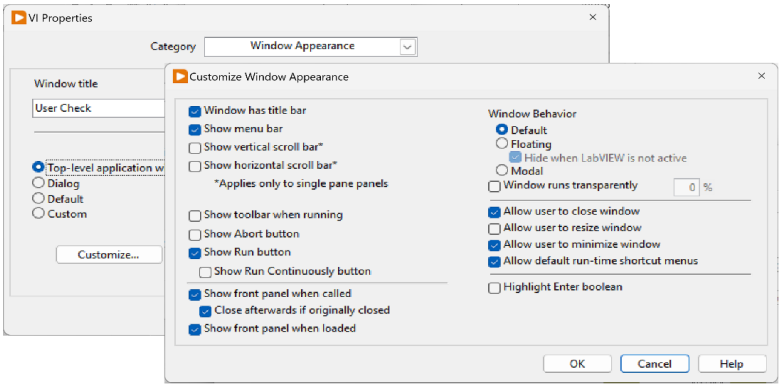
Jednym z czynników oceny interfejsu graficznego, a zarazem całej aplikacji, jest procent popełnianych błędów w trakcie wykonywania zadania. Poprawę efektywności można uzyskać przez ograniczenie możliwości popełnienia błędu, co wiąże się z ograniczeniem liczby obiektów panelu do obiektów niezbędnych do obsługi programu (rys. 7.1). Elementy, które nie są prezentowane na panelu, nie rozpraszają, uniemożliwiają popełnienie błędu, a jednocześnie ułatwiają obsługę systemu [12, 13, 15].

Poza obiektami świadomie prezentowanymi na panelu (rozmiar okna, umieszczenie poza obszarem wyświetlania) można ograniczać także typowe elementy okna systemu operacyjnego, takie jak menu, paski przewijania, przyciski okna.



Rys. 7.1. Minimalizacja elementów w oknie programu [16]

Precyzyjne dopasowanie wyświetlanych elementów okna dostępne jest przez okno dialogowe właściwości aplikacji (File /VI Properties /Window Appearance /Custom / Customize Window Appearance). Szybkie usunięcie wszystkich zbędnych elementów można uzyskać przez zadeklarowanie prezentacji okna jako dialogowego (File / VI Properties /Window Appearance /Dialog) (rys. 7.2).

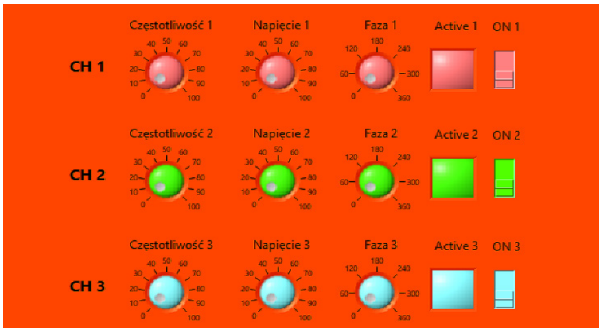


Rys. 7.2. Okno dialogowe konfiguracji wyświetlanych elementów okna systemu

7.2.3. Elementy panelu czołowego – barwy

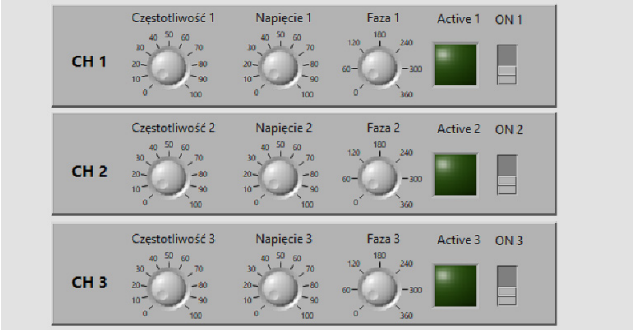
Głównym czynnikiem wpływającym zarówno na estetykę, jak i łatwość korzystania z aplikacji jest stosowany schemat kolorystyczny. Istnieje duża grupa zaleceń w tym zakresie. Zalecane jest, aby:

- wykorzystywane kolory były stonowane, nie powinny męczyć oka,
- kolory były do siebie dopasowane (np. niebieski i zielony, czerwony i żółty itp.) lub kontrastowe: biały i czarny,
- kolor tła nie był zbyt intensywny (rys. 7.3), tło musi być spokojne, najlepiej jednolite, ewentualnie ze stopniowymi przejściami,



Rys. 7.3. Nadmierna paleta kolorów zastosowana do grupowania obiektów

- uwzględnić schemat kolorów, który wpływa na nastrój użytkownika. Niebieski jest bardziej „prestiżowym" kolorem (podobnie jak większość chłodnych kolorów),
- stosować różne kolory do oznaczenia zmian stanów systemu. Tam, gdzie to możliwe, powiązać kolory z realizacją zadań, tzn. zadania podobne powinny mieć podobne kolory,
- korzystać z domyślnych kolorów (zapewnienia to obsługę w każdym systemie operacyjnym),
- uważać na kombinacje kolorów, które mogą powodować szybkie zmęczenie oczu (np. czerwony z niebieskim),



Rys. 7.4. Grupowanie obiektów bez stosowania barw

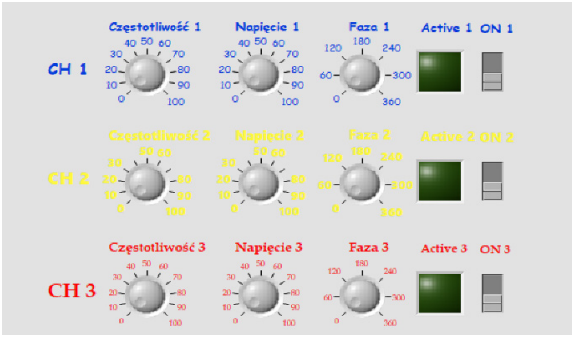
- ograniczać liczbę stosowanych barw. Zaleca się zaczynać ze standardowym szarym i korzystać z jednego–dwóch odcieni,
- wyróżniać ważne elementy: przebiegi wykresów, przyciski [STOP],
- stosować intensywniejszy kolor dla mniejszych elementów, które wymagają wyróżnienia,
- tam, gdzie to możliwe, zamiast koloru używać odstępów i grupowania obiektów (rys. 7.4),
- wzorować się na panelach przyrządów rzeczywistych, o ile takie istnieją.

7.2.4. Panel systemowy – porządkowanie, grupowanie, czcionki

Porządkowanie i grupowanie elementów jest drugim podstawowym czynnikiem wpływającym zarówno na estetykę, jak i łatwość korzystania z aplikacji. Podobnie jak w przypadku barw, istnieją zalecenia dotyczące porządkowania i grupowania obiektów [13, 14]:

- do oddzielania i separowania obiektów stosować przerwy (odstępy) i wyrównania,
- wyrównanie do lewej (lub prawej) funkcjonuje lepiej niż wyśrodkowanie,
- obiekty można grupować za pomocą elementów dekoracyjnych,
- nie stosować wymuszonego grupowania przy użyciu zmian kroju czcionek (może to dawać wrażenie bałaganu),

- nie stosować niewielkich zmian rozmiarów czcionek – może to wyglądać jak błędne formatowanie. Maksymalnie cztery rozmiary czcionki,
- należy unikać stosowania wielu różnych czcionek (rys. 7.5). Maksymalnie trzy kroje czcionek w całym systemie. Zdecydowanie unikać stosowania czcionek ozdobnych.

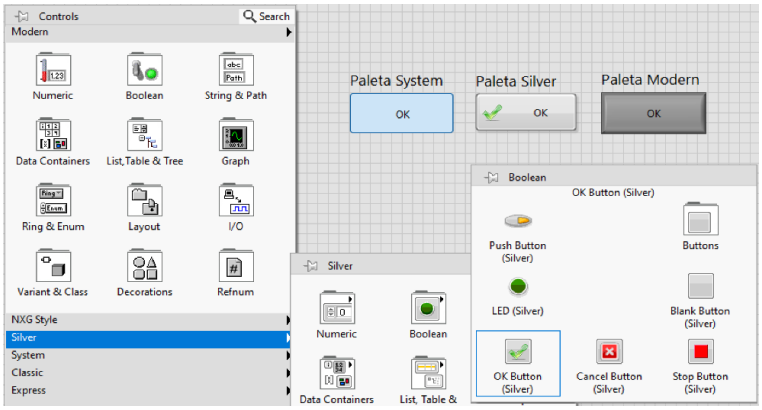


Rys. 7.5. Nieudana próba grupowania obiektów poprzez zmianę kroju i barwy czcionki

7.2.5. Efektywne wykorzystanie panelu

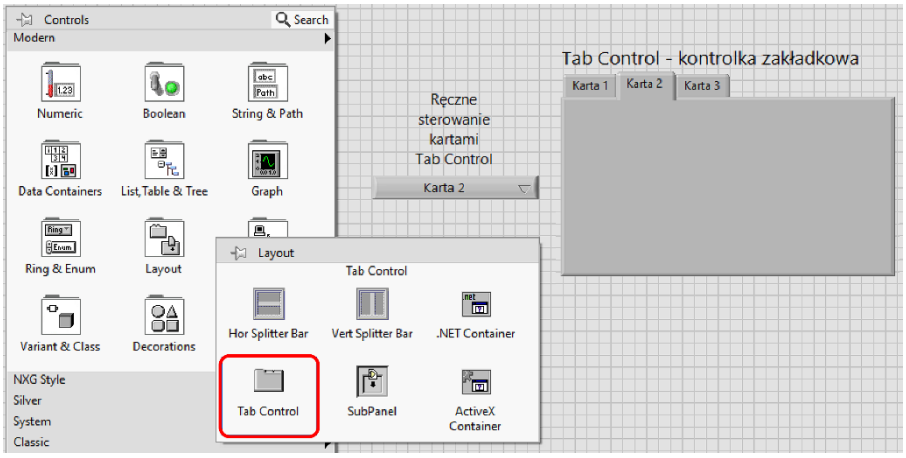
Wśród obiektów i narzędzi, które pozwalają na efektywne wykorzystanie powierzchni panelu czołowego systemu pomiarowego, należy wymienić kontrolki (obiekty) systemowe, kontrolki zakładkowe oraz grupę obiektów dekoracyjnych.

Kontrolki (obiekty) systemowe umieszczone na subpalecie Controls /System dostarczane są w celu zapewnienia spójnej komunikacji z użytkownikiem za pomocą okien dialogowych (rys. 7.6). Stosowanie tego typu obiektów zapewnia, że w trakcie wyświetlania komunikatów, obiekty systemowe będą dostosowywały swoją barwę i kształt do standardowych kontrolek okien dialogowych systemu operacyjnego, na którym użytkownik uruchomił aplikację.



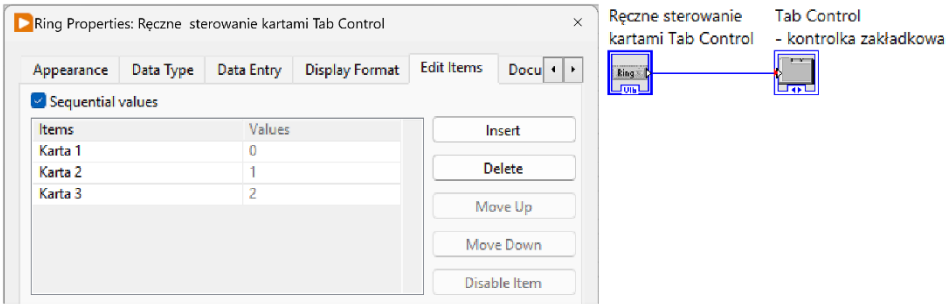
Rys. 7.6. Subpaleta System wraz z porównaniem przycisku Stop w trzech stylach

Kolejnym obiektem pozwalającym na efektywne wykorzystanie powierzchni panelu czołowego jest kontrolka zakładkowa (subpaleta Containers /Tab Control) (rys. 7.7). Dzięki niej można powiększać powierzchnię dostępną w jednym oknie aplikacji poprzez „nakładanie" na siebie kolejnych kart kontrolki zakładkowej.



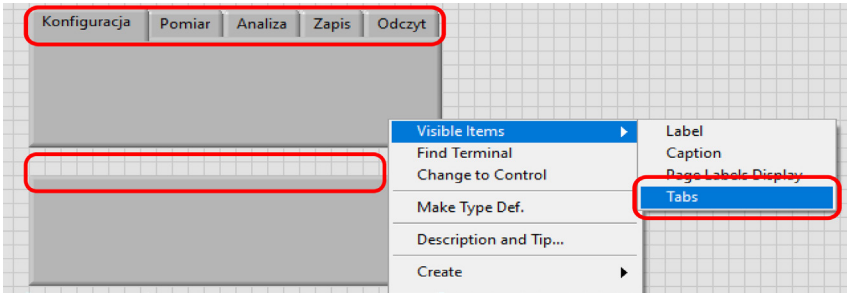
Rys. 7.7. Lokalizacja Tab Control na paletach panelu

Sterowanie kartami kontroli zakładkowej realizowane jest poprzez zmienną liczbową (typ liczby całkowite). Dzięki temu można w dość prosty sposób programowo lub poprzez interakcję z użytkownikiem wybrać kartę z dostępnymi obiektami (rys. 7.8).



Rys. 7. 8. Sterowanie Tab Control za pomocą menu Ring

Ciekawy efekt można uzyskać poprzez ukrycie kart kontrolki zakładkowej (rys. 7.9). W ten sposób można dynamicznie „wymieniać ekrany" dostępne w pojedynczym oknie aplikacji bez zmiany konfiguracji pozostałych elementów panelu.



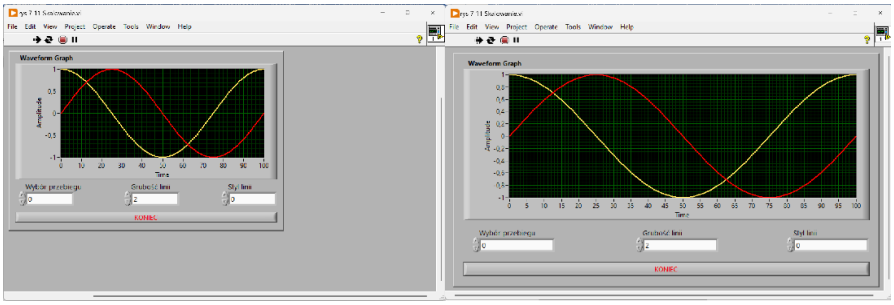
Rys. 7.9. Tab Control z ukrytymi nazwami kart

Subpaleta Decorations występuje na kilku paletach głównych. W zależności od palety nadrzędnej ma różną zawartość. Obiekty z palety Decorations nie mają swojego odwzorowania na schemacie blokowym. Głównym celem stosowania obiektów z tej palety jest:

- separacja,
- grupowanie obiektów za pomocą figur geometrycznych,
- zamieszczanie opisów,
- graficzne wiązanie fizycznie współpracujących elementów (rys. 7.10),
- pomoc użytkownikowi w obsłudze interfejsu.

Rys. 7.10. Grupowanie obiektów za pomocą elementów dekoracyjnych

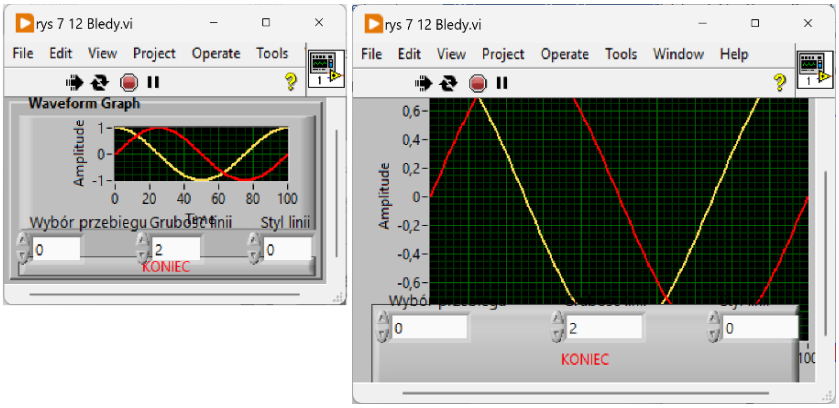
W celu maksymalnego wykorzystania całej powierzchni ekranu monitora, sugerowana jest maksymalizacja powierzchni okna. Jednocześnie należy pamiętać, że rozmiary obiektów panelu nie są bezpośrednie związane z rozmiarem okna. Za automatyczną maksymalizację okna oraz automatyczne skalowanie obiektów panelu odpowiadają tzw. właściwości aplikacji. Są one dostępne w dwóch kategoriach File /VI Properties /Window Size oraz Window Run-Time Position. Najbardziej elastyczne dopasowanie rozmiaru obiektów uzyskuje się za pomocą opcji Scale all objects on front panel as the window resizes (rys. 7.11). Opcja oddziałuje na wszystkie obiekty graficzne na panelu czołowym.



Rys. 7.11. Efekt zmiany rozmiaru okna programu przy braku skalowania i ze skalowaniem

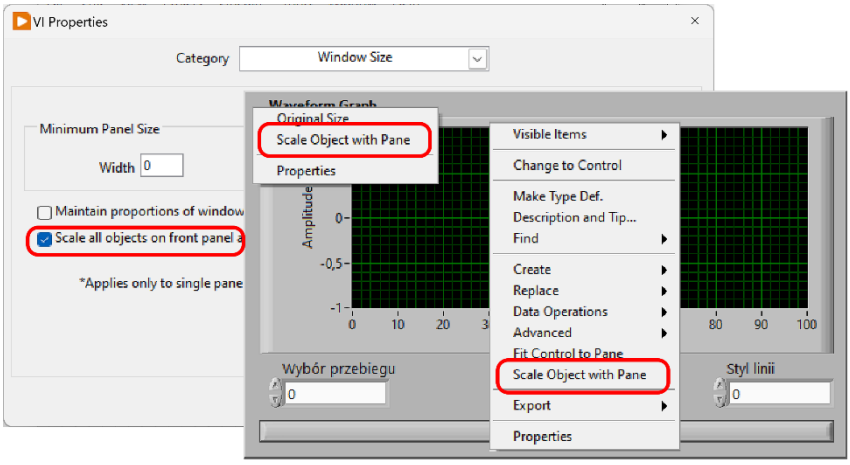
Stosowanie opcji Scale all objects on front panel as the window resizes wymaga uwzględnienia dwóch czynników:

- Rozmiar czcionki nie podlega automatycznemu skalowaniu. Można uzyskać efekt skalowania przez umieszczanie podpisów jako elementów graficznych lub zastosować programowe skalowanie czcionki za pomocą węzłów właściwości.
- Nadmierne zminimalizowanie okna powoduje zniekształcenie położenia obiektów panelu, co skutkuje problemami z wyświetlaniem przy powiększeniu rozmiaru (rys. 7.12). Temu efektowi można zapobiegać przez zadeklarowanie minimalnego rozmiaru okna, w którym elementy wyświetlane są poprawnie.



Rys. 7.12. Efekt nadmiernego ograniczenia rozmiaru okna

Poza opcją Scale all objects on front panel as the window resizes, dotyczącą wszystkich elementów graficznych panelu, istnieje możliwość indywidualnego skalowania pojedynczych obiektów (rys. 7.13). Takie zachowanie deklarowane jest za pomocą menu kontekstowego obiektu i polecenia Scale object with Pane.

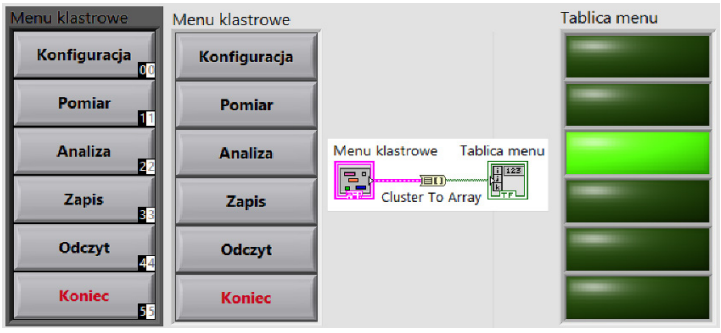


Rys. 7.13. Deklaracja skalowania wszystkich obiektów oraz pojedynczego obiektu

7.2.6. Menu klastrowe

Kolejnym obiektem, który wprowadza wyraźny podział funkcyjny panelu dla użytkownika, a zarazem ułatwia opracowanie kodu, jest menu klastrowe. Menu klastrowe jest pojedynczym obiektem łączącym wiele przycisków logicznych w pojedynczy klaster. Z punktu widzenia użytkownika występuje prostota wyboru wielu funkcji w jednym miejscu panelu (operacji, działań), zaś z punktu widzenia programisty mamy do czynienia z obiektem, który generuje dane pozwalające na łatwe rozgałęzianie kodu.

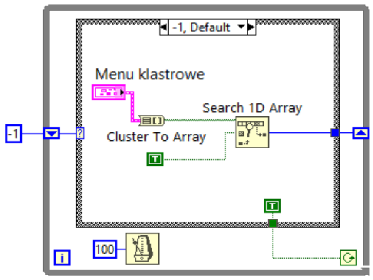
Pozorną trudnością jest praca z klastrem, który w swej istocie magazynuje dane różnych typów. W przypadku menu klastrowego typ danych nie ma znaczenia, ponieważ wykorzystuje się wartość wynikającą z kolejności w klastrze. Liczba odzwierciadlająca wybór użytkownika, a zarazem wybrany element klastra, konwertowana jest na postać tablicową (Cluster To Array) (rys. 7.14). Dzięki przeszukiwaniu tablicy (Search 1D Array) można podjąć decyzję o dalszym wykonywaniu kodu programu.



Rys. 7.14. Idea pozyskiwania danych z klastra stanowiącego menu

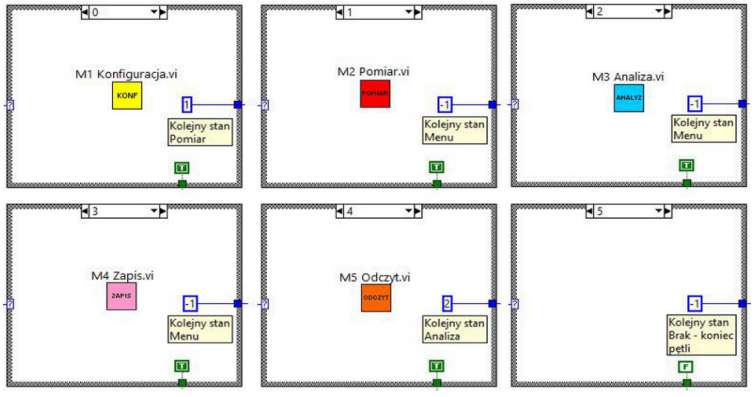
Połączenie menu klastrowego z programem opartym na architekturze maszyny stanu skutkuje uniwersalną strukturą, w której w kompaktowy sposób można elastycznie sterować kolejnymi podprogramami (procedurami). Takie kompaktowe i elastyczne rozwiązanie uzyskuje się przez połączenie (rys. 7.15 i 7.16):

- menu klastrowego – pozwalającego na dokonanie prostego wyboru przez użytkownika aplikacji,
- węzła Cluster To Array – konwertującego stan menu klastrowego do postaci macierzy, którą można analizować za pomocą funkcji tablicowych,
- węzła Search 1D Array – identyfikującego, która z pozycji menu klastrowego została wybrana,



Rys. 7.15. Aplikacja oparta na maszynie stanu sterowana menu klastrowym [14]

- pętli While – pozwalającej na ciągłą pracę programu,
- struktury wyboru – grupującej operacje (podprogramy), odpowiadające poleceniom reprezentowanym przez menu,
- rejestru przesuwnego – pozwalającego na elastyczne przechodzenie pomiędzy podprogramami (przypadkami struktury wyboru). Pętla While może zostać wyposażona w dodatkowe komórki rejestru przesuwnego, służące do przekazywania danych pomiędzy „przypadkami" struktury wyboru.



Rys. 7.16. Procedury aplikacji opartej na maszynie stanu ze sterowaniem menu klastrowym

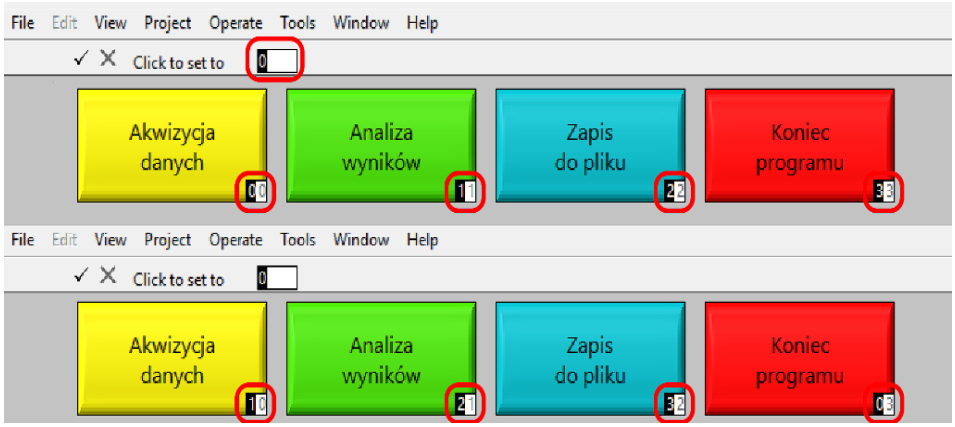
7.2.7. Obsługa obiektów panelu za pomocą klawisza tabulacji

Domyślne ustawienie wybierania elementów panelu czołowego za pomocą tabulatorów polega na tym, że kolejne naciśnięcia klawisza tabulacji powoduje wybór kolejnego elementu. Elementy wybierane są w takiej kolejności, w jakiej zostały zamieszczone na panelu. Kombinacja klawiszy Shift + Tab powoduje odwrócenie tej kolejności.

Jednocześnie za pomocą klawiatury można zmieniać stan wybranego elementu. W najprostszym przypadku jest to zmiana stanu logicznego elementów wejściowego na przeciwny, przy użyciu klawisza Spacji i Enter. W przypadku kontrolek liczbowych można modyfikować wartość dzięki klawiszom kierunkowym.

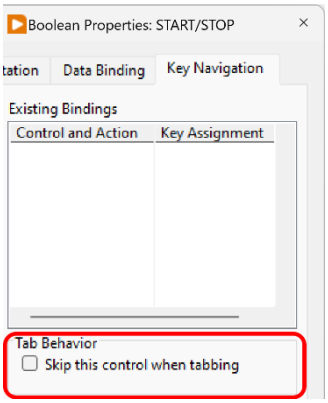
Domyślne ustawienie wybierania kolejnych elementów za pomocą klawisza tabulacji można modyfikować poprzez polecenie Set Tabbing Order z menu Edit. Okno konfiguracji następstwa wybierania elementów klawiszem Tab jest identyczne, jak w przypadku konfigurowania kolejności zmiennych w klastrze (rys. 7.17).

Towarzyszące obiektom liczby, znajdujące się na białym tle, wskazują na aktualny porządek wybierania. Wybór elementu za pomocą kursora myszy nadaje odpowiadającą kolejności liczbę, wyświetlaną w górnej części okna. Dwa dodatkowe pola wyboru (check mark i anuluj) pozwalają na zatwierdzenie lub anulowanie wprowadzonych zmian.



Rys. 7.17. Dwie konfiguracje kolejności wybierania elementów za pomocą klawisza Tab

Możliwe jest wykluczenie obiektu z selekcji za pomocą klawisza Tab. Taka opcja dostępna jest na karcie Key Navigation arkusza właściwości pomijanego obiektu. Opcja pozwalająca na wykluczenie nosi nazwę Skip this control when tabbing (rys. 7.18).



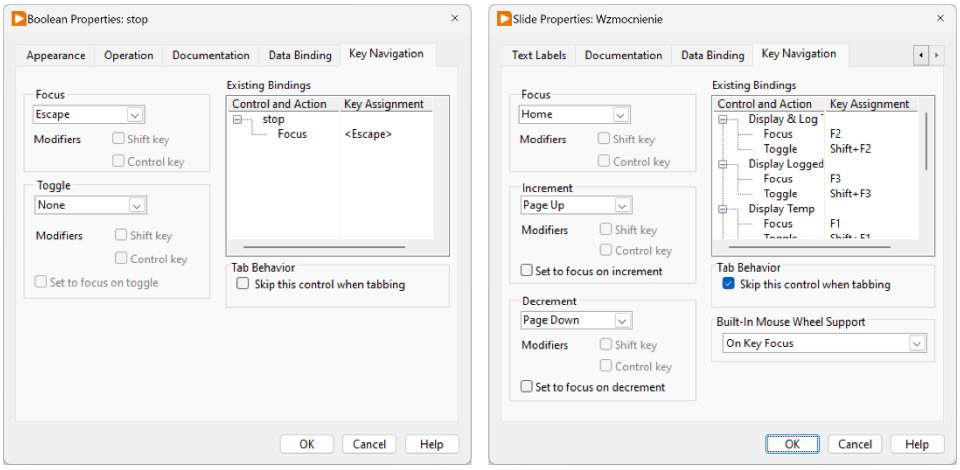
Rys. 7.18. Wykluczanie wybierania elementów za pomocą klawisza Tab

7.2.8. Indywidualizacja aplikacji – klawisze skrótów obiektów

Poza domyślnym sekwencyjnym wybieraniem elementów za pomocą klawisza Tab oraz zmiany stanu za pomocą klawiszy Spacji, Enter i kierunkowych, możliwe jest deklarowanie dla konkretnych obiektów stanów:

- wskazania elementu (Focus),
- zmiany stanu (np. przełączenia, Toggle).

Deklaracja klawisza skrótu prowadzona jest na karcie Key Navigation arkusza właściwości obiektu. Karta Key Navigation posiada także opcję pozwalająca na wykluczenie obiektu z obsługi za pomocą tabulatora (Skip this control when tabbing) (rys. 7.19). Występuje tutaj ograniczenie możliwych do wykorzystania klawiszy głównie do klawiszy funkcyjnych.

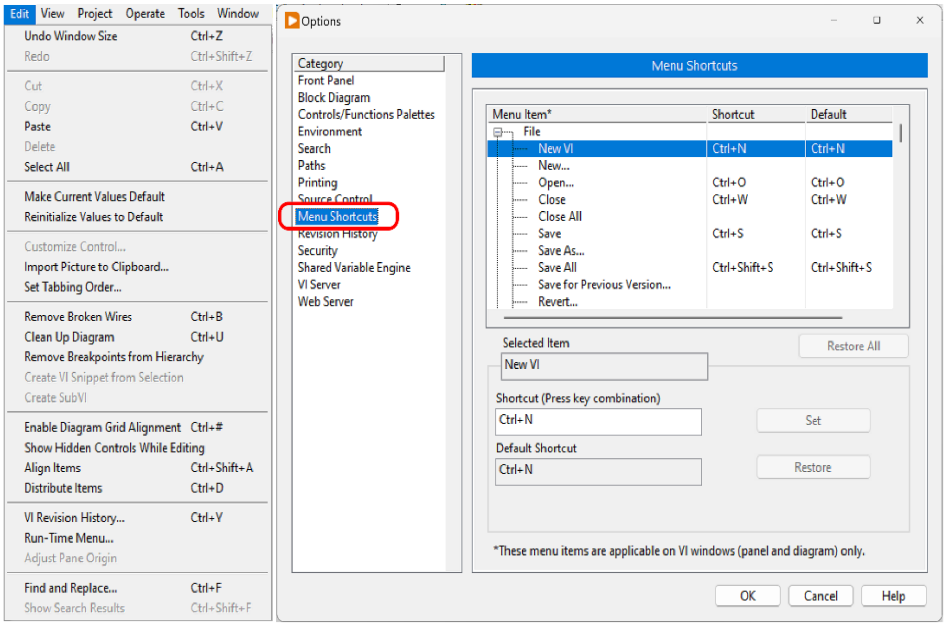


Rys. 7.19. Arkusze właściwości dla obiektów binarnych i numerycznych

7.2.9. Skróty klawiaturowe środowiska programowania

Podobnie jak w typowych aplikacjach graficznych systemu operacyjnego Windows, środowisko LabVIEW posiada możliwość przyśpieszania pracy poprzez stosowanie skrótów klawiaturowych. Z jednej strony są to typowe skróty spotykane w innych aplikacjach takie jak: Nowy plik – Ctrl + N, Otwórz – Ctrl + O, Zapisz – Ctrl + S, Drukuj – Ctrl + P. Z drugiej strony istnieją kombinacje klawiszy predefiniowane do obsługi LabVIEW, takie jak Remove Brokern Wires – Ctrl + B, Clean Up Diagram – Ctrl + U, Uruchom – Ctrl + R.

Istnieje możliwość modyfikowania istniejących skrótów, wprowadzania nowych skrótów dla poleceń menu oraz odtwarzania domyślnej konfiguracji klawiszy skrótów. Takie możliwości udostępnia okno dialogowe Menu Shortcuts. Dostęp do niego można uzyskać przez menu Tools /polecenie Advanced i wybór kategorii Menu Shortcuts (rys. 7.20).



Rys. 7.20. Menu LabVIEW wraz z oknem konfiguracyjnym Menu Shortcuts

W zależności od charakteru prowadzonych działań (operacje na plikach, operacje na obiektach panelu i schematu blokowego, edycja tekstu, nawigacja w oknach itp.) skróty klawiaturowe i operacje przy użyciu kursora myszy uzyskują różną wagę. Do skrótów klawiaturowych (i operacji z wykorzystaniem myszy i klawiatury) w środowisku LabVIEW, które należałoby pamiętać i stosować, należą te zamieszczone w tabeli 7.1.

Tab. 7.1. Wybrane skróty klawiaturowe środowiska LabVIEW

Skrót	Funkcja
Ctrl + E	Przełączanie pomiędzy oknami panelu czołowego i schematem blokowym
Ctrl + T	Wyświetlanie okien panelu i schematu obok siebie na całej powierzchni ekranu
Ctrl + H	Wyświetlanie okna pomocy kontekstowej
Ctrl + R	Uruchamianie programu (Run)
Ctrl + .	Przerywanie wykonywania programu (Abort)
Ctrl + B	Usuwanie ze schematu blokowego niekompletnych połączeń (Remove Broken Wires)
Pojedyncze kliknięcie myszą	Wybranie pojedynczego prostego odcinka połączenia
Podwójne kliknięcie myszą	Wybranie całej gałęzi połączenia
Potrójne kliknięcie myszą	Wybranie całej grupy połączeń posiadających wspólne dane
Spacja	Zmiana kierunku prowadzenia połączenia
Ctrl + B	Automatyczna organizacja (oczyszczanie) schematu (Cleanup Diagram)
Ctrl + L	Wyświetlanie okna listy błędów

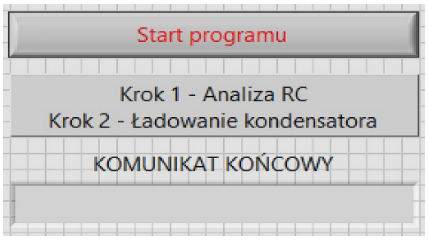
7.3. Zadania

7.3.1. Deklarowanie aktywności okien podprogramów

Cel zadania: zapoznanie ze sposobem wykonywania podprogramów zamieszczonych w kodzie programu głównego.

Zakres zadania: podprogramy programu głównego mogą być wykonywane zarówno w tle, jak i mogą prezentować swój panel czołowy. W przypadku podprogramów, w których użytkownik powinien wykazać aktywność (np. zadeklarować wartości zmiennych, wskazać miejsce zapisu pliku itp), konieczna jest deklaracja, że taki program zostanie wywołany w trybie pierwszoplanowym. Ćwiczenie dotyczy praktycznej deklaracji (w tym wymuszania) wybranego sposobu wywołania podprogramu.

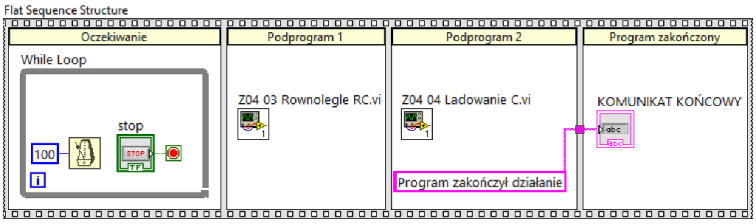
1. **Budowa panelu czołowego**. Program będzie działał w czterech etapach. Pierwszy etap to oczekiwanie na aktywność użytkownika, drugi etap – wywołanie podprogramu do symulacji działania obwodu równoległego RC, trzeci etap – wywołanie podprogramu do symulacji ładowania kondensatora, czwarty etap – wyświetlenie komunikatu o zakończeniu działania.
2. Uruchomić środowisko LabVIEW z pustym plikiem programu. Zbudować panel czołowy zaprezentowany na rysunku.



3. Panel zawiera:
- przycisk Stop (Controls /Modern /Boolean oraz Stop Button). Komunikat przycisku zmodyfikować na „Start programu”,
- etykietę z opisem działania programu (Controls /Modern /Decorations oraz Label). Wprowadzić tekst etykiety na „Krok 1 – Analiza RC. Krok 2 – Ładowanie kondensatora”. Za pomocą belki narzędziowej Text Settings zadeklarować wyrównanie tekstu do środka,
- kontrolkę wyjściowej zmiennej łańcuchowej String Indicator (Controls /Modern / String & Path oraz String Indicator). Zmienić nazwę zmiennej na „KOMUNIKAT KOŃCOWY”. Za pomocą belki narzędziowej Text Settings zadeklarować wyrównanie tekstu do środka.

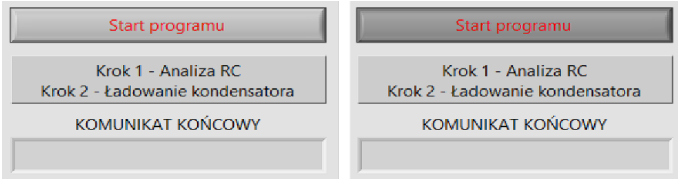
4. Zapisać plik programu pod nazwą (071_Sposob_dzialania.vi) w lokalizacji wskazanej przez prowadzącego.

5. **Schemat blokowy programu**. Przejść do okna schematu blokowego. Aktualnie kod zawiera jedynie dwa elementy, które są ikonami obiektów umieszczonych na panelu aplikacji.
6. Zamieścić płaską strukturę sekwencyjną sterującą kolejnością wykonywania podprogramów (Functions /Programming /Structures /Flat Sequence Structure). Poszerzyć strukturę do czterech ramek za pomocą menu kontekstowego struktury Add Frame After /Before.
7. W pierwszej ramce struktury sekwencyjnej zamieścić pętlę While, która generuje kod oczekiwania na wybranie przez użytkownika przycisku „Start programu”, co powoduje przejście do drugiej ramki struktury sekwencyjnej (Functions / Programming /Structures /While Loop).
8. Do pętli wprowadzić węzeł opóźnienia Wait Until Next ms Multiple. Deklarację stałej liczbowej należy przeprowadzić za pomocą menu kontekstowego węzła (menu kontekstowe /Create /Constant /wprowadzenie oczekiwanej wartości).
9. Dodać stałą łańcuchową String Constant (Programming /String oraz String Constant). Do stałej przypisać komunikat „Program zakończył działanie”.
10. Poprzez paletę Programming i polecenie Select a VI zamieścić w kolejnych ramkach struktury wyboru opracowane wcześniej podprogramy:
- (Z04 03 Rownolegle RC.vi),
- (Z04 04 Ladowanie C.vi).



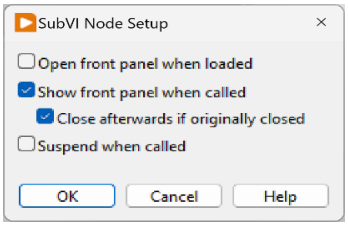
11. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
12. Zapisać plik programu pod aktualną nazwą (071_Sposob_dzialania.vi).

13. **Uruchamianie programu**. Uruchomić program w trybie jednokrotnym (Run).
14. Wybrać przycisk „Start programu".



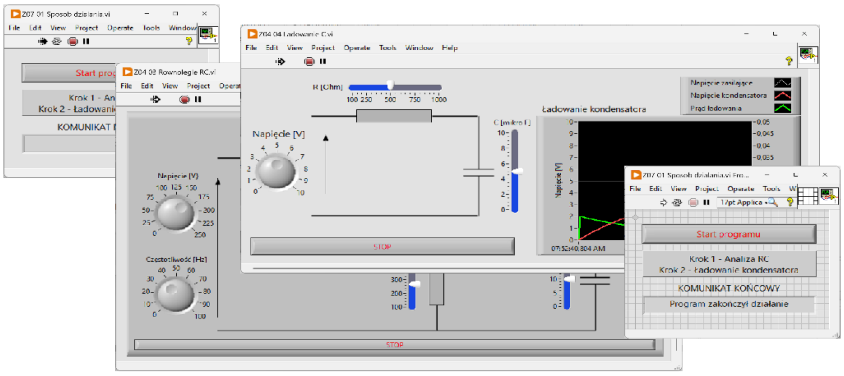
15. Zwrócić uwagę, że mimo wielokrotnego wybierania przycisku, brak jest efektów polegających na możliwości modyfikowania zmiennych podprogramów i wyświetlania wyników.
16. Zakończyć działanie programu przyciskiem Abort Execution.
17. Przejść do schematu blokowego programu głównego.
18. W odniesieniu do obu podprogramów (043_Rownolegle_RC.vi, 044_Ladowanie_C.vi) wykonać konfigurację za pomocą menu kontekstowego ikon podprogramów i polecenia SubVI Node Setup:

- Show front panel when called,
- Close afterwards if originally closed.



19. Zapisać plik programu pod aktualną nazwą.
20. Uruchomić program w trybie jednokrotnym (Run).
21. Wybrać przycisk „Start programu".

22. Przeprowadzić krótkie obserwacje w oknach podprogramów. Zwrócić uwagę, że program główny zakończył działanie, wyświetlając komunikat „Program zakończył działanie”, wraz z zakończeniem drugiego z podprogramów.



23. Zapisać plik programu pod aktualną nazwą (Z07 01 Sposob dzialania.vi).
24. Zamknąć plik programu po zakończeniu testowania.

7.3.2. Praca bez myszy. Klawisze skrótów

Cel zadania: nabycie umiejętności deklaracji klawiszy skrótów dla ikon obiektów panelu czołowego reprezentujących wejściowe zmienne logiczne. Wprowadzenie do budowy aplikacji, które można obsługiwać bez konieczności korzystania z myszy.

Zakres zadania: opracowanie programu zawierającego cztery przyciski panelu czołowego, których zadaniem jest:

A) wyświetlanie komunikatu informacyjnego,

B) wywołanie programu do symulacji działania obwodu równoległego RC,

C) wywołanie podprogramu do symulacji ładowania kondensatora,

D) kończenie działania programu.

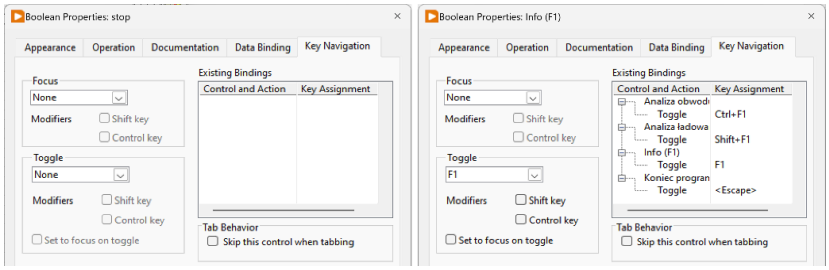
1. **Budowa panelu czołowego**. Uruchomić środowisko LabVIEW z pustym plikiem programu. Zbudować panel czołowy zaprezentowany na rysunku.
2. Panel zawiera cztery przyciski Stop (Controls /Modern /Boolean oraz Stop Button).
3. Zmodyfikować etykiety i teksty poszczególnych przycisków na:

- etykieta: „Info (F1)”, tekst przycisku „F1”,
- etykieta: „Analiza obwodu równoległego RC (Ctrl + F1)”, tekst przycisku „Ctrl + F1”,
- etykieta: „Analiza ładowania kondensatora (Shift + F1)”, tekst przycisku „Shift + F1”,
- etykieta: „Koniec programu (Esc)”, tekst przycisku „Esc”.

4. Za pomocą karty Key Navigation arkusza właściwości każdego z przycisków, zadeklarować klawisze skrótów dla następujących przycisków logicznych:

- F1,
- Ctrl + F1,
- Shift + F1,
- Esc.

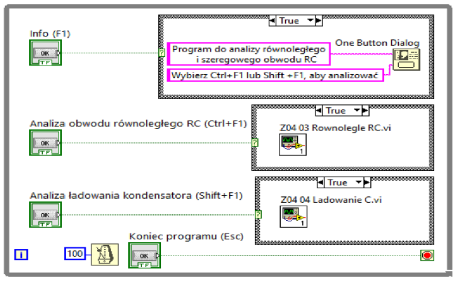
Dostęp do karty Key Navigation arkusza właściwości można uzyskać za pomocą menu kontekstowego przycisku i wybrania polecenia Properties lub Advanced /Key Navigation.



5. Zapisać plik programu pod nazwą (072_Bez_myszy.vi) w lokalizacji wskazanej przez prowadzącego.

6. **Schemat blokowy programu**. Przejść do okna schematu blokowego. Aktualnie kod zawiera ikony czterech przycisków logicznych stop.
7. Zamieścić pętlę While (Functions /Programming /Structures /While Loop). Pozawala ona na ciągłą pracę programu uruchomionego w trybie jednokrotnym.
8. W pętli zamieścić węzeł opóźnienia Wait Until Next ms Multiple. Deklarację stałej liczbowej (100 ms) należy przeprowadzić za pomocą menu kontekstowego węzła (menu kontekstowe /Create /Constant /wprowadzenie oczekiwanej wartości).
9. Zamieścić trzy struktury wyboru Case (Functions /Programming /Structures /Case Structure). Struktury te nie posiadają zawartości dla przypadku False (niewykonywany jest żaden kod). Wybór przycisku panelu czołowego generuje zmienną logiczną True jednej ze struktur wyboru i wykonanie zamieszczonego w strukturze kodu lub podprogramu.

10. W polu przypadku True struktury wyboru „Info (F1)” zamieścić okno dialogowe z jednym przyciskiem (Programming /Dialog & User Interface /One Button Dialog). Za pomocą menu kontekstowego (prawy przycisk myszy) zadeklarować teksty komunikatu (Program do analizy równoległego i szeregowego obwodu RC) i tytułu przycisku (Wybierz Ctrl + F1 lub Shift + F1, aby analizować).



11. Poprzez paletę Programming i polecenie Select a VI… zamieścić w kolejnych ramkach struktury wyboru opracowane wcześniej podprogramy:
- (043_Rownolegle_RC.vi),
- (044_Ladowanie_C.vi).
12. W odniesieniu do obu podprogramów (043_Rownolegle_RC.vi), (044_Ladowanie_C.vi) wykonać konfigurację za pomocą menu kontekstowego ikon podprogramów i polecenia SubVI Node Setup:
- Show front panel when called,
- Close afterwards if oryginally closed.
13. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu.
14. Zapisać plik programu pod aktualną nazwą (072_Bez_myszy.vi).

15. **Uruchamianie programu**. Usunąć mysz poza zasięg ręki, tak aby przypadkowo nie korzystać z niej przy obsłudze programu.
16. Uruchomić program w trybie jednokrotnym (Ctrl + R).
17. Aby wyświetlić komunikat informacyjny, wybrać F1. Aby zamknąć okno, wybrać klawisz Enter.
18. Aby wywołać program do symulacji działania obwodu równoległego RC, wybrać Ctrl + F1. Przeprowadzić krótką obserwację działania. Za pomocą klawisza Tab przejść do przycisku Stop podprogramu. Aby zamknąć okno podprogramu, wybrać klawisz Enter.
19. Aby wywołać program do symulacji ładowania kondensatora, wybrać Shift + F1. Przeprowadzić krótką obserwację działania. Za pomocą klawisza Tab przejść do przycisku Stop podprogramu. Aby zamknąć okno podprogramu, wybrać klawisz Enter.

20. Zakończyć działanie programu głównego klawiszem Esc.
21. Zamknąć plik programu głównego.

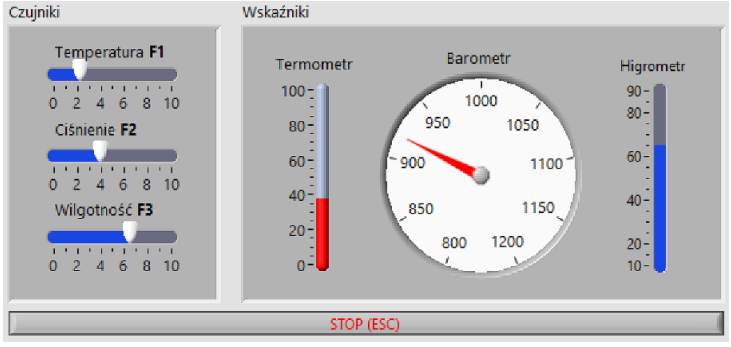
7.3.3. Klawisze skrótów w klastrach

Cel zadania: utrwalenie umiejętności deklaracji klawiszy skrótów dla elementów panelu czołowego, reprezentujących wejściowe zmienne numeryczne i logiczne. Rozbudowa aplikacji, obsługiwanej dotychczas wskaźnikiem myszy, o możliwość korzystania z klawiatury.

Zakres zadania: modyfikacja opracowanej uprzednio aplikacji wykorzystującej klastry i stałą klastrową o funkcję obsługi za pomocą klawiatury. Domyślna konfiguracja aplikacji pierwszoplanowej LabVIEW pozwala na wybieranie kolejnych elementów panelu za pomocą klawisza Tab. Takie działanie nie zawsze jest efektywne w przypadku występowania na panelu obiektów klastra. Przy wskazywaniu elementów za pomocą klawisza Tab, mogą być pomijane elementy umieszczone wewnątrz klastra. Bezpośrednie zdefiniowanie klawiszy skrótów dla elementów wewnątrz klastra usuwa to ograniczenie.

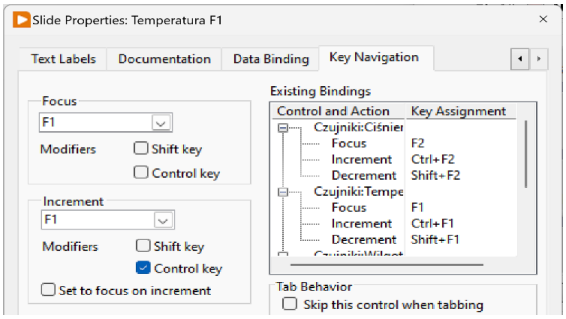
1. Modyfikacja panelu czołowego. Uruchomić w środowisku LabVIEW uprzednio opracowany program (037_Klaster_stala.vi). Zmodyfikować panel, aby wyglądał tak, jak zaprezentowany na rysunku.
2. Zmiany mają charakter kosmetyczny i polegają na uzupełnieniu etykiet elementów :

- z „Temperatura” na „Temperatura **F1**”,
- z „Ciśnienie” na „Ciśnienie **F2**”,
- z „Wilgotność” na „Wilgotność **F3**”,
- z „STOP” na „STOP (ESC)”.



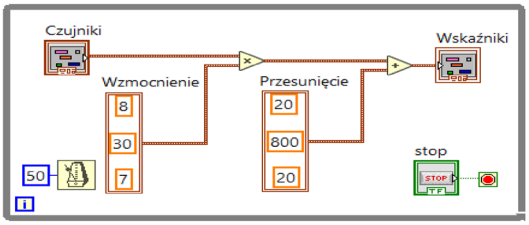
3. Za pomocą karty Key Navigation arkusza właściwości (lub menu kontekstowe / Advanced /Key Navigation…) każdego z elementów, zadeklarować klawisze skrótów:

- „Temperatura F1" – wybór (Focus) F1, zwiększanie wartości (Increament) Ctrl + F1, zmniejszanie wartości (Decreament) Shift + F1,
- „Ciśnienie F2" – wybór F2, zwiększanie wartości Ctrl + F2, zmniejszanie wartości Shift + F2,
- „Wilgotność F3" – wybór F3, zwiększanie wartości Ctrl + F3, zmniejszanie wartości Shift + F3,
- „STOP (ESC)" – przełączenie (Toggle) – klawisz Esc.



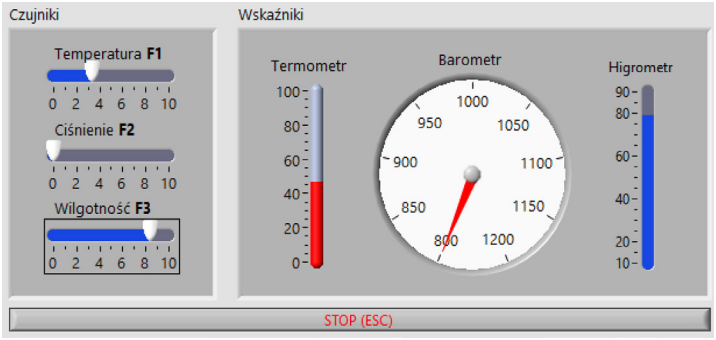
4. Zapisać plik programu pod nazwą (073_Bez_myszki.vi) w lokalizacji wskazanej przez prowadzącego.

5. **Schemat blokowy programu**. Przejść do okna schematu blokowego.
6. Zamieścić pętlę While (Functions /Programming /Structures /While Loop). Pozawala ona na ciągłą pracę programu uruchomionego w trybie jednokrotnym.
7. W pętli zamieścić węzeł opóźnienia Wait Until Next ms Multiple. Deklarację stałej liczbowej (50 ms) należy przeprowadzić za pomocą menu kontekstowego węzła (menu kontekstowe /Create /Constant /wprowadzenie oczekiwanej wartości).



8. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu.
9. Zapisać plik programu pod aktualną nazwą (073_Bez_myszki.vi).

10. **Uruchamianie programu**. Usunąć mysz poza zasięg ręki, tak aby przypadkowo nie korzystać z niej przy obsłudze programu.
11. Uruchomić program w trybie jednokrotnym (Ctrl + R).
12. Wybrać zmienną „Temperatura F1" za pomocą klawiatury (klawisz F1). Podjąć próbę zmiany wielkości wejściowej na mniejszą i większą za pomocą klawiszy kierunkowych góra/dół lub lewo/prawo.

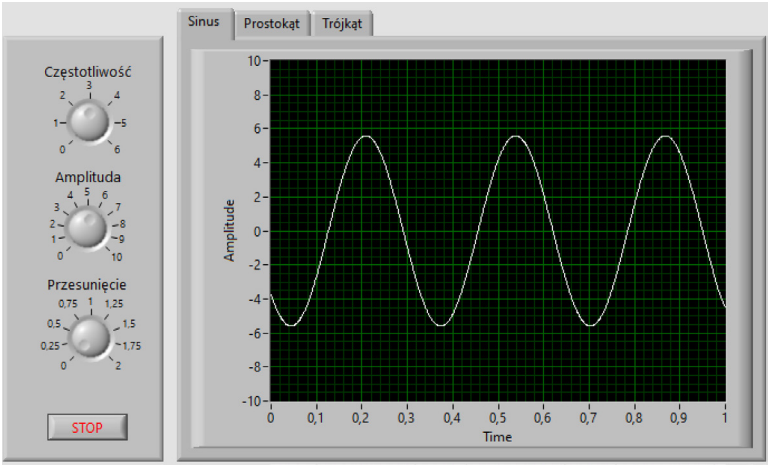


13. Wybrać zmienną „Ciśnienie F2" za pomocą klawiatury (klawisz F2). Podjąć próbę zmiany wielkości wejściowej na mniejszą i większą za pomocą klawiszy kierunkowych góra/dół lub lewo/prawo.
14. Wybrać zmienną „Wilgotność F3" za pomocą klawiatury. Podjąć próbę zmiany wielkości wejściowej na mniejszą i większą za pomocą klawiszy kierunkowych.
15. Podjąć próbę bezpośredniej zmiany wielkości wejściowej „Temperatura F1" na mniejszą i większą za pomocą klawiszy skrótu Ctrl + F1 oraz Shift + F1.
16. Podjąć próbę zmian wielkości wejściowych „Ciśnienie F2" oraz „Wilgotność F3" na mniejszą i większą za pomocą klawiszy.
17. Jeśli próby zmian wartości wejściowych zakończyły się sukcesem, zakończyć działanie programu głównego klawiszem Esc.
18. Zamknąć plik programu głównego (Alt + F4).

7.3.4. Kontrolka zakładkowa

Cel zadania:	zapoznanie się z obiektem panelu czołowego, który oszczędza miejsce i przyśpiesza pracę, czyli kontrolką zakładkową (tab control).
Zakres zadania:	kontrolka zakładkowa jest obiektem, który w systemach operacyjnych stosowany jest często jako element arkuszy właściwości. W trakcie ćwiczenia zostanie opracowany wirtualny generator, który będzie prezentował przebiegi na oddzielnych kartach kontrolki zakładkowej. Jednocześnie wybór karty kontrolki będzie wpływał na kod wykonywanego programu.

1. **Budowa panelu czołowego**. Uruchomić środowisko LabVIEW z pustym plikiem programu. Docelowo wymagana będzie budowa panelu czołowego zaprezentowanego na rysunku.



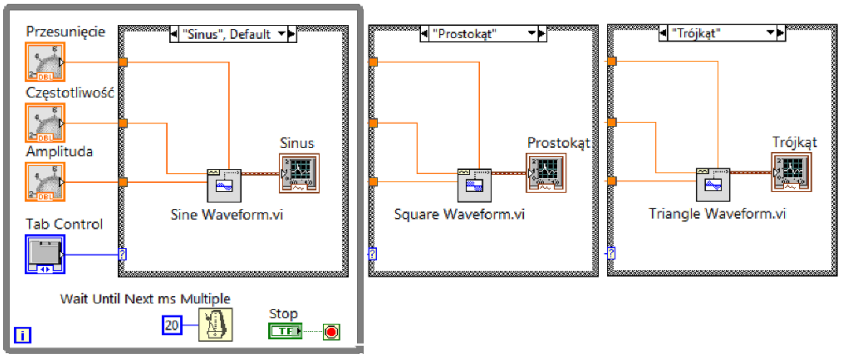
2. Panel zawiera:
- przycisk Stop (Controls /Modern /Boolean – Stop Button). Komunikat przycisku powinien brzmieć „STOP”. Należy usunąć etykietę przycisku,
- trzy zmienne wejściowe w postaci pokręteł (Controls /Modern /Numeric – Knob). Należy zmodyfikować terminale wejściowe w poniższy sposób:
  1 – etykieta: „Częstotliwość”, dane wejściowe: zakres = 0–6,
  2 – etykieta: „Amplituda”, dane wejściowe: zakres = 0–10,
  3 – etykieta: „Przesunięcie”, dane wejściowe: zakres = 0–2, – obiekt wykresu Waveform Graph. Trzeba usunąć legendę przebiegu oraz etykietę wykresu. Po sformatowaniu wykresu należy go skopiować jeszcze dwukrotnie.

3. Pod zamieszczonymi (w dolnej warstwie) pokrętłami i przyciskiem Stop wprowadzić element graficzny Raised Beveled Box (Controls /Modern /Decorations).
4. Umieścić obiekt kontrolki zakładkowej Tab Control (Controls /Modern /Containers).
- Usunąć etykietę kontrolki (menu kontekstowe /Visible Items /Tabs).
- Dodać trzecią kartę kontrolki (menu kontekstowe /Add Page After (Before)).
- Zmienić nazwy kart na „Sinus”, „Prostokąt”, „Trójkąt”.
- Na każdej z kart zamieścić jeden wykres Waveform Graph.

5. Zapisać plik programu pod nazwą (074_Karty.vi) w lokalizacji wskazanej przez prowadzącego.

6. **Schemat blokowy programu**. Przejść do okna schematu blokowego. Zawiera on jedynie ikony elementów zamieszczonych na panelu czołowym. Należy zwrócić uwagę, że ikona Tab control ma postać zmiennej wejściowej, co należy intepretować w ten sposób, że wybierając karty kontrolki zakładkowej, zmieniamy wartość zmiennej Tab Control. Wartości tej zmiennej mogą być wykorzystane w kodzie programu.

7. Zamieścić pętlę While (Functions /Programming /Structures /While Loop).
8. W pętli umieścić węzeł opóźnienia Wait Until Next ms Multiple. Deklarację stałej liczbowej (20 ms) należy przeprowadzić za pomocą menu kontekstowego węzła (menu kontekstowe /Create /Constant /wprowadzenie oczekiwanej wartości).

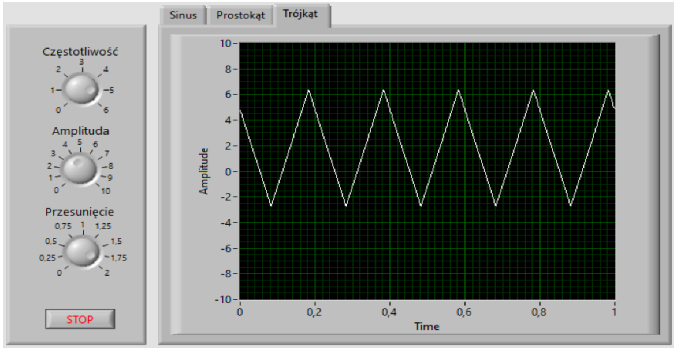


9. Wewnątrz pętli zamieścić strukturę wyboru (Case structure) (palety Functions / Programming /Structures).
- Dodać trzeci przypadek za pomocą menu kontekstowego (Add Case After (Before)).
- Wprowadzić nazwy przypadków „Sinus", „Prostokąt", „Trójkąt". Ważne jest, aby nazwy przypadków były w pełni zgodne z nazwami kart kontrolki zakładkowej.

10. Zamieścić generatory przebiegów czasowych zgodnych w odpowiednich przypadkach struktury wyboru, tj. przypadek „Sinus" – podprogram Sine Waveform.vi, przypadek „Prostokąt" – podprogram Square Waveform.vi, przypadek „Trójkąt" – podprogram Triangle Waveform.vi (Functions /Waveform /Analog Waveform / Waveform Generation lub Functions /Signal Processing /Waveform Generation).

11. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
12. Zapisać plik programu pod aktualną nazwą (074_Karty.vi).

13. **Kontrola sposobu funkcjonowania programu**. Uruchomić program w trybie jednokrotnym. Program powinien pierwotnie generować przebieg z domyślnej struktury wyboru. Jeśli schemat blokowy jest identyczny z zamieszczonym na poprzednim rysunku, automatycznie generowany będzie przebieg sinusoidalny.
14. Skontrolować reakcję na modyfikację wartości zmiennych wejściowych „Częstotliwość", „Amplituda", „Przesunięcie".
15. Przejść na kolejną kartę kontrolki zakładkowej (np. na przebieg prostokątny). Zaobserwować, że zmiana karty (wykresu) spowodowała również modyfikację wykonywanego kodu. Dokonać modyfikacji przebiegu za pomocą pokręteł panelu.



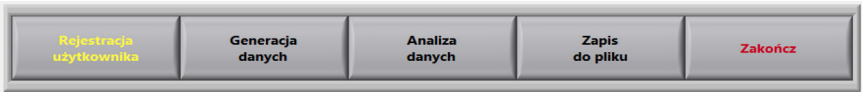
16. Zmienić kartę kontrolki zakładkowej (np. na przebieg trójkątny). Dokonać modyfikacji przebiegu i zamknąć okno programu.

7.3.5. Klaster menu

Cel zadania: wprowadzenie do tworzenia i korzystania z menu klastrowego. Klaster menu stanowi wygodną konstrukcję pozwalającą na elastyczne sterowanie kodem programu.

Zakres zadania: w trakcie ćwiczenia zostanie opracowane menu oparte na klastrze. Wybór poszczególnych przycisków będzie skutkował wykonaniem kodu zamkniętego w strukturach wyboru. Na potrzeby ćwiczenia (zamiast pełnych programów) zostaną wykorzystane komunikaty przypisane przyciskom.

1. **Budowa panelu czołowego**. Uruchomić środowisko LabVIEW z pustym plikiem programu. Docelowo wymagana będzie budowa panelu czołowego zaprezentowanego na rysunku.



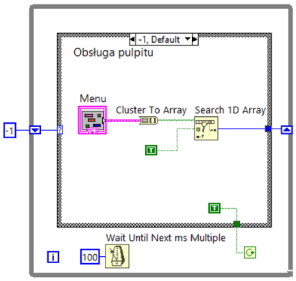
2. Panel zawiera:

- pięć przycisków Stop (Controls /Modern /Boolean – Stop Button). Należy usunąć etykietę przycisków, następnie komunikaty przycisków zmienić na „Rejestracja użytkownika", „Generacja danych", „Analiza danych", „Zapis do pliku", „Zakończ". Dodatkowo można wprowadzić formatowania czcionki przycisku, modyfikujące rozmiar, wytłuszczenie, barwę. Powiększyć rozmiar przycisków tak, by były zbliżone wyglądem do zamieszczonych na rysunku,
- klaster wejściowy Cluster (Controls /Modern /Array, Matrix & Cluster),

3. Wprowadzić przyciski do wnętrza klastra. Uporządkować położenie przycisków za pomocą poleceń przycisków paska narzędzi (Allign, Distribute, Resize Objects oraz Resize).
4. Uporządkować klaster (menu kontekstowe klastra i polecenie Reorder Controls in Cluster), aby kolejność była zgodna z zamieszczoną na rysunku.

5. Pod zamieszczonym (w dolnej warstwie) klastrem wprowadzić element graficzny Raised Beveled Box (Controls /Modern /Decorations).
6. Zapisać plik programu pod nazwą (075_System.vi) w lokalizacji wskazanej przez prowadzącego.

7. **Schemat blokowy programu**. Przejść do okna schematu blokowego. Zawiera ono jedynie ikonę zamieszczoną na panelu klastra.
8. Umieścić pętlę While (Functions /Programming /Structures /While Loop). Za pomocą menu kontekstowego terminala warunkowego zmienić jego stan na „Continue if True".
9. W pętli zamieścić węzeł opóźnienia Wait Until Next ms Multiple. Deklarację stałej liczbowej (100 ms) należy przeprowadzić za pomocą menu kontekstowego węzła (menu kontekstowe /Create /Constant /wprowadzenie oczekiwanej wartości).



10. Zamieścić strukturę wyboru Case (Functions /Programming /Structures /Case Structure).
11. Na krawędzi pętli, za pomocą menu kontekstowego, zadeklarować występowanie rejestru przesuwnego (Add Shift Register).
12. Wprowdzić stałą liczbową (Functions /Programming /Numeric – Numeric Constant), zadeklarować jej wartość na „-1" i przyłączyć do komórki wejściowej rejestru.
13. Wprowadzić dwie stałe logiczne True (Functions /Programming /Boolean – True Constant).
14. W domyślnym przypadku struktury Case zamieścić:

- ikonę klastra utworzonego na panelu czołowym,
- funkcję Cluster to Array (Functions /Programming /Array),
- węzeł Search 1D Array (Functions /Programming /Array).

15. Do przypadku domyślnego przypisać etykietę (edycja w miejscu) „Obsługa pulpitu".
16. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
17. Zapisać plik programu pod aktualną nazwą (075_System.vi).

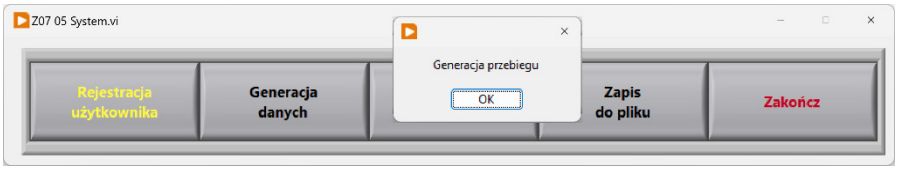
18. Uzupełnić strukturę wyboru o pozostałych pięć przypadków (menu kontekstowe).
19. Zamieścić wewnątrz nowych ramek struktury:
- etykieta z komentarzem (lewy góry róg każdej ramki),
- węzeł okna dialogowego One Button Dialog (Functions /Programming /Dialog & User Interface – One Button Dialog),
- stałą łańcuchową z tekstem zgodnym z etykietą ramki (Functions /Programming / String – String Constant),
- stałą liczbową (Functions /Programming /Numeric – Numeric Constant), o wartości na „-1”,
- stałe logiczne True (Functions /Programming /Boolean – True Constant).

20. Ostatnia z ramek kończąca program różni się od pozostałych:
- brakiem węzła okna dialogowego One Button Dialog,
- stałą logiczną False wprowadzającą zakończenie wykonywania pętli i programu.

21. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
22. Opracować oryginalną ikonę programu.

23. Zapisać plik programu pod aktualną nazwą (075_System.vi).

24. **Kontrola sposobu funkcjonowania programu**. Uruchomić program w trybie jednokrotnym. Program powinien reagować komunikatami zgodnymi z komunikatami zamieszczonymi na przyciskach menu.
25. Wybierać kolejno przyciski począwszy od „Rejestracji użytkownika” oraz zatwierdzać wyświetlane okna dialogowe.
26. W przypadku niezgodności komunikatu z oczekiwanym, sprawdzić kolejność przycisków w klastrze lub kolejność ramek struktury Case.



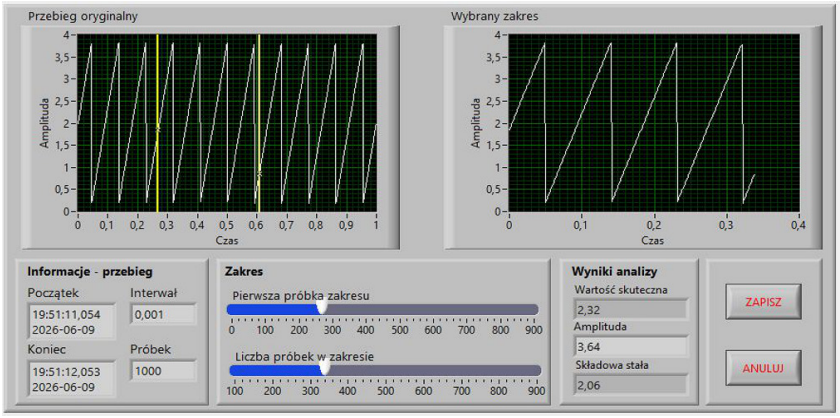
27. Zakończyć działanie programu przyciskiem Zakończ i zamknąć okno programu.

7.3.6. Analizator – węzły właściwości

Cel zadania: rozwijanie umiejętności stosowania węzłów właściwości w przypadku węzłów właściwości wykresu czasowego.

Zakres zadania: opracowanie programu do wybierania części przebiegu czasowego poddawanego analizie. Węzły właściwości zostaną wykorzystane do sterowania położeniem kursorów wykresu czasowego.

1. **Budowa panelu czołowego**. Celem programu jest opracowanie wirtualnego analizatora sygnału. Analizator pozwala na wczytanie przebiegu czasowego z pliku, wyświetlenie przebiegu i jego parametrów, wybranie części przebiegu, wyświetlenie wybranej części przebiegu, określenie wartości maksymalnej, skutecznej i składowej stałej wybranej części przebiegu. Program można zakończyć bez zapisu lub z zapisem wybranej części przebiegu.
2. Uruchomić środowisko LabVIEW z pustym plikiem programu. Zbudować panel czołowy zaprezentowany na rysunku.



3. Panel zawiera:

- dwa obiekty wykresu Waveform Graph (Control /Modern /Graph). Usunąć legendy przebiegów, zmienić tytuły na „Przebieg oryginalny” oraz „Wybrany zakres”, zmienić etykiety osi na „Czas” oraz „Amplituda”.
- dwa przyciski Stop (Controls /Modern /Boolean – Stop Button). Komunikaty przycisków zmodyfikować na „Zapisz: i „Anuluj”,
- dwie zmienne wejściowe w postaci suwaków Horizontal Pointer Slide (Controls / Modern /Numeric). Zmodyfikować terminale wejściowe w poniższy sposób:
  1 – etykieta: „Pierwsza próbka zakresu”, dane wejściowe: zakres = 0–900,
  2 – etykieta: „Liczba próbek w zakresie”, dane wejściowe: zakres = 100–900,

- pięć zmiennych wyjściowych typu liczbowego (Controls /Modern /Numeric – Numeric Indicator). Wprowadzić etykiety obiektów: Interwał, Próbek, Wartość skuteczna, Amplituda, Składowa stała,
- w dolnej warstwie pięć elementów graficznych Raised Beveled Box (Controls / Modern /Decorations) optycznie grupujących elementy panelu,
- trzy etykiety tekstowe (edycja w miejscu) o zawartości: „Informacje – przebieg", „Zakres", „Wyniki analizy",
- umieszczone poza krawędzią okna obiekty klastrów błędów Error In 3D.ctl Error Out 3D.ctl oraz (Controls /Modern /Array, Matrix & Cluster),
- wskaźnik LED (Controls /Modern /Boolean /Round LED) w obszarze okna niewidocznym dla użytkownika.

4. Obiekty zamieszczone w lewej części okna, oznaczone etykietami „Początek" i „Koniec", to zmienne wyjściowe znaczników czasowych, które zostaną wprowadzone za pomocą schematu blokowego.
5. Zapisać plik programu pod nazwą (076_Analizator.vi) w lokalizacji wskazanej przez prowadzącego.

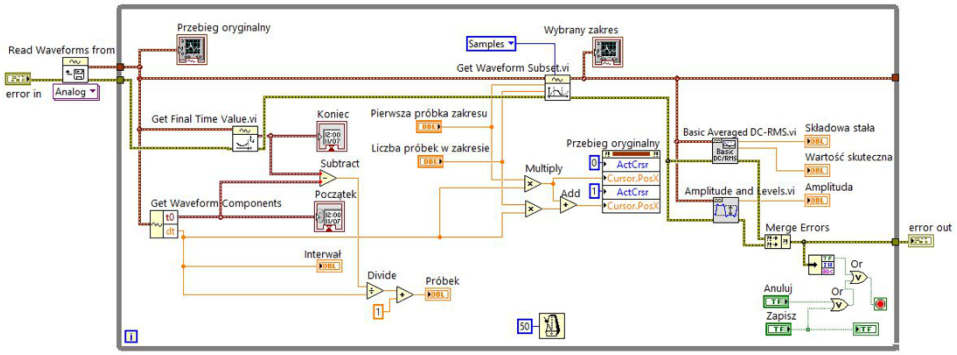
6. **Schemat blokowy programu**. Przejść do okna schematu blokowego. Schemat blokowy zawiera jedynie ikony elementów zamieszczonych na panelu czołowym. Widok obiektów panelu czołowego można różnicować za pomocą polecenia kontekstowego View As Icon.

7. Schemat blokowy uzupełnić o pętlę While (Functions /Programming /Structures / While Loop).
8. W pętli zamieścić węzeł opóźnienia Wait Until Next ms Multiple. Deklarację stałej liczbowej (50 ms) należy przeprowadzić za pomocą menu kontekstowego terminala wejściowego węzła.
9. Do okna schematu blokowego wprowadzić główne węzły funkcji:

- Read Waveforms from File.vi (Functions /Programming /Waveform /Waveform File I/O),

- Get Final Time Value.vi (Functions /Programming /Waveform),
- Get Waveform Subset.vi (Functions /Programming /Waveform). Za pomocą menu kontekstowego skonfigurować zmienną wejściową terminala „start/duration format" jako „Samples",
- Basic Averaged DC-RMS.vi (Functions /Programming /Waveform),
- Amplitude and Levels.vi (Functions /Programming /Waveform Analog Waveform / Waveform Measurements),
- Get Waveform Components (Functions /Programming /Waveform Analog Waveform),
- Merge Errors (Functions /Programming /Dialog & User Interface),
- Unbundle (Functions /Programming /Cluster, Class & Variant).

10. Zamieścić wskaźniki wyjściowe znaczników czasu funkcji:
- Get Final Time Value.vi – za pomocą menu kontekstowego wyjścia „tf" (Create Indictator). Znacznik czasu zostanie równocześnie zamieszczony na panelu czołowym. Zmienić nazwę etykiety na „Koniec",
- Get Waveform Subset.vi – za pomocą menu kontekstowego wyjścia „t0" (Create Indictator). Znacznik czasu zostanie równocześnie zamieszczony na panelu czołowym. Zmienić nazwę etykiety na „Początek".



11. Dodać funkcje matematyczne i logiczne. Z głównej palety Programming wybierać odpowiednio subpalety Numeric lub Boolean, a z nich:
- funkcję mnożenia Multiply (2 obiekty),
- funkcję dzielenia Divide (1 obiekt),
- funkcję dodawania Add (2 obiekty),
- funkcję odejmowania Subtract (1 obiekt),
- funkcję sumy logicznej Or (2 obiekty).

Wymagane wartości stałych deklarować za pomocą menu kontekstowych wejść węzłów.

12. Wprowadzanie węzła właściwości dla wykresu z oryginalnym przebiegiem czasowym. Celem będzie zadeklarowanie węzła, który:
- wybiera pierwszy z kursorów wykresu (oznaczony jako 0),
- ustawia jego pozycję zgodnie z pozycją suwaka „Pierwsza próbka zakresu”,
- wybiera drugi z kursorów wykresu (oznaczony jako 1),
- ustawia jego pozycję zgodnie z pozycją suwaków „Pierwsza próbka zakresu” oraz „Liczba próbek w zakresie”.

W celu aktywowania właściwości „Aktywny kursor” – ActCrsr z menu kontekstowego wykresu „Przebieg oryginalny” należy wybrać polecenie Create /Property Node /Active Cursor.

W kolejnych krokach:
- za pomocą menu kontekstowego utworzonego węzła zmienić jego charakter na wejściowy (Change All To Write),
- wybrać lewym przyciskiem myszy drugą z pozycji węzła właściwości i zadeklarować oczekiwaną właściwość jako Cursor Position X (Cursor /Cursor Position/ Cursor X),
- powtórzyć operację ustawiania właściwości Active Cursor dla trzeciego elementu węzła oraz Cursor Position X dla czwartego elementu węzła,
- wybór kursorów zadeklarować za pomocą menu kontekstowego pierwszego i trzeciego wejścia węzła właściwości. Wyboru kursorów dokonuje się przez przypisanie stałych liczbowych (0 i 1),

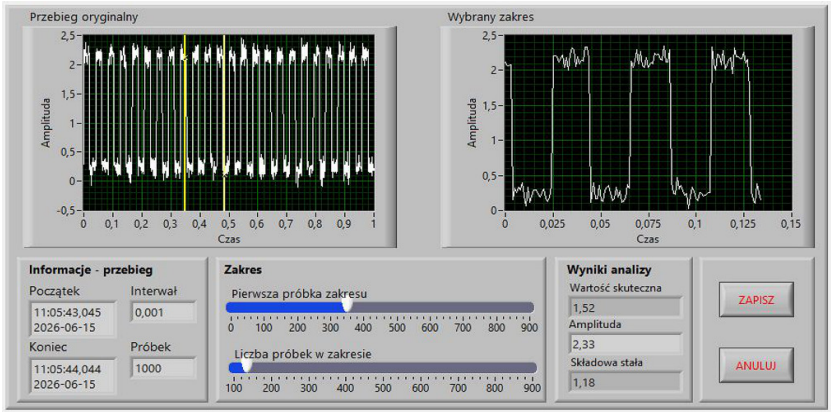
13. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu.
14. Zapisać plik programu pod aktualną nazwą (076_Analizator.vi).

15. **Czynności końcowe**.
16. Skonfigurować dwa terminale wyjściowe programu. Dwie zmienne wyjściowe to:
- terminal w postaci obiektu wykresu „Wybrany zakres”,
- terminal logiczny, który stanowi wskaźnik logiczny umieszczony poza widocznym obszarem okna (Boolean).

17. Opracować oryginalną ikonę programu.

18. Zapisać plik programu pod aktualną nazwą (076_Analizator.vi).

19. **Uruchamianie programu**. Do testowania programu wymagane są pliki z przebiegami czasowymi, które powinien udostępnić prowadzący. Uruchomić program w trybie jednokrotnym.
20. Bezpośrednio po uruchomieniu wyświetlane jest okno dialogowe pozwalające na wskazanie pliku z przebiegiem. Wskazać plik i zatwierdzić wybór przyciskiem [OK].
21. Sprawdzić dostępne informacje o przebiegu (lewy dolny róg okna).



22. Zmieniać analizowany zakres za pomocą suwaków „Pierwsza próbka zakresu” oraz „Liczba próbek w zakresie”.
23. W zależności od wybranego zakresu prezentowanego na wykresie „Wybrany zakres”, zmianom powinny ulegać wartości takich parametrów jak: Wartość skuteczna, Amplituda, Składowa stała.
24. Zakończyć pracę programu przyciskiem Zapisz lub Anuluj. Program powinien zakończyć działanie bez dokonywania zapisu, ponieważ taka funkcja nie została jeszcze wprowadzona.
25. Zamknąć plik programu.

7.4. Pytania kontrolne

1. Co może przyczynić się do nieefektywnego opracowania interfejsu graficznego przez użytkownika aplikacji?
2. W jaki sposób można uzyskać szybki efekt usunięcia rozpraszających bądź zbędnych elementów okna aplikacji?
3. Jakie problemy mogą pojawić się po zastosowaniu automatycznego skalowania wszystkich obiektów, razem z rozmiarem okna aplikacji?
4. W jaki sposób można uzyskać dynamiczne sterowanie obiektów w trakcie funkcjonowania programu?

8. Wymiana informacji między równocześnie działającymi procedurami systemu diagnostycznego

8.1. Cel zajęć

W środowiskach opartych na graficznym interfejsie mogą występować problemy z kolejnością wykonywania działań, które nie są zdeterminowane przez konkretne struktury mające swoją reprezentację graficzną. Takie nieuporządkowane działania przysparzają dodatkowo problemów z przekazywaniem wartości zmiennych w zdeterminowanych punktach czasu. Rozdział ten ma za zadanie przedstawić „naturalną" i wymuszoną kolejność wykonywania kodu oraz funkcje pozwalające na wymianę danych zarówno pomiędzy procedurami wewnątrz programu, jak i pomiędzy programami.

Celem zajęć jest praktyczne zapoznanie z:

- „naturalnym" porządkiem wykonywania kodu węzłów środowiska programowania,
- wymuszaniem kolejności realizacji procedur,
- wymianą danych w zakresie aplikacji za pomocą obiektu zmiennej lokalnej i globalnej,
- wymianą danych pomiędzy aplikacjami za pomocą zmiennej globalnej,
- wymianą danych w obszarze pojedynczej aplikacji, pomiędzy aplikacjami oraz w środowisku sieciowym za pomocą protokołu DataSocket.

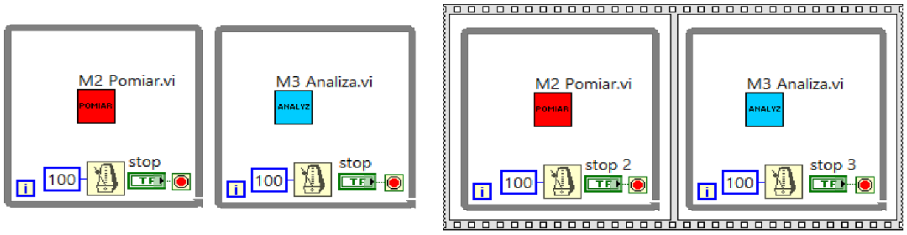
8.2. Wstęp

8.2.1. Zasada wykonywania kodu programu

W tekstowych środowiskach programistycznych (np. C#) kolejność wykonywania operacji definiowana jest za pomocą kolejnych komend i procedur. Programista determinuje kolejność wykonywania kodu. Środowisko LabVIEW wykorzystuje tzw. model strumienia danych. W przypadku skomplikowanego kodu trudno bez uruchomienia programu określić, jaka jest kolejność uruchamiania kolejnych węzłów (funkcji, podprogramów). Obowiązująca zasada działania programu kolejnych węzłów kodu polega na tym, że:

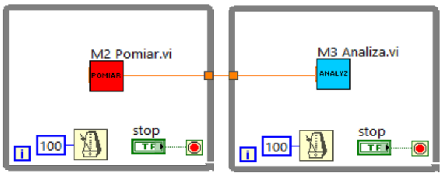
- węzeł jest wykonywany w przypadku dostarczenia wszystkich wymaganych danych wejściowych,
- po wykonaniu węzła dane udostępniane są we wszystkich terminalach wyjściowych węzła.

Brak połączenia pomiędzy węzłami skutkuje ich równoległą pracą. Bez dodatkowych elementów porządkujących kod, praca takich równoległych węzłów pozostaje niezsynchronizowana. Przy braku połączeń można wymusić kolejność wykonywania węzłów, stosując strukturę sekwencyjną (rys. 8.1).



Rys. 8.1. Operacje równoległe bez determinowania czasu oraz operacje z kolejnością wykonywania wymuszoną strukturą sekwencyjną

Model działania programu oparty jest na rozpoczęciu wykonywania operacji węzła po dostarczeniu do niego wszystkich zmiennych wejściowych. Po przeprowadzeniu wszystkich operacji zadeklarowanych w węźle wartości zmiennych dostępne są jednocześnie na wszystkich wyjściach. Taki mechanizm wymiany danych pomiędzy węzłami eliminuje równoległość wykonywania operacji.



Rys. 8.2. Brak równoległości wykonywania kodu przy przekazywaniu danych za pomocą połączeń

Jak widać na rysunku 8.2, wprowadzenie połączeń jednocześnie, eliminuje równoległość wykonywanych operacji. Druga pętla może zostać uruchomiona jedynie w przypadku zakończenia pierwszej pętli. Powyższe czynniki powodują, że w środowisku LabVIEW zostały zaimplementowane mechanizmy przekazywania danych w inny sposób niż za pomocą połączeń („drutowania"). Obiektami, które pozwalają na przekazywanie danych bez stosowania połączeń, są między innymi obiekty zmiennej lokalnej i globalnej.

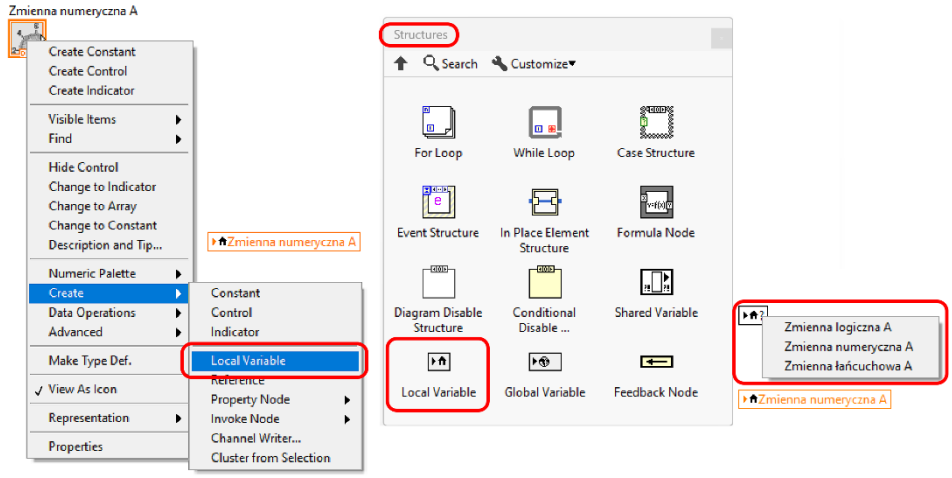
8.2.2. Zmienna lokalna

Obiekt zmiennej lokalnej (local variable) należy traktować jako kolejną kopię wartości wybranej zmiennej w pamięci. Zmienna lokalna może przechowywać dane zarówno zmiennej wejściowej, jak i wyjściowej. Lokalność zmiennej wynika z ograniczenia jej funkcjonowania do zakresu pojedynczego programu.

Głównym zastosowaniem zmiennej lokalnej jest udostępnianie wartości zmiennej, której jest przypisana. Można ją traktować jako obszar pamięci, do której można załadować wartość dowolnej zmiennej lub wartość z tego obszaru odczytać. Zmienna lokalna jest tylko obiektem schematu blokowego, tzn. jej ikona nie występuje na panelu czołowym aplikacji.

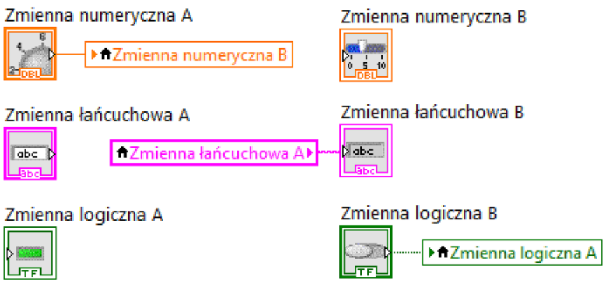
Wprowadzenie zmiennej lokalnej można zrealizować dwiema metodami (rys. 8.3):

- poprzez menu kontekstowe zmiennej, dla której chcemy zdefiniować zmienną lokalną i polecenie Create /Local Variable,
- poprzez utworzenie obiektu zmiennej lokalnej (obiekt Local Variable z subpalety Structures), a następnie wybranie z menu kontekstowego zmiennej lokalnej, fizycznej zmiennej, którą ma odwzorowywać.



Rys. 8.3. Dwie metody wprowadzania zmiennej lokalnej

Domyślnie zmienna lokalna ma charakter zmiennej wejściowej, tzn. można deklarować jej wartość. Zastąpienie charakteru zmiennej lokalnej z „do zapisu" (cieńsza ramka) na „do odczytu" (grubsza ramka) i odwrotnie wykonywane jest przez menu kontekstowe i polecenia Change to Read /Write (rys. 8.4).

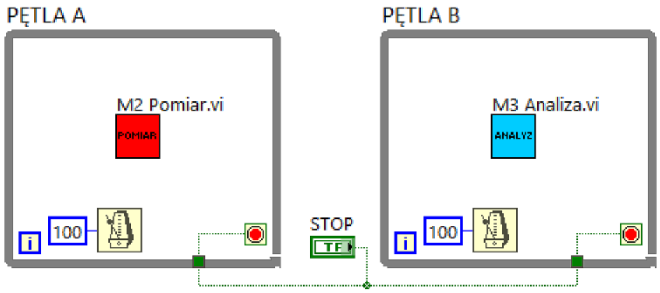


Rys. 8.4. Zmienne lokalne typu liczbowego, łańcuchowego oraz logicznego przeznaczone do zapisu i odczytu wartości zmiennych

Jak uprzednio wspomniano, zmienne lokalne pozwalają na przypisywanie (przekazywanie) wartości zmiennych bez konieczności wykonywania na schemacie blokowym połączeń (drutowania), które wprowadzają sekwencyjność wykonywania kodu.

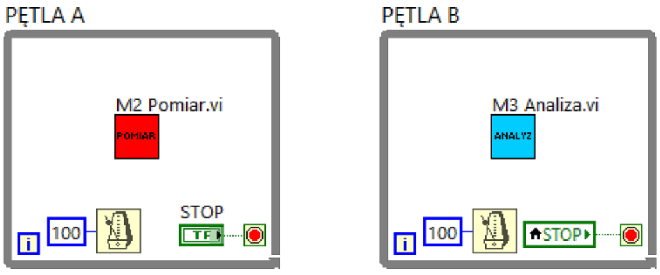
Dzięki temu można prowadzić równoległe operacje z wymianą danych. Klasycznym przypadkiem prostego wykorzystania zmiennych lokalnych jest użycie ich do równoległego kończenia działania pętli.

Na rysunku 8.5 zamieszczono schemat programu wykorzystującego pojedynczą zmienną logiczną do sterowania pętlami równoległymi. Analiza funkcjonowania kodu wskazuje, że nie ma możliwości efektywnego sterowania pętlami. Ponieważ terminale wejściowe pętli odczytywane są przed wykonaniem pętli, to możliwe są do uzyskania dwa sposoby działania programu. Jeśli przycisk [STOP] będzie miał domyślnie wartość „1", to pętle zostaną wykonane jednokrotnie i zostaną zakończone. Jeśli przycisk [STOP] będzie miał domyślnie wartość „0", to pętle zostaną uruchomione i nie będzie możliwości zatrzymania ich przyciskiem [STOP], ponieważ jego stan jest odczytywany przed inicjacją pętli.



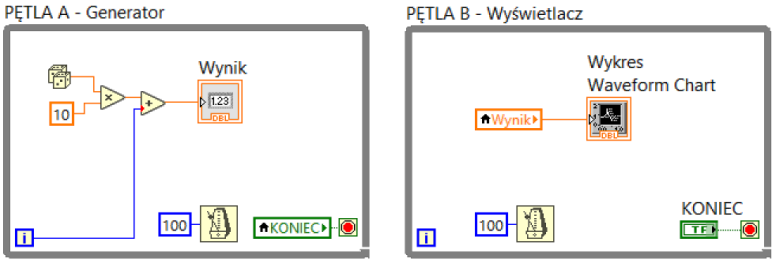
Rys. 8.5. Niepoprawne prowadzenie sterowania pętlami

Problem z kontrolą stanu przycisku [STOP] można usunąć przez umieszczenie tego przycisku w pętli, gdzie jego stan będzie kontrolowany na koniec wykonania kodu pętli. Druga z pętli może byś sterowna zmienną lokalną przycisku [STOP], która będzie miała taką samą wartość jak przycisk [STOP] z pierwszej pętli. Skutkiem będzie jednoczesne zatrzymanie pętli (rys. 8.6).



Rys. 8.6. Sterowanie pętlami za pomocą zmiennej lokalnej (STOP)

Inny przykładem „dwukierunkowego" zastosowania zmiennych lokalnych zamieszczono na rysunku 8.7. W tym przypadku zmienna lokalna Wynik w pętli B, przekazuje wartości rzeczywistej zmiennej Wynik z pętli A. W podobny sposób logiczna zmienna lokalna Wynik w pętli A, przekazuje wartości zmiennej logicznej [STOP] z pętli B.

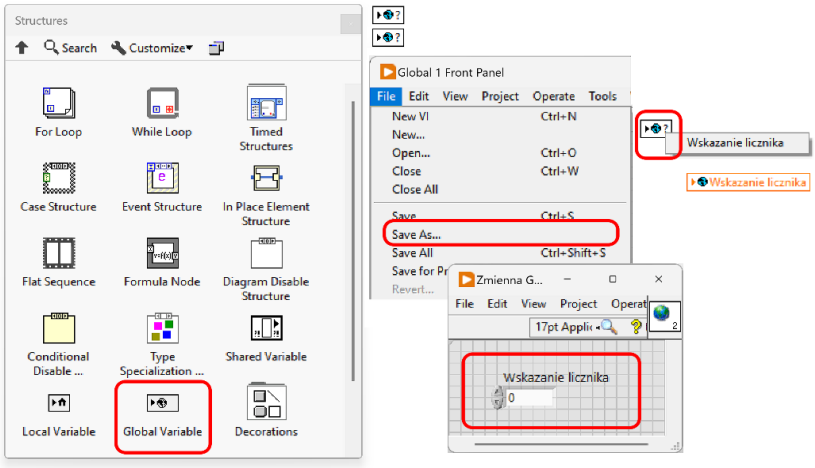


Rys. 8.7. Zastosowanie 2 zmiennych lokalnych w działaniach równoległych

8.2.3. Zmienna globalna

Zmienna lokalna (local variable) jest obiektem pozwalającym na wymianę danych w granicach jednej aplikacji. Zmienna globalna (global) przesuwa granicę wymiany danych tak, że za jej pomocą mogą transmitować dane różne równolegle działające aplikacje.

Praktycznie zmienna globalna jest typem specyficznego programu, który posiada sam panel czołowy. Zmienne zamieszczone na tym panelu przechowują wartości zmiennych dostępne dla różnych aplikacji. W odróżnieniu od zmiennej lokalnej, zmiennej globalnej nie da się utworzyć kontekstowo.



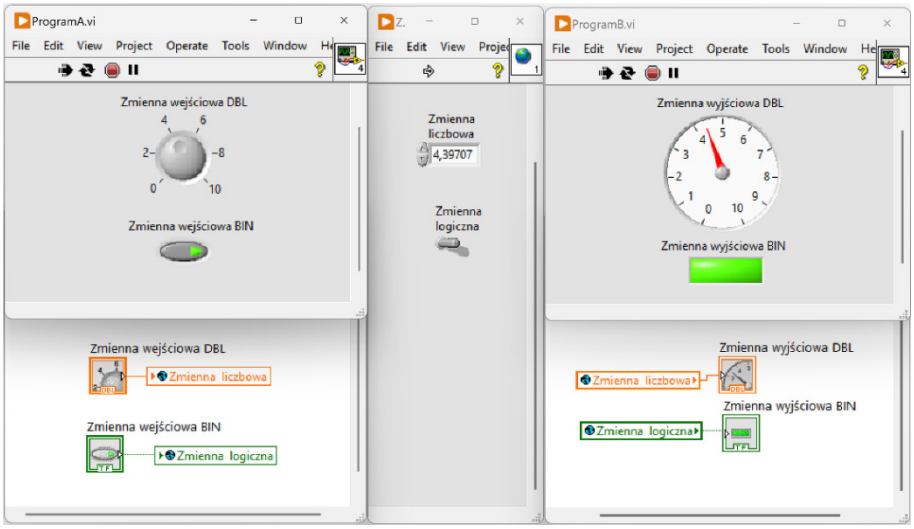
Rys. 8.8. Etapy zamieszczania zmiennej globalnej (global variable)

Proces zamieszczania zmiennej globalnej jest kilkuetapowy (rys. 8.8) i obejmuje:

- zamieszczenie niezdefiniowanej zmiennej globalnej – subpalety Structures /Global Variable,
- edycję panelu zmiennej globalnej,
- wprowadzenie zmiennych o typach danych zgodnych z przekazywanymi później typami danych,
- zapisanie pliku zmiennej globalnej (plik z rozszerzeniem .vi),
- wybór z menu kontekstowego zmiennej globalnej wartości nazwy wykorzystywanej zmiennej,
- w przypadku zamieszczania już istniejącej zmiennej globalnej (wcześniej zapisanego pliku) na schemacie blokowym programu, postępowanie jest takie, jak przy zamieszczaniu podprogramu.

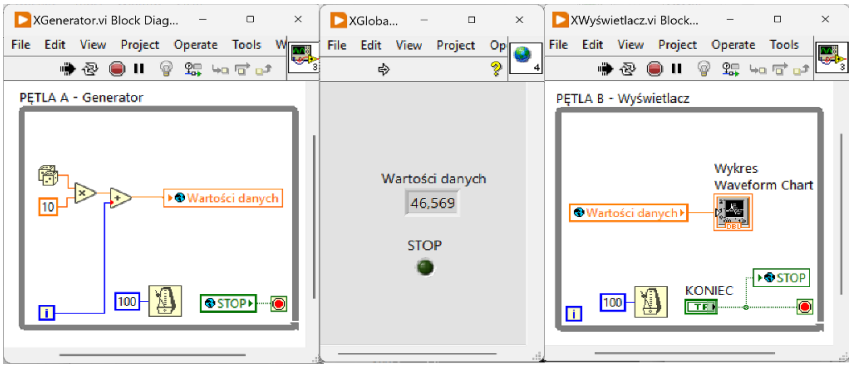
Domyślnie zmienna globalna (tak jak lokalna) ma charakter zmiennej wejściowej, tzn. można deklarować jej wartość. Zmiana charakteru zmiennej z typu „do zapisu" (cieńsza ramka) na „do odczytu" (grubsza ramka) i odwrotnie wykonywana jest przez menu kontekstowe i polecenia Change to Read /Write.

Zasada posługiwania się zmiennymi globalnymi została zaprezentowana na rysunku 8.9. W tym przypadku, w ramach aplikacji ProgramA.vi, jedna wartość zmiennoprzecinkowa i jedna wartość logiczna przypisywane są dwóm obiektom zmiennej globalnej. Wprowadzone wartości przechowywane są jako elementy pliku zmiennej globalnej. Aplikacja ProgramB.vi, za pomocą obiektów zmiennej globalnej, odczytuje wartości zapisane w pliku zmiennej globalnej. Wydaje się, że zmienną globalną najtrafniej jest rozpatrywać jako wartość wielkości zapisanej w pliku podprogramu.



Rys. 8.9. Idea udostępniania danych za pomocą zmiennych globalnych

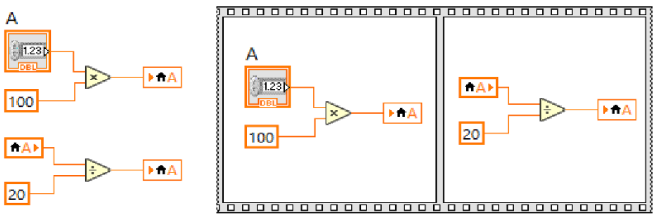
Na rysunku 8.10 zamieszczono przykład „dwukierunkowego" zastosowania zmiennych lokalnych. Przykład opisywał przypadek równoległego działania dwóch procedur (pętli), połączonego z wymianą danych pomiędzy pętlami za pomocą zmiennych lokalnych. Dzięki zmiennym globalnym możliwe jest opracowanie niezależnych programów, które również będą wymieniały dane. W przypadku programów z rysunku 8.10 program XGenerator jest źródłem danych, które zapisuje do zmiennej globalnej „Wartości danych". Wartości ze zmiennej globalnej prezentowane są w programie XWyświetlacz. Druga zmienna globalna [STOP] sterowana jest w XWyświetlacz, a jej wartość wykorzystywana do zakończenia pętli, a tym samym programu XGenerator.



Rys. 8.10. Zastosowanie zmiennych globalnych w działaniach równoległych dwóch aplikacji

8.2.4. Zmienna lokalna i globalna – uwagi i zalecenia

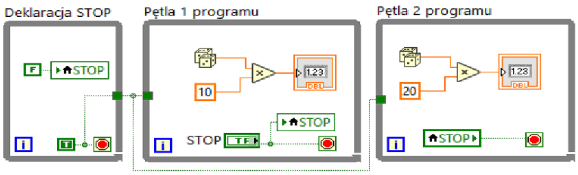
Jak uprzednio nadmieniono, w środowisku LabVIEW stosowany jest model strumienia danych. Jedyną obowiązującą regułą sterującą kolejnością obsługi węzłów jest uruchamianie kolejnych węzłów w ten sposób, że a) węzeł jest aktywowany w przypadku dostarczenia wszystkich wymaganych danych wejściowych, b) po wykonaniu węzła dane udostępniane są na wszystkich terminalach wyjściowych węzła [12]. Zmienne lokalne i globalne przekazują wartości zmiennych bez stosowania połączeń, więc oszacowanie na podstawie wyglądu kodu kolejności działania programu może być kłopotliwe (rys. 8.11).



Rys. 8.11. Metoda porządkowania kodu ze zmiennymi lokalnymi i globalnymi

Pierwszym aspektem, który należy uwzględnić, jest zapobieganie powstaniu warunków „współzawodnictwa" (wyścigu), czyli sytuacji, w jakiej brak jest wyraźnej zależności w kolejności wykonywania operacji, co może prowadzić do nieprecyzyjnego (nieoczekiwanego) wyniku. W kłopotliwych sytuacjach oczekiwane jest stosowanie struktur porządkujących, takich jak np. Sequence Structure.

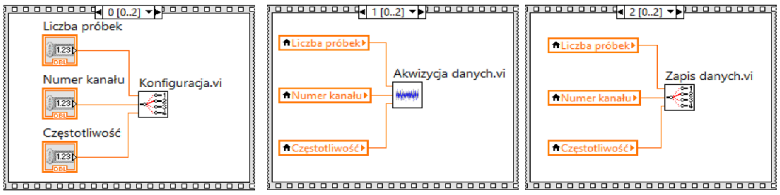
Drugim problemem jest ustalenie, jaką wartość początkową przypisano zmiennym lokalnym i globalnym. Zalecane jest świadome przypisywanie wartości zmiennym. W takim przypadku mogą być wykorzystane zarówno struktury sekwencyjne, jak i pętle, w których jedynym działaniem będzie przypisanie zmiennym lokalnym i globalnym wartości (rys. 8.12).



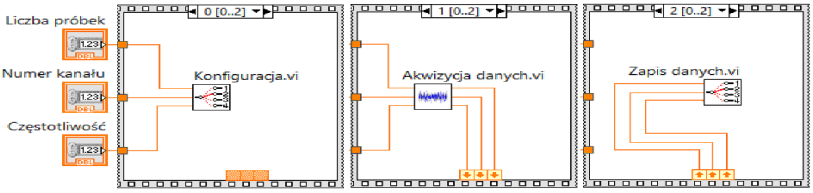
Rys. 8.12. „Inicjowanie" zmiennej lokalnej wartością wstępną

Kolejnym problemem jest nadużywanie zmiennych lokalnych i globalnych, do czego prowadzi pozorna łatwość dostępu do wartości zmiennej z każdego punktu schematu blokowego (rys. 8.13 i 8.14). Zalecane jest, aby zmienne lokalne i globalne wykorzystywać tylko w ostateczności, ponieważ [12, 13, 15]:

- każdy odczyt zmiennej prowadzi do kopiowania danych,
- dostęp do zmiennych podnosi zapotrzebowanie na pamięć,
- dostęp do zmiennych obniża prędkość działania aplikacji.



Rys. 8.13 Niepoprawne, nadmierne wykorzystanie zmiennych lokalnych



Rys. 8.14. Eliminacja zmiennych przez „drutowanie" i zmienną struktury sekwencyjnej

8.2.5. Technologia DataSocket

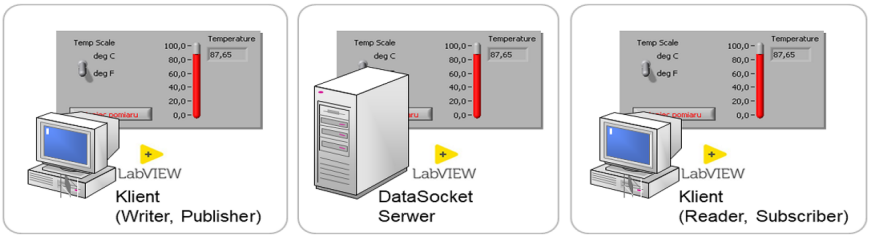
Konstrukcja i podstawy technologii DataSocket nie są szczegółowo opisane w materiałach twórców LabVIEW. W materiałach tych nacisk kładziony jest raczej na zastosowanie technologii w dystrybucji danych. Na podstawie sposobu obsługi, technologia DataSocket może zostać zakwalifikowana do warstwy aplikacji w modelu warstwowym protokołów TCP/IP. Za pomocą narzędzi związanych z DataSocket można uzyskać dostęp do danych dystrybuowanych z wykorzystaniem protokołów i źródeł:

- dstp: (DataSocket transfer protocol),
- http: (hypertext transfer protocol),
- ftp: (file transfer protocol),
- opc: (OLE for Process Control),
- fieldpoint:, logos:, lookout: (wewnętrzne protokoły komunikacyjne przeznaczone do oprogramowania lub grup urządzeń),
- file: (pliki lokalne komputera).

Spośród możliwych do wykorzystania w ramach całej technologii DataSocket protokołów, natywnym jest protokół DataSocket Transfer Protocol (dstp), charakteryzowany jako wygodny mechanizm monitorowania aktualnych wartości danych pomiarowych dystrybuowanych przez internet.

Tak jak cała technologia DataSocket, protokół dstp to protokół warstwy aplikacji. Protokół dstp jest zaimplementowany na szczycie TCP, dzięki czemu zapewnia komunikację zorientowaną na połączenie między serwerem a klientem [11]. Innymi słowy, klienci utrzymują sesję podczas komunikacji z serwerem, który śledzi informacje o połączeniu klienta.

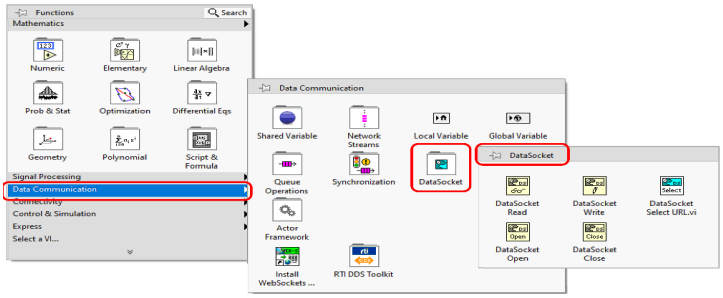
Konstrukcja systemu przekazywania danych jest nieco podobna do obsługi zmiennej globalnej (global variable) i wymaga trzech składników (rys. 8.15).



Rys. 8.15. Struktura komunikacyjna technologii DataSocket

Dostawcę danych określa się mianem wydawcy lub autora (writer, publisher). Centrum systemu stanowi pośredniczący w wymianie danych serwer (DataSocket server). Trzecim składnikiem jest jeden lub więcej odbiorców określanych jako subskrybenci lub czytelnicy (subscibers, readers). Trzy wymienione składniki mogą być rozlokowane na trzech fizycznych urządzeniach, dostępna jest również każda inna konfiguracja, czyli wszystkie trzy składniki mogą funkcjonować na jednym urządzeniu.

Narzędzia (węzły) do skonfigurowania komunikacji za pomocą DataSocket umieszczone są na subpalecie schematu blokowego DataSocket (paleta Programming / Data Communication /DataSocket) (rys. 8.16).

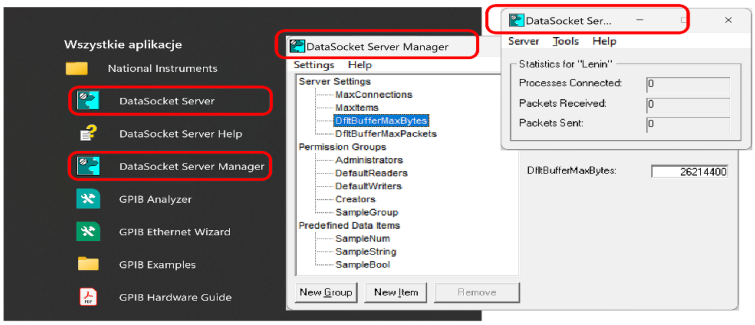


Rys. 8.16. Węzły obsługi komunikacji za pomocą DataSocket

Funkcjonalność pięciu węzłów z tej palety można podzielić na trzy grupy:

- DataSocket Write, DataSocket Read – węzły do bezpośredniego sterowania zapisem i odczytem danych z serwera DataSocket,
- DataSocket Open, DataSocket Close – węzły do sterowania połączeniem (wskazywaniem serwera, typu połączenia, czasu oczekiwania),
- DataSocket Select URL – podprogram ułatwiający wybór serwera (wskazanie nazwy) w przypadku pracy w środowisku z wieloma źródłami danych DataSocket.

Jednocześnie z instalacją środowiska LabVIEW instalowany jest serwer DataSocket. Dostęp do pliku uruchomieniowego serwera oraz menedżera konfiguracji serwera jest domyślnie dostępny w grupie programów National Instruments (rys. 8.17). Dopiero współpraca aplikacji udostępniającej dane, serwera i aplikacji pobierającej dane tworzy pełne środowisko, pozwalające na korzystanie z zalet protokołu dstp.

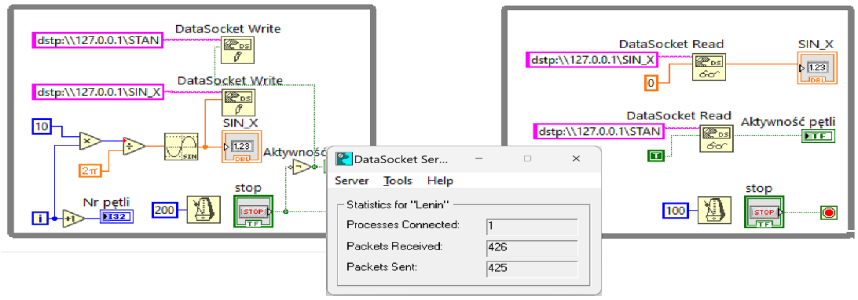


Rys. 8.17. Okno obsługi serwera i menedżera DataSocket

Na rysunku 8.18 zaprezentowano przykładową konfigurację trzech elementów wymieniających dane za pośrednictwem protokołu dstp. Kluczową operacją jest wysłanie

danych do serwera dstp. Obsługę zapewnia węzeł DataSocket Write. W celu zdefiniowania sposobu zapisu należy zadeklarować:

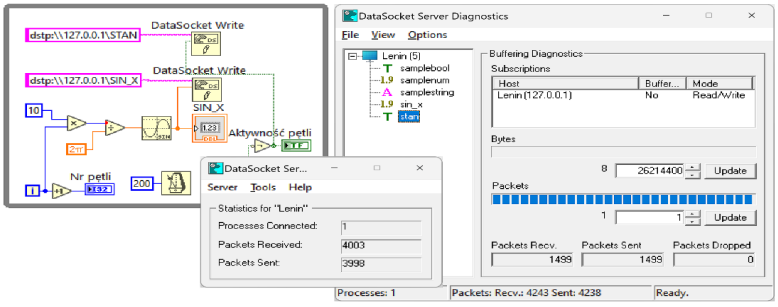
- rodzaj protokołu komunikacyjnego (tutaj dstp),
- nazwę komputera (urządzenia), na którym uruchomiony jest serwer (tutaj komputer lokalny wskazany za pomocą adresu IP 127.0.0.1),
- nazwę zmiennej, pod którą dostarczane są wartości (tutaj zmienna liczbowa z nazwą SIN_X oraz zmienna logiczna z nazwą STAN).



Rys. 8.18. Aplikacje wykorzystujące protokół dstp

Serwer wskazuje na podłączone jedno źródło danych wraz z liczbą otrzymanych i przesłanych pakietów.

Bardziej szczegółowych danych o połączeniach dostarcza polecenie Diagnostics z menu Tools serwera (rys. 8.19). Okno DataSocket Server Diagnostics prezentuje aktualny (bieżący stan serwera). Można zauważyć, że jeszcze przed zapisaniem danych do serwera transmituje on przykładowe dane logiczne (samplebool, wartość logiczna = True), liczbowe (samplenum, wartość liczbowa = 3,14) oraz łańcuchowe (samplestring, wartość = „abc"). Na liście obsługiwanych danych widać również samodzielnie deklarowane dane w postaci zmiennej liczbowej (nazwa SIN_X) oraz zmiennej logicznej (nazwa STAN). Prawa część okna prezentuje podsumowanie odebranych i przesłanych dalej danych mierzonych w pakietach danych oraz bajtach danych.



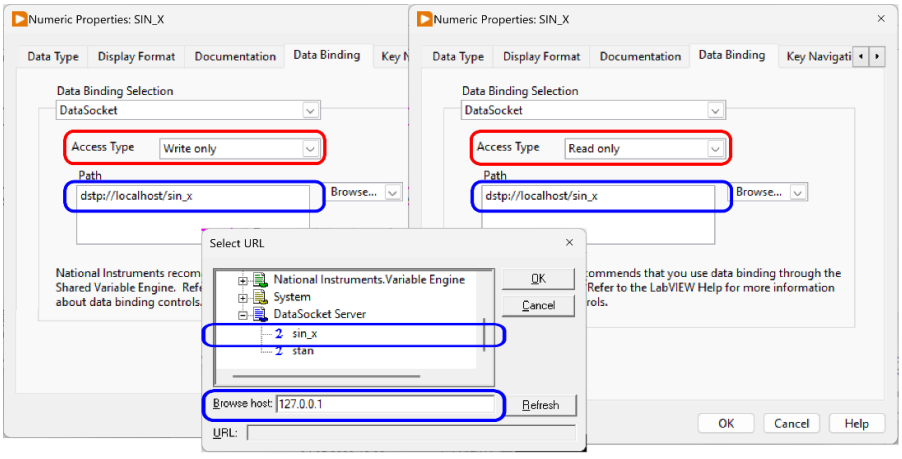
Rys. 8.19. Informacje szczegółowe o przesyłanych danych (menu Tools /Diagnostics)

Okna dialogowe DataSocket Server oraz DataSocket Server Diagnostics mają znaczenie informacyjne, ponieważ ukazują bieżący stan pracy serwera. Możliwość konfigurowania serwera DataSocket udostępnia DataSocket Server Manager. Za jego pomocą można zmieniać ustawienia w zakresie:

- dopasowania serwera do wydajności fizycznej urządzenia, na którym serwer jest uruchomiony oraz do przepustowości sieci (czyli maksymalnej liczby połączeń, publikowanych obiektów, przesyłanych pakietów i rozmiaru danych),
- kształtowania bezpieczeństwa przez definiowanie grup, które będą mogły zarządzać serwerem, deklarować obsługiwane typy obiektów, zapisywać i odczytywać dane,
- publikowania przykładowych danych – rozwijania, minimalizowania listy dostępnych domyślnie obiektów poza trzema predefiniowanymi.

W przypadku funkcjonującego serwera dstp istnieje możliwość publikowania i pozyskiwania z niego danych z pominięciem węzłów DataSocket Write i DataSocket Read. Konfigurację przeprowadza się na karcie Data Binding arkusza właściwości zmiennej. W kolejnych etapach konfiguracji należy (rys. 8.20):

- zadeklarować mechanizm połączenia (tutaj DataSocket (inne typy to np. Shared Variable)) i typ dostępu (Read, Write, Read/Write),
- wyszukać funkcję wspierającą wybrany mechanizm połączenia (tutaj DSTP Server),
- zlokalizować źródło danych na liście dostępnych obiektów (Browse…),
- wybrać nazwę danych publikowanych na serwerze.



Rys. 8.20. Przypisanie wartości z serwera dstp przez arkusz właściwości

Protokół dstp, podobnie jak http, może być wykorzystywany w sieciach rozległych. W związku z potencjalnym zagrożeniem atakami, które mogą nadejść z sieci zewnętrznej, systemy operacyjne i sieci chronione są za pomocą firewalli programowych

i sprzętowych. Funkcje ochronne mogą zablokować port, który wykorzystywany jest przez protokół dstp lub inne funkcje środowiska LabVIEW. Poniżej zamieszczono zestawienie portów dla wybranych usług i protokołów LabVIEW (tab. 8.1).

Tab. 8.1. Zestawienie portów dla wybranych usług i protokołów obsługiwanych w środowisku LabVIEW [9]

Funkcja lub moduł	Domyślny port serwera
MAX Hardware Identification	UDP 44515, UDP 44525, TCP 44516
LabVIEW Real-Time	TCP 3079
LabVIEW Remote Front Panels	TCP 8000 (bez SSL), TCP 433 (z SSL)
LabVIEW Web Services	TCP 8080
LabVIEW VI Server	TCP 3363
NI VISA Server	TCP 3537
FTP VIs (LabVIEW Internet Toolkit)	TCP 20 (active mode), 21 (passive mode)
Email VIs (SMTP)	TCP 25
HTTP Client VIs	TCP 80
Network Shared Variables	TCP 2343, UDP 6000-6010, TCP 59110 i wyższe
DataSocket (DSTP)	TCP 3015
LabVIEW TCP oraz UDP VIs	NA
Time Synchronization (NTP, SNTP)	TCP 123
NI Package Manager	HTTPS 80

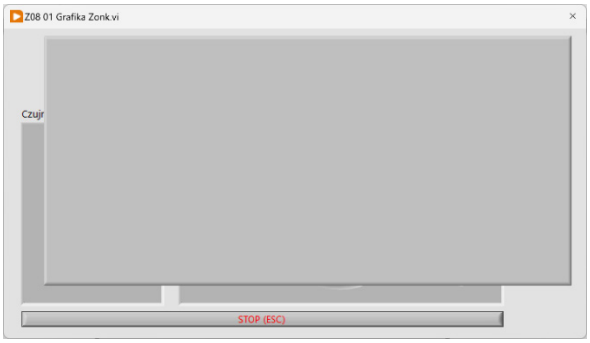
8.3. Zadania

8.3.1. Poprawki w „gotowych" aplikacjach

Cel zadania: zapoznanie z metodą edycji „gotowych" programów o konfiguracji utrudniającej modyfikację schematu blokowego.

Zakres zadania: przedstawienie metody dostępu do kodu źródłowego i panelu czołowego programu wstępnie opracowanego dla użytkownika końcowego. Metoda sprawdza się jedynie w sytuacji wstępnego skonfigurowania programu, które może polegać na zadeklarowaniu automatycznego uruchamiania kodu po wczytaniu aplikacji oraz samoczynnego zamykania panelu po zakończeniu wykonywania kodu. Przy takiej konfiguracji użytkownik nie ma możliwości wprowadzenia jakichkolwiek zmian w oknie panelu i schemacie programu.

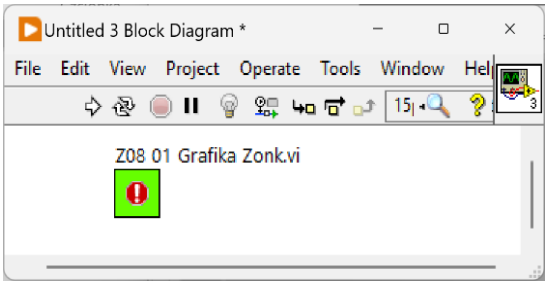
1. **Zapoznanie z aplikacją**. Uruchomić w środowisku LabVIEW uprzednio przygotowany przez prowadzącego program (081_Grafika_Zonk.vi).



2. Wyświetlony panel czołowy może wyglądać tak, jak zaprezentowany na rysunku.
3. Widoczny na rysunku błąd aplikacji polega na przypadkowym umieszczeniu obiektu graficznego na pierwszym planie panelu. Uniemożliwia to obsługę jakiejkolwiek funkcji poza przyciskiem Stop.
4. Wybrać przycisk Stop.
5. Aplikacja może zakończyć działanie wraz z zamknięciem całego środowiska LabVIEW. Analogicznie wykorzystanie ikony krzyżyka zamykającego okno aplikacji zamknie aplikację, pozostawiając aktywne środowisko LabVIEW, ale niekiedy bez możliwości edycji programu.

6. **Przejście do trybu edycji i usunięcie problemów**. Uruchomić środowisko LabVIEW z pustym plikiem programu (Blank.vi).
7. Zamieścić plik programu (081_Grafika_Zonk.vi) jako podprogram programu głównego Blank.vi. Poza:

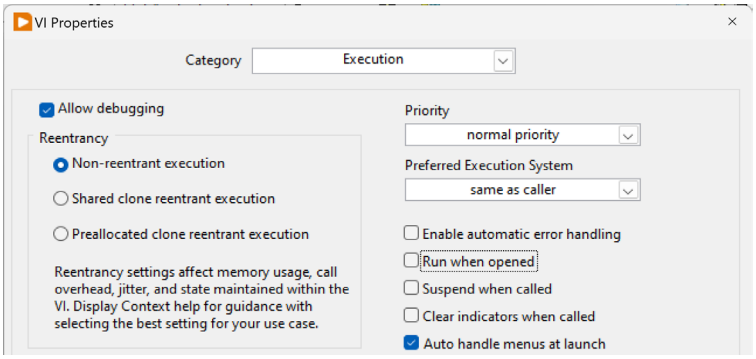
- przeciągnięciem za pomocą kursora myszy widocznej ikony podprogramu na obszar schematu programu głównego,
- przeciągnięciem za pomocą kursora myszy pliku programu widocznego w eksploratorze systemu operacyjnego możliwe jest
- wykorzystanie polecenia Select a VI (Programming /Select a VI) oraz wskazanie zamieszczanego pliku podprogramu.



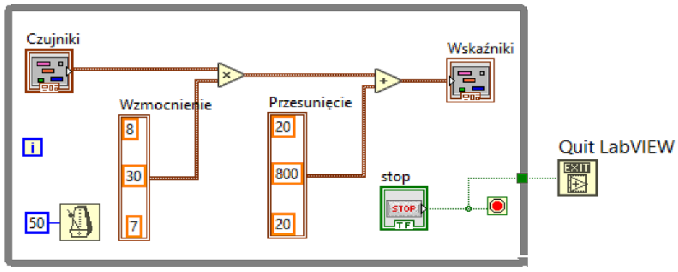
8. Przejść do edycji programu (Z08 01 Grafika Zonk.vi). Edycja podprogramu możliwa jest poprzez podwójne kliknięcie ikony podprogramu (opcjonalnie menu kontekstowe /Open Front Panel).
9. Powodem braku możliwości dostępu do (081_Grafika_Zonk.vi) były następujące czynniki:

- automatyczne uruchamianie,
- brak ikon pozwalających na przerwanie działania programu,
- automatyczne zamykanie środowiska wraz z zakończeniem pracy programu.

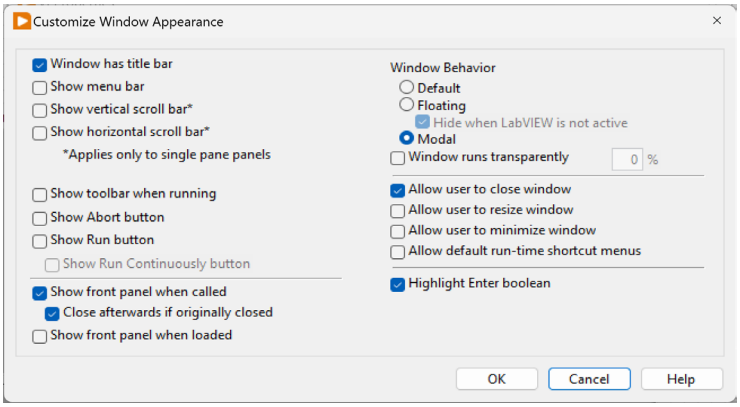
10. Anulowanie automatycznego uruchamiania aplikacji po wczytaniu programu. Przejść do grupy Execution arkusza właściwości programu (Z08 01 Grafika Zonk.vi) (menu File /VI Properties /Execution). Usunąć wybór opcji Run when opened. Zatwierdzić wybór przyciskiem [OK].



11. Anulowanie automatycznego zamykania środowiska LabVIEW wraz z zakończeniem pracy programu. Przejść do schematu blokowego programu. Usunąć węzeł Quit LabVIEW oraz osierocone połączenia kodu.



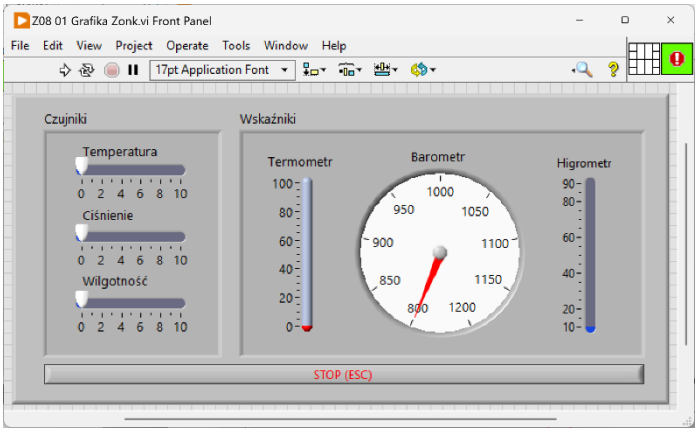
12. Przywracanie brakujących ikon pozwalających na przerwanie działania programu. Przejść do grupy Window Appearance arkusza właściwości programu (081_Grafika-Zonk.vi) (menu File /VI Properties /Window Appearance). Wybrać pracę programu w trybie domyślnym Default. Zatwierdzić wybór przyciskiem [OK].



13. Zapisać plik programu pod nazwą (081_Grafika_OK.vi).

14. **Usunięcie problemów z grafiką panelu czołowego**. Przejść do panelu czołowego programu (081_Grafika_OK.vi).

15. Widoczny na rysunku błąd aplikacji polega na przypadkowym umieszczeniu obiektu graficznego na pierwszym planie panelu. Uniemożliwia to obsługę jakiejkolwiek funkcji poza przyciskiem Stop.
16. Nie ma możliwości wybrania za pomocą kursora myszy pierwszoplanowego obiektu graficznego. W takich przypadkach istnieje prawdopodobieństwo wystąpienia blokowania (lock) obiektu. Taka konfiguracja ma tutaj miejsce.
17. Ponieważ nie ma możliwości wybrania pojedynczego obiektu, należy zaznaczyć (wybrać) wszystkie obiekty. Z grupy Reorder paska narzędzi wybrać polecenie Unlock.
18. Poprzez przesuwanie na płaszczyźnie i w warstwach (Reorder /Move to Back) spozycjonować pierwszoplanowy element graficzny tak, aby stanowił tło dla pozostałych elementów.



19. Zapisać plik programu pod istniejącą nazwą (081_Grafika_OK.vi).

20. **Kontrola sposobu funkcjonowania programu**. Uruchomić program w trybie jednokrotnym. Wynikiem działania programu powinna być reakcja zmiennych wyjściowych zamieszczonych w klastrze Wskaźniki na zmiany elementów wejściowych z klastra Czujniki.
21. Zamknąć plik programu po zakończeniu testowania.

8.3.2. Modyfikacja programu z wykorzystaniem zmiennych lokalnych

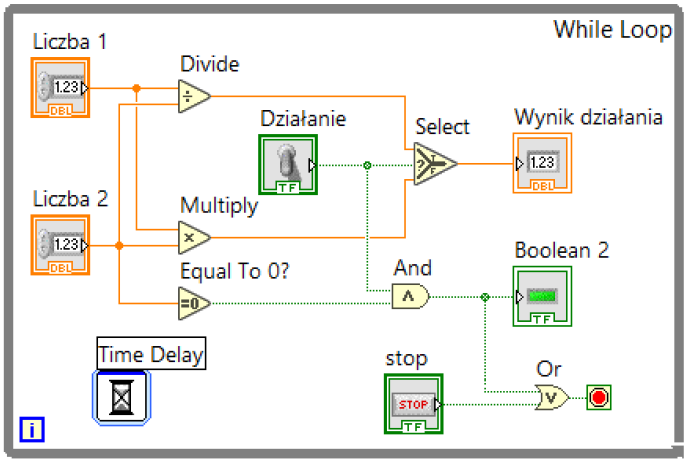
Cel zadania: zapoznanie się ze stosowaniem zmiennych lokalnych i węzłów właściwości w zakresie modyfikacji wartości zmiennych i zachowania obiektów panelu czołowego.

Zakres zadania: w trakcie ćwiczenia zmodyfikowany zostanie uprzednio wykonany program w taki sposób, aby unikać przypadkowych wartości zmiennych wejściowych. Deklaracja wartości zmiennych przeprowadzona będzie za pomocą zmiennych lokalnych. Jednocześnie zostaną zastosowane węzły właściwości do zmiany zachowania dwóch elementów panelu czołowego.

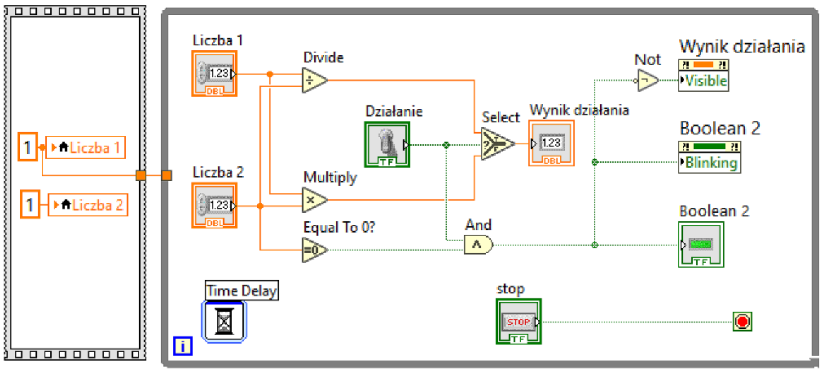
1. **Kontrola funkcjonowania programu przed modyfikacją**. Otworzyć uprzednio opracowany plik (031_Mnozenie.vi).
2. Zwrócić uwagę, że zmienne „Liczba 1” i „Liczba 2” posiadają zadeklarowane wartości 2. Deklaracja została wprowadzona przez menu kontekstowe i polecenia Data Operations /Make Current Value Default.
3. Uruchomić program w trybie jednokrotnym.

4. Zmieniać wartości zmiennych „Liczba 1” i „Liczba 2” oraz sposobu działania (Dzielenie, Mnożenie) w ten sposób, aby wartość „Liczba 2” nie osiągnęła wartości 0.
5. Zakończyć działanie przyciskiem [STOP].
6. Zwrócić uwagę na to, że wartości zmiennych „Liczba 1” i „Liczba 2” pozostały takie, jak przy ostatniej operacji.
7. Uruchomić program w trybie jednokrotnym. Zwrócić uwagę na to, że podobnie jak poprzednio wartości zmiennych „Liczba 1” i „Liczba 2” pozostały takie, jak przed uruchomieniem.
8. Zmienić sposób działania na Dzielenie i zadeklarować wartość „Liczba 2” równą 0.
9. Program powinien zakończyć działanie. W polu „Wynik działania” powinna się wyświetlić wartość nieskończoność (Inf).
10. Podjąć próbę ponownego uruchomienia programu. Program nie powinien się uruchomić do momentu zmiany wartości zmiennej „Liczba 2” na inną niż 0.

11. **Schemat blokowy programu**. Przejść do okna schematu blokowego.



12. Zapisać plik programu (031_Mnozenie.vi) pod nową nazwą (082_Mnozenie.vi).
13. Za pomocą menu kontekstowego zamieścić zmienne lokalne ikon terminali „Liczba 1” i „Liczba 2”.
14. Zamieścić jedną ramkę struktury sekwencyjnej (Functions /Programming /Structures – Flat Sequence Structure).
15. Wprowadzić zmienne lokalne „Liczba 1” i „Liczba 2” do wewnątrz struktury sekwencyjnej. Za pomocą menu kontekstowego zadeklarować wartości zmiennych lokalnych „Liczba 1” i „Liczba 2” (np. wartości 1) i wykonać połączenia zgodnie ze schematem blokowym.

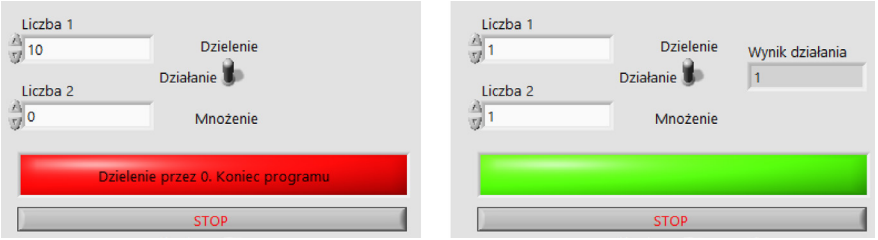


16. Za pomocą menu kontekstowego zamieścić węzły właściwości zmiennych (ikon):
- Wynik działania – węzeł wartość (Create /Property Node /Value),
- Boolean 2 – węzeł „migania” (Create /Property Node /Blinking).

17. Zmienić domyślny charakter wszystkich węzłów „do odczytu” na wejściowy „do zapisu”. Zmiany dokonać za pomocą menu kontekstowego i polecenia Change to Write (Change All to Write).
18. Zmodyfikować zestaw obiektów schematu przez:
- usunięcie funkcji logicznej Or,
- wprowadzenie funkcji logicznej Not (Functions /Programming /Boolean – Not).

19. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
20. Zapisać plik programu pod aktualną nazwą (082_Mnozenie.vi).

21. **Kontrola funkcjonowania programu**. Uruchomić program w trybie jednokrotnym.
22. Zwrócić uwagę, że zmienne „Liczba 1” i „Liczba 2” posiadają zadeklarowane wartości 1.
23. Zmieniać wartości zmiennych „Liczba 1” i „Liczba 2” oraz sposobu działania (Dzielenie, Mnożenie) w ten sposób, aby wartość „Liczba 2” nie osiągnęła wartości 0.
24. Zakończyć działanie przyciskiem [STOP].
25. Zwrócić uwagę na to, że wartości zmiennych „Liczba 1” i „Liczba 2” pozostały takie, jak przy ostatniej operacji.
26. Uruchomić program w trybie jednokrotnym. Zwrócić uwagę na to, że wartości zmiennych „Liczba 1” i „Liczba 2” posiadają jawnie zadeklarowane wartości (tutaj 1).
27. Zmienić sposób działania na „Dzielenie” i zadeklarować wartość „Liczba 2” równą 0.
28. Program powinien zasygnalizować „dzielenie przez 0” za pomocą migającego przycisku logicznego oraz ukrycia zmiennej wyjściowej „Wynik działania”.
29. Zakończyć działanie przyciskiem [STOP].



30. Uruchomić program w trybie jednokrotnym.
31. Zwrócić uwagę, że ponownie zmienne „Liczba 1" i „Liczba 2" posiadają zadeklarowane wartości 1, zaś „Wynik działania" jest wyświetlany.
32. Zakończyć działanie przyciskiem [STOP].
33. Zamknąć plik programu.

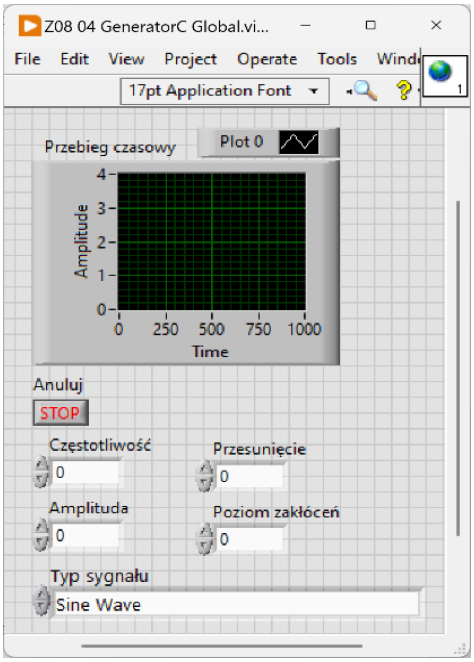
8.3.3. Wymiana danych za pomocą zmiennych globalnych

Cel zadania:	wprowadzenie do stosowania zmiennych globalnych do wymiany danych pomiędzy programami.
Zakres zadania:	na podstawie uprzednio opracowanego programu (Z08 03 Generator.vi) zostaną wykonane dwie aplikacje. Pierwsza z nich odpowiedzialna będzie za generację sygnału. Druga z nich będzie sterowała pierwszą aplikacją w zakresie parametrów generowanego przebiegu oraz będzie pozwalała na równoczesne zakończenie obu aplikacji. Wymiana danych realizowana będzie za pomocą zmiennej globalnej.

1. **Przygotowanie plików programów**. Pobrać plik aplikacji (083_Generator.vi). Wykonać dwie kopie pliku aplikacji i nadać im nazwy:

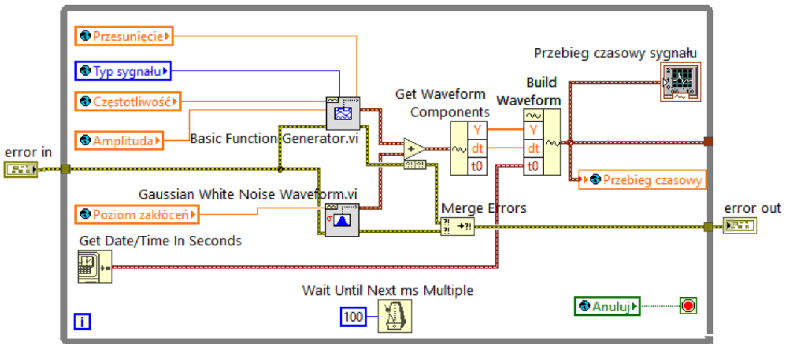
- (084_GeneratorA Engine.vi),
- (084_GeneratorB Control.vi).

2. **Opracowanie programu „silnika" i zmiennej globalnej**. Otworzyć plik programu 084_GeneratorA Engine.vi. Przejść do schematu blokowego.
3. Wprowadzić zmienną globalną do wnętrza pętli While (subpaleta Functions / Programming /Structures /Global Variable). Zamieszczona zmienna globalna nie zawiera jeszcze żadnych danych.
4. Przejść do edycji zmiennej globalnej (przez dwukrotne kliknięcie ikony zmiennej globalnej lub polecenie Open Front Panel z menu kontekstowego). Zostanie otworzony plik programu wyposażony jedynie w panel czołowy.



5. Na panelu czołowym zamieścić:
- wykres Waveform graph (Controls /Modern /Graph) i nadać mu nazwę „Przebieg czasowy sygnału”,
- przycisk logiczny Stop (Controls /Modern /Boolean) i nadać mu nazwę „Anuluj”,
- cztery zmienne liczbowe Numeric Control (Controls /Modern /Numeric) i nadać im nazwy „Częstotliwość”, „Amplituda”, „Przesunięcie” i „Poziom zakłóceń”.
- kopiować obiekt wyliczeniowy „Typ sygnału” z panelu programu (084_GeneratorA Engine.vi) na panel czołowy zmiennej globalnej.

6. Zapisać plik zmiennej globalnej pod nazwą (084_GeneratorC Global.vi). Zamknąć okno panelu zmiennej globalnej.
7. Przejść do schematu blokowego aplikacji (084_GeneratorA Engine.vi). Wybrać lewym przyciskiem myszy obiekt zmiennej globalnej. Na rozwijalnej liście powinny być dostępne wszystkie zmienne, zamieszczone uprzednio na panelu czołowym zmiennej globalnej. Cienka ramka, wejście i strzałka umieszczona z lewej strony symbolu zmiennej lokalnej wskazują, że jest to zmienna do zapisu (magazynowania) wartości.
8. Skopiować symbol zmiennej globalnej. Za pomocą polecenia Change To Read z menu kontekstowego zmienić charakter jednej ze zmiennych.
9. Zmienną „do zapisu” skonfigurować tak, aby dotyczyła „Przebiegu czasowego” (lewy przycisk myszy) i przyłączyć do źródła sygnału węzła Build Waveform.

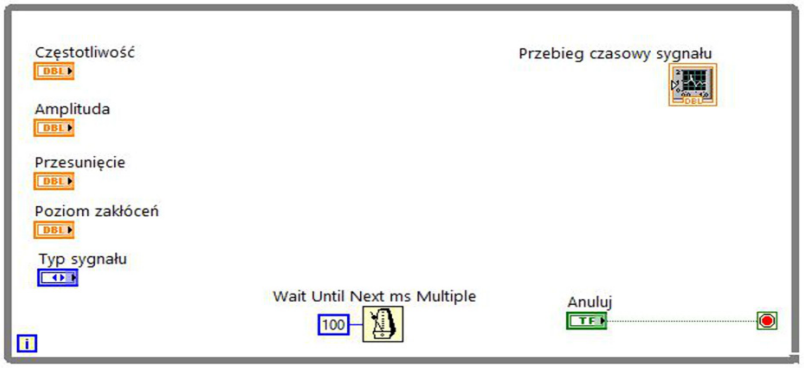


10. Zmienną „do odczytu” skopiować jeszcze pięć razy.
11. Zmieniając wartości zmiennej globalnej (lewy przycisk myszy) zastąpić zmienne wejściowe:

- „Częstotliwość”,
- „Amplituda”,
- „Przesunięcie”,
- „Poziom zakłóceń”,
- „Typ sygnału”,
- „Anuluj”.

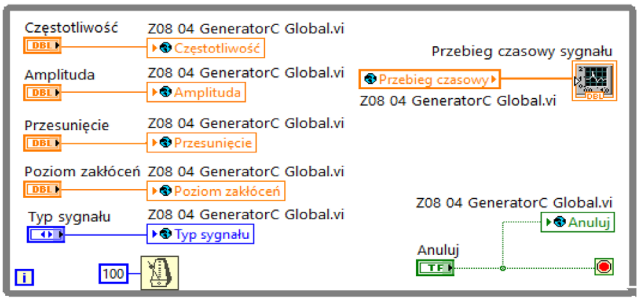
12. Usunąć nadmiarowe elementy przyłączone do terminalu warunkowego pętli tak, aby terminal warunkowy pętli sterowany był zmienną globalną „Anuluj”.
13. Zapisać plik pod aktualną nazwą (084_GeneratorA Engine.vi).

14. **Opracowanie programu do sterowania i prezentacji danych**. Otworzyć plik programu (084_GeneratorB Control.vi). Przejść do schematu blokowego.
15. Usunąć praktycznie wszystkie elementy poza pętlą While i obiektami sterującymi przebiegiem i wykresem czasowym.



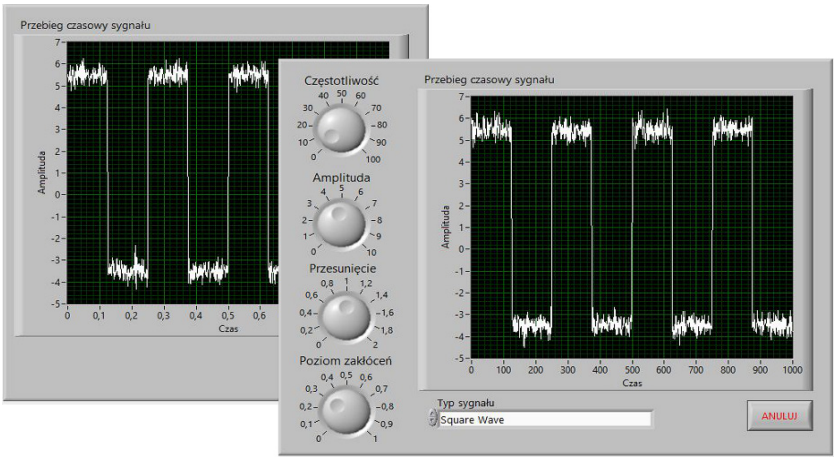
16. Wewnątrz pętli zamieścić obiekt zmiennej globalnej. Wprowadzenie zmiennej globalnej wykonuje się tak, jak zamieszczenie pliku podprogramu, czyli możliwe są poniższe działania:
- za pomocą polecenia Select a VI z palety Functions należy wybrać plik zmiennej globalnej z zasobów dyskowych,
- jeśli widoczne jest okno eksploratora plików i plik programu zmiennej globalnej, to możliwe jest przeciągnięcie ikony pliku programu zmiennej globalnej na schemat blokowy,
- jeśli otwarty jest panel programu zmiennej globalnej, to możliwe jest przeciągnięcie ikony w prawym górnym rogu panelu programu zmiennej na schemat blokowy,
- jeśli otwarty jest plik innego programu z poszukiwaną zmienną, to możliwe jest skopiowanie obiektu zmiennej globalnej z tego programu.

17. Zmieniając wartości zmiennej globalnej (lewy przycisk myszy) i charakter zmiennej (odczyt lub zapis) wprowadzić zmienne:
- „Częstotliwość”,
- „Amplituda”,
- „Przesunięcie”,
- „Poziom zakłóceń”,
- „Typ sygnału”,
- „Anuluj”.



18. Zapisać plik pod aktualną nazwą (084_GeneratorB Control.vi).

19. **Kontrola funkcjonowania program**ów. Wyświetlić obok siebie okna programów (084_GeneratorA Engine.vi) i (084_GeneratorB Control.vi).
20. Uruchomić program (084_GeneratorA Engine.vi).
21. Uruchomić program (084_GeneratorB Control.vi).
22. Zmieniać parametry sygnału w oknie programu (084_GeneratorB Control.vi). Wartości zmiennych sterujących przekazywane są za pomocą zmiennych globalnych do programu (084_GeneratorA Engine.vi). Wygenerowany przebieg przekazywany jest za pomocą zmiennej globalnej do wykresu czasowego programu (084_GeneratorB Control.vi).



23. Zakończyć pracę programu (084_GeneratorB Control.vi) przyciskiem Anuluj. Program powinien zakończyć działanie w tym samym czasie, co program (084_GeneratorA Engine.vi). Wartość przycisku Anuluj programu (084_GeneratorB Control.vi) została przypisana terminalowi warunkowemu pętli programu (084_GeneratorA Engine.vi) za pomocą zmiennej globalnej, co spowodowało zakończenie pracy programu.
24. Zamknąć pliki wszystkich programów (i zmiennej globalnej, jeśli plik był otwarty).

8.3.4. Wymiana danych za pomocą mechanizmu DataSocket

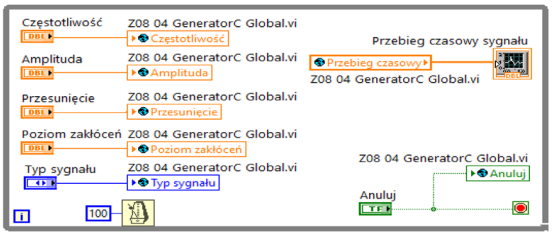
Cel zadania: wprowadzenie do stosowania protokołu do wymiany danych pomiędzy programami i wykorzystania go do przesyłania danych sterujących.

Zakres zadania: na podstawie uprzednio opracowanych aplikacji zostaną wykonane dwa programy. Pierwszy z nich będzie odpowiedzialny za deklarowanie parametrów generowanego przebiegu zmiennej odpowiedzialnej za równoczesne zakończenie obu aplikacji. Drugi z programów będzie realizował generację sygnału. Wymiana danych realizowana będzie za pomocą mechanizmu DataSocket.

1. **Przygotowanie plików programów**. Odnaleźć wśród utworzonych uprzednio plików pliki aplikacji (084_GeneratorB Control.vi) oraz (084_GeneratorA Engine.vi). Wykonać kopie plików aplikacji i nadać im nazwy:

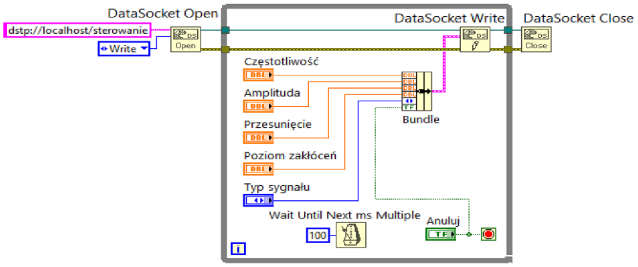
- (084_GeneratorB Control.vi) z nową nazwą (085_ADSocket Sterowanie.vi),
- (084_GeneratorA Engine.vi) z nową nazwą (085_BDSocket Przebieg).vi.

2. **Opracowanie programu sterującego**. Otworzyć plik programu (085_ADSocket Sterowanie.vi). Przejść do schematu blokowego.



3. Z pętli While usunąć wszystkie zmienne globalne oraz obiekt wykresu o etykiecie „Przebieg czasowy sygnału”.
4. Zamieścić węzeł funkcji klastrowej Bundle (Functions /Programming /Cluster, Class & Variant).
5. Poszerzyć ikonę węzła Bundle tak, aby można było przyłączyć sześć zmiennych wejściowych. Do wejść przyłączyć zmienne:

- „Częstotliwość”,
- „Amplituda”,
- „Przesunięcie”,
- „Poziom zakłóceń”,
- „Typ sygnału”,
- „Anuluj”.



6. Z subpalety DataSocket (Functions /Data Communications /DataSocket) zamieścić węzły:

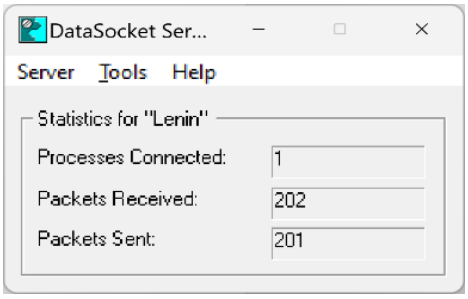
- DataSocket Open,
- DataSocket Write,
- DataSocket Close.

7. Konfiguracja węzła DataSocket Open. Minimalnym wymaganiem przy konfiguracji węzła DataSocket Open jest wskazanie serwera protokołu dstp wraz z nazwą, pod którą dostępne będą dane oraz tryby pracy węzła. Taką konfigurację zapewnia postępowanie zgodne z poniższymi wskazówkami.

- Zamieścić stałą łańcuchową String Constant (Functions /Programming /String).

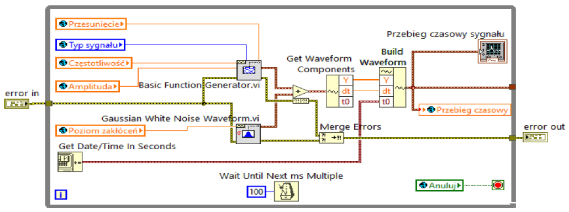
- Wprowadzić adres URL zgodnie z formatem *protokół://serwer/dane*. W przypadku wykonywania ćwiczenia na pojedynczej stacji roboczej (tj. programie sterującym, sterowanym i serwerze protokołu dstp na tym samym komputerze) można wskazać własną stację roboczą za pomocą adresowania localhost lub 127.0.0.1. W przypadku testowania pomiędzy różnymi jednostkami należy wcześniej sprawdzić:
  - jaki jest adres komputera obsługującego serwer dstp,
  - czy zabezpieczenie firewalla nie będzie blokowało protokołu dstp,
  - czy program antywirusowy nie będzie blokował serwera i protokołu dstp.
- Zadeklarować nazwę, pod którą dane będą przesyłane i publikowane na serwerze (na nazwę musi składać się dowolny ciąg znaków akceptowany przez system operacyjny). W przypadku bieżącego przykładu zastosowano adres *dstp://localhost/sterowanie*.
- Dla terminala „mode" węzła DataSocket Open zadeklarować stałą (menu kontekstowe /Create /Constant), a z listy rozwijalnej stałej wybrać „Write".

8. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu (oraz rysunkiem).
9. Zapisać plik programu pod aktualną nazwą (085_ADSocket Sterowanie.vi).

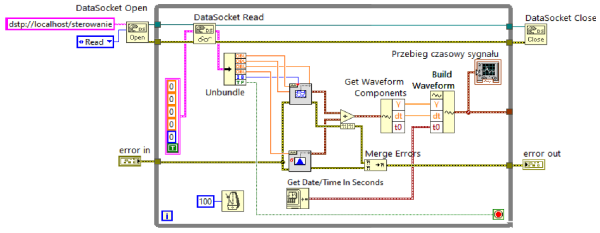


10. Uruchomić serwer DataSocket. Przy typowej instalacji środowiska LabVIEW skrót będzie się znajdował w menu Start – Start /Wszystkie programy /National Instruments /DataSocket – DataSocket Server.
11. Uruchomić aplikację (085_ADSocket Sterowanie.vi).
12. Sprawdzić, czy zmienne „Processes Connected", „Packets Received" oraz „Packets Sent" zmieniają się wraz z czasem działania programu (Z08 05 ADSocket Sterowanie.vi).
13. Zakończyć działanie aplikacji (085_ADSocket Sterowanie.vi) i zamknąć wszystkie jej okna.

14. **Opracowanie programu sterowanego**. Otworzyć plik programu (085_BDSocket Przebieg.vi). Przejść do schematu blokowego.
15. Z pętli While usunąć wszystkie zmienne globalne.



16. Zamieścić węzeł funkcji klastrowej UnBundle (Functions /Programming /Cluster, Class & Variant).



17. Zamieścić stałą klastrową Cluster Constant (Functions /Programming /Cluster, Class & Variant). Do powłoki stałej klastrowej wprowadzić:

- cztery stałe liczbowe zmiennoprzecinkowe DBL Numeric Constant (Functions / Programming /Numeric),
- jedną stałą liczbową całkowitą Numeric Constant (Functions /Programming / Numeric),
- jedną stałą logiczną Numeric Constant (Functions /Boolean).

Kolejność stałych w klastrze powinna być zgodna z kolejnością typów danych wprowadzanych do funkcji Bundle w poprzednim programie (Z08 05 ADSocket Sterowanie.vi).

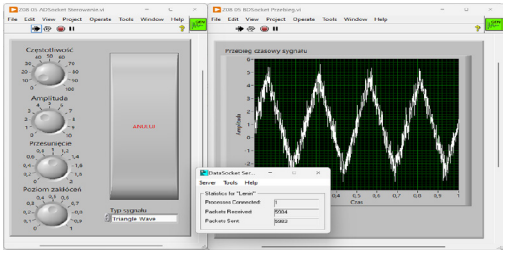
18. Z subpalety DataSocket (Functions /Data Communications /DataSocket) zamieścić na schemacie blokowym węzeł DataSocket Read.
19. Do terminala wejściowego „type" węzła DataSocket Open przyłączyć stałą klastrową, zaś do terminala wyjściowego „data" węzeł UnBundle.
20. Z subpalety DataSocket (Functions /Data Communications /DataSocket) zamieścić węzły: DataSocket Open, DataSocket Close.
21. Konfiguracja węzła DataSocket Open musi być zgodna z konfiguracją węzła DataSocket Open używanego do publikacji danych na serwerze. Minimalnym wymaganiem jest wskazanie serwera protokołu dstp wraz z nazwą, pod którą dostępne będą dane oraz tryby pracy węzła. Pożądany efekt można uzyskać, postępując zgodnie z poniższymi wskazówkami:

- Zamieścić stałą łańcuchową String Constant (Functions /Programming /String). Wprowadzić adres URL zgodnie z formatem *protokół://serwer/dane*. W przypadku bieżącego przykładu zastosowano adres *dstp://localhost/sterowanie*.

- Dla terminala „mode” węzła DataSocket Open zadeklarować stałą (menu kontekstowe /Create /Constant), a z listy rozwijalnej stałej wybrać „Write”.

22. Wykonać połączenia pomiędzy elementami kodu zgodnie z realizowanymi funkcjami programu.
23. Zapisać plik programu pod aktualną nazwą (085_BDSocket Przebieg.vi).

24. **Kontrola funkcjonowania program**ów. Wyświetlić obok siebie okna programów (085_ADSocket Sterowanie.vi) i (085_BDSocket Przebieg.vi).
25. Uruchomić serwer DataSocket, jeśli został zatrzymany (Start /Wszystkie programy /National Instruments /DataSocket – DataSocket Server).
26. Uruchomić program (085_ADSocket Sterowanie.vi).
27. Uruchomić program (085_BDSocket Przebieg.vi).



28. Zmieniać parametry sygnału w oknie programu (085_ADSocket Sterowanie.vi). Wartości zmiennych sterujących przekazywane są za pomocą serwera i protokołu dstp do programu (085_BDSocket Przebieg.vi). Program (085_BDSocket Przebieg.vi) generuje i prezentuje przebieg oparty na wartościach z programu (085_ADSocket Sterowanie.vi).
29. Zakończyć pracę programu (085_ADSocket Sterowanie.vi) przyciskiem Anuluj. Program powinien zakończyć działanie w tym samym czasie co program (085_ BDSocket Przebieg.vi).
30. Zamknąć pliki wszystkich programów oraz zakończyć pracę serwera DataSocket.

8.4. Pytania kontrolne

1. Jaka powinna być kolejność kroków prowadzących do wykonania kodu w środowisku LabVIEW?
2. Dlaczego należy unikać stosowania zmiennych lokalnych i globalnych?
3. Czy do odczytania wartości dystrybuowanych za pomocą protokołu dstp konieczne jest stosowanie odrębnego węzła DataSocket Read?

9. Zdalny nadzór nad procesami kontrolno-pomiarowymi w środowisku LabVIEW. Generowanie wykonywalnych plików aplikacji

9.1. Cel zajęć

Rozdział zawiera treści związane z publikowaniem danych pomiarowych w internecie oraz budową rozbudowanych systemów kontrolno-pomiarowych, w tym tworzeniem zdalnego front panelu programu. W rozdziale omówione zostanie przygotowanie złożonych projektów, sprawdzanie prawidłowego funkcjonowania podprogramów w ramach całego projektu, zapoznanie się z funkcjami ułatwiającymi obsługę projektów aplikacji oraz tworzenie wykonywalnego pliku samodzielnej aplikacji.

Celem zajęć jest w szczególności:

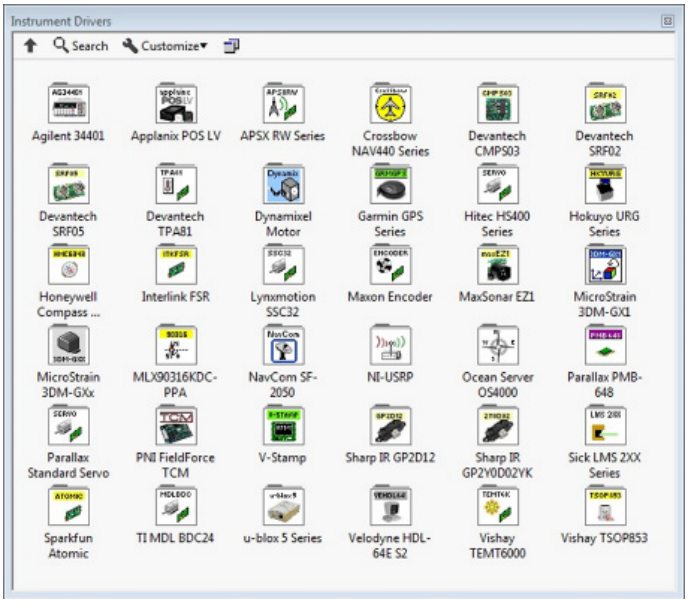
- poznanie zasad komunikacji między jednostką zarządzającą a układami odpowiadającymi za akwizycję danych pomiarowych,
- poznanie możliwości tworzenia procedur związanych z transmisją danych w podstawowych standardach i za pomocą najczęściej używanych protokołów,
- opracowanie aplikacji prezentującej możliwości użycia zdalnego pulpitu i tworzenia rozproszonych układów kontrolno-pomiarowych,
- tworzenie projektu aplikacji,
- budowa programu z plikami wykonywalnymi.

9.2. Wstęp

9.2.1. Komunikacja ze sprzętem pomiarowym

Komunikacja między elementami systemu kontrolno-pomiarowego może odbywać się za pomocą różnych środków transmisyjnych, różnych standardów oraz różnych protokołów komunikacyjnych.

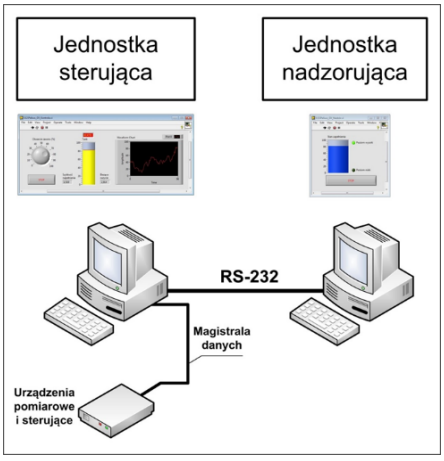
W środowisku LabVIEW najpopularniejszym środkiem przekazywania danych i informacji są magistrale danych skonfigurowane z kartami pomiarowymi (USB, PCI) lub dedykowanymi urządzeniami sterującymi procesami kontrolno-pomiarowymi (CompactRIO), ale także urządzenia sieciowe i moduły WiFi, GSM. W celu szybszego tworzenia procedur odpowiadających za komunikację między komputerem a sprzętem pomiarowym producenci sprzętu pomiarowego udostępniają sterowniki poszczególnych urządzeń, a w środowisku LabVIEW istnieją odpowiednio skonstruowane moduły funkcyjne. Po zainstalowaniu odpowiednich pakietów danych w zakładce Instrument Drivers palety Functions można znaleźć odpowiedni moduł (rys. 9.1) [2, 17].



Rys. 9.1. Paleta Instrument Drivers – zdefiniowane sterowniki
i moduły funkcyjne sprzętu pomiarowego

Do przesyłania danych między zewnętrznym sprzętem pomiarowym w środowisku LabVIEW wykorzystuje się ciągle popularne interfejsy komunikacyjne, takie jak RS-232, RS-485, GPIB.

Na rysunku 9.2 przedstawiono budową oraz wewnętrzne powiązania w systemach kontrolno-pomiarowych o charakterze rozproszonym.



Rys. 9.2. Wieloelementowy system kontrolno-pomiarowy o charakterze rozproszonym

Interfejsy komunikacyjne to urządzenia lub układy elektroniczne przeznaczone do łączenia pod względem fizycznym i programowym różnych urządzeń mikroprocesorowych (komputerów i mikrokontrolerów). Interfejsy komunikacyjne obsługują przepływ danych miedzy elementami systemu transmisji danych, sterują przepływem informacji. Interfejsy komunikacyjne działają według założonych standardów. Definiują one zarówno postać sprzętową (budowę układu), jak i programową (protokół komunikacyjny).

Zasady (standardy RS – recommended standard) określają liczbę elementów w systemie, prędkość i sposób przesyłania danych oraz reguły współpracy jednostek układu.

Interfejs komunikacyjny RS-232 jest magistralą komunikacyjną przeznaczoną do szeregowej transmisji danych na niewielkie odległości i praktycznie między dwoma jednostkami [2, 17].

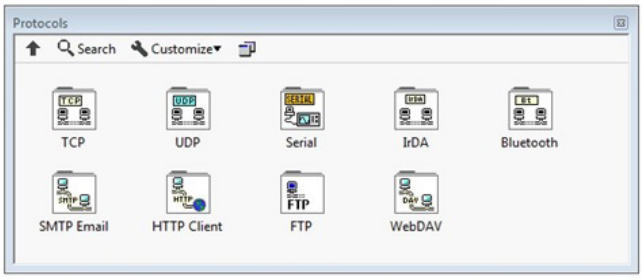
Interfejs komunikacyjny RS-485 (EIA-485) jest przemysłową magistralą komunikacyjną służącą do obsługi informacji w systemie wieloelementowym (do 32) i na znaczne odległości. Można budować w miarę rozległe systemy transmisji danych (do 1000m) [2, 17].

Interfejs komunikacyjny GPIB (IEEE 488) to magistrala przeznaczona do równoległej transmisji danych na niewielkie odległości. Standard powszechnie stosowany w automatycznych systemach pomiarowych i systemach kontrolno-pomiarowych [2, 17].

Paleta Instrument I/O (rys. 9.3) zawiera przyrządy I/O oraz funkcje umożliwiające budowę połączeń z przyrządami obsługującymi interfejsy GPIB, szeregowymi, modułowymi, PXI i innymi typami przyrządów. Czasami wymaga doinstalowania konkretnego sterownika do obsługi danego urządzenia.

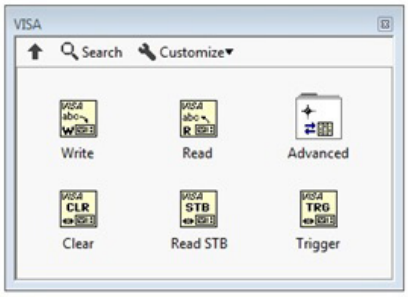
Rys. 9.3. Paleta Instrument I/O w środowisku LabVIEW

Paleta Protocols (rys. 9.4) zawiera moduły i funkcje pozwalające na wymianę danych przez interfejsy urządzeń przy użyciu protokołów, takich jak TCP/IP, UDP, porty szeregowe, IrDA, Bluetooth, SMTP, HTTP i FTP.



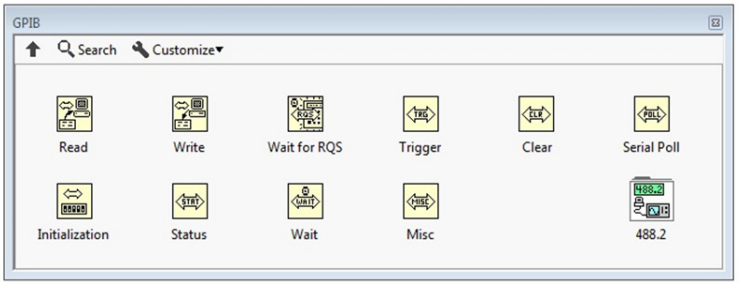
Rys. 9.4. Paleta Protocols w środowisku LabVIEW

VISA (Virtual Instrument Software Architecture) – powszechnie stosowany standard umożliwiający komunikację komputerów z urządzeniami kontrolno-pomiarowymi (patrz rys. 9.5).



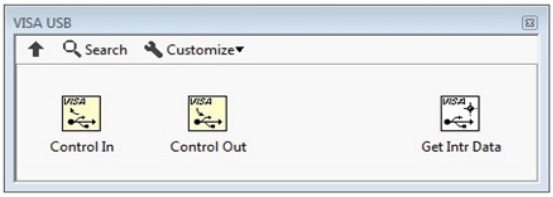
Rys. 9.5. Paleta VISA i dostępne w niej moduły funkcyjne

GPIB (General Purpose Interface Bus) – magistrala komunikacyjna ogólnego przeznaczenia – inny powszechnie stosowany standard zapewniający komunikację komputerów z urządzeniami kontrolno-pomiarowymi (rys. 9.6).



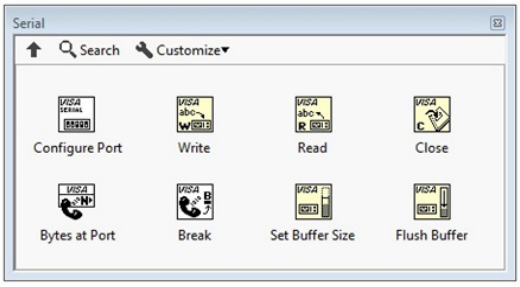
Rys. 9.6. Paleta GPIB i dostępne w niej moduły funkcyjne

USB (Universal Serial Bus) – uniwersalna magistrala szeregowa – interfejs, który zastępuje klasyczne porty szeregowe i równoległe, umożliwia komunikację między urządzeniami mikroprocesorowymi (rys. 9.7). Powszechnie stosowany w systemach pomiarowych do przesyłania danych.



Rys. 9.7. Paleta VISA USB w środowisku LabVIEW

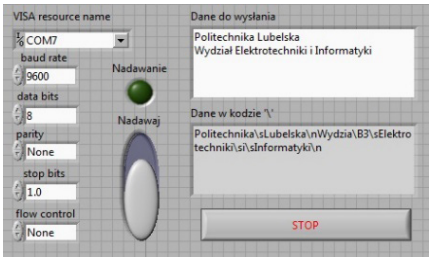
Paleta obsługi portów szeregowych Serial (rys. 9.8) – zawiera podstawowe narzędzie, służące do budowy procedur programowych umożliwiających przesyłanie i odczyt danych realizowanych za pomocą standardowych portów szeregowych ogólnych i przemysłowych, zwłaszcza RS-232 i RS-485.



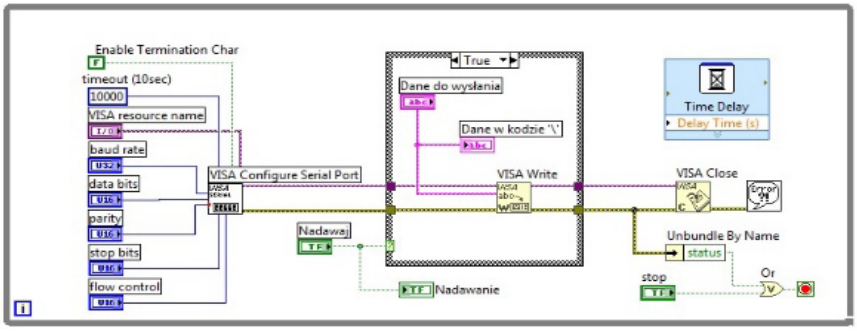
Rys. 9.8. Paleta Serial z dostępnymi modułami funkcyjnymi

W przypadku zastosowania wybranego standardu komunikacyjnego należy z dostępnych modułów funkcyjnych przygotować procedurę transmisji i odbioru danych, a następnie zintegrować ją z algorytmem tworzącym informacje do przesyłu lub dekodującym przesyłane dane.

Procedura transmisji danych za pomocą interfejsu RS-232 – nadawanie. Na rysunku 9.9 przedstawiony został front panel programu z zamieszczonymi elementami konfigurującymi proces komunikacyjny. Natomiast na rysunku 9.10 widoczny jest schemat blokowy procedury umożliwiającej przesyłanie danych z użyciem interfejsu RS-232.

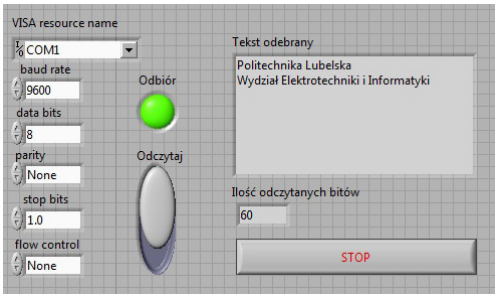


Rys. 9.9. Front panel aplikacji do nadawania sygnału w standardzie RS-232

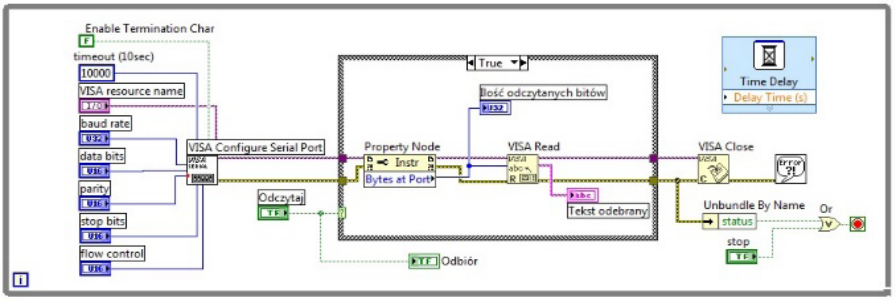


Rys. 9.10. Procedura nadawania w standardzie RS-232

Procedura przesyłu danych za pomocą interfejsu RS-232 – odbiór. Na rysunku 9.11 mamy front panel programu z zamieszczonymi elementami konfigurującymi proces komunikacyjny. Natomiast na rysunku 9.12 przedstawiony został schemat blokowy procedury umożliwiającej odbiór danych z użyciem interfejsu RS-232.



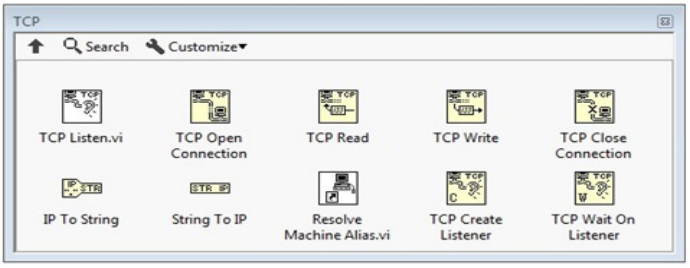
Rys. 9.11. Front panel aplikacji do nadawania sygnału w standardzie RS-232



Rys. 9.12. Procedura nadawania w standardzie RS-232

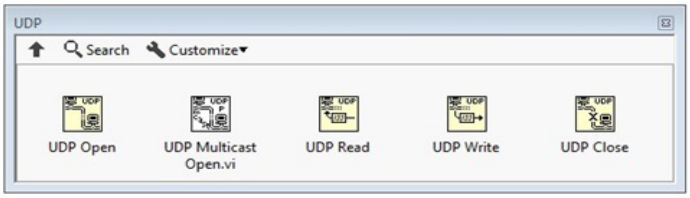
W środowisku LabVIEW istnieje możliwość użycia bardziej zaawansowanych technik transmisyjnych z wykorzystaniem standardów i interfejsów komunikacyjnych typu TCP/IP, UPD, IrDA lub Bluetooth.

TCP/IP (Transmission Control Protocol/Internet Protocol) – model warstwowej struktury protokołów komunikacyjnych. Podstawowy i powszechnie stosowany protokół do wymiany danych w sieciach komputerowych i sieciach przemysłowych (rys. 9.13).



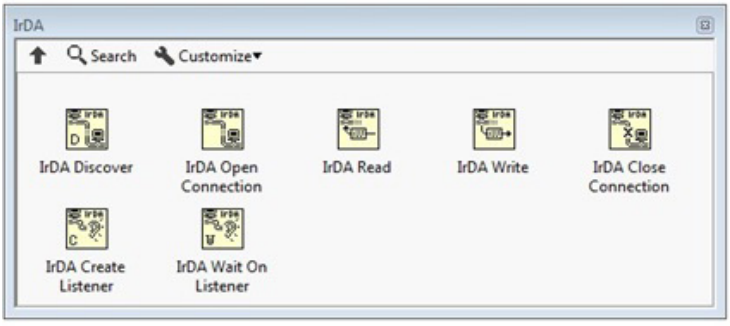
Rys. 9.13. Paleta TCP z dostępnymi modułami funkcyjnymi

UDP (User Datagram Protocol) – protokół pakietów użytkownika. Jeden z protokołów internetowych stosowany jest w warstwie transportowej modelu OSI. Używany do transmisji multimedialnych (rys. 9.14).



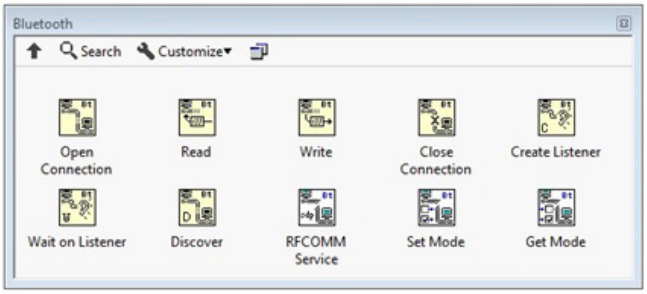
Rys. 9.14. Paleta protokołu UDP

IrDA (Infrared Data Association) – międzynarodowy standard transmisji danych w zakresie podczerwieni (850–900 nm). Bezprzewodowy system transmisji danych za pomocą fal elektromagnetycznych w zakresie podczerwieni (rys. 9.15). Wykorzystywany w transmisji bliskiej między urządzeniami.



Rys. 9.15. Paleta komunikacji IrDA z dostępnymi modułami funkcyjnymi

Bluetooth – standard bezprzewodowej komunikacji krótkiego zasięgu pomiędzy różnymi urządzeniami elektronicznymi (opisany w specyfikacji IEEE 802.15.1). Urządzenia pracują w paśmie ISM 2,4GHz. Teoretyczny zasięg Bluetooth jest uzależniony od klasy mocy do 1/10/100 m w przestrzeni otwartej. Dostępne w środowisku moduły konfiguracyjne zostały przedstawione na rysunku 9.16.



Rys. 9.16. Paleta narzędzi protokołu Bluetooth

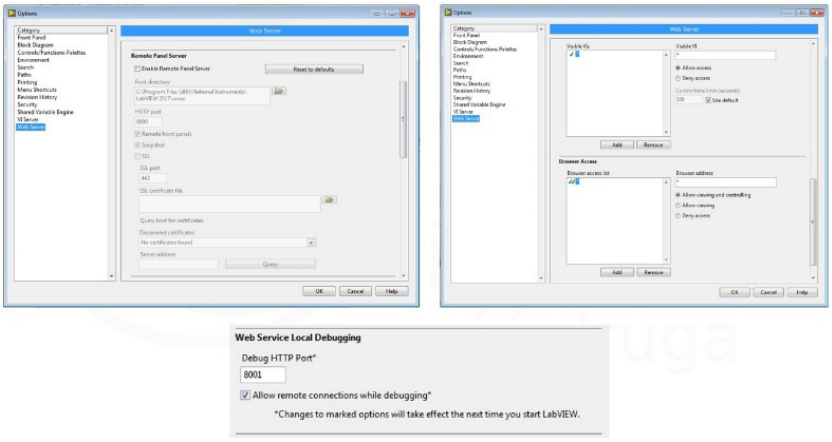
9.2.2. HTTP Serwer i zdalny dostęp do aplikacji

HTTP Web Serwer w LabVIEW pozwala na uzyskanie niezależności od platformy systemowej na danym komputerze. Transmisja danych odbywa się za pomocą powszechnego protokołu TCP/IP. Zastosowanie Web Serwera (HTTP) umożliwia budowę zdalnych pulpitów do zdalnego przeglądania i sterowania panelem czołowym [14].

Zdalny dostęp do programu jest możliwy za pomocą interfejsu LabVIEW (Connect to Remote Panel) i przeglądarki internetowej (np. IE, Firefox, Chrome). Na rysunku 9.17 został przedstawiony układ aplikacji i jej zdalny pulpit.

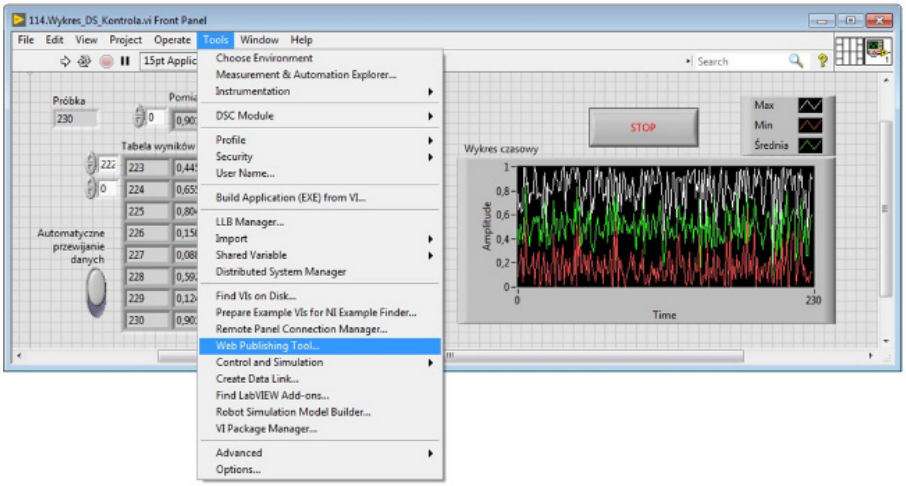
Rys. 9.17. Proces tworzenia zdalnych pulpitów aplikacji w środowisku LabVIEW

W celu zbudowania komunikacji umożliwiającej publikowanie zdalnego pulpitu niezbędna jest konfiguracja Web Serwera. Ta operacja możliwa jest w oknie dialogowym Options (wywoływane zakładką Tools /Options). Na rysunku 9.18 przedstawiono okna dialogowe służące do konfiguracji Web Serwera.



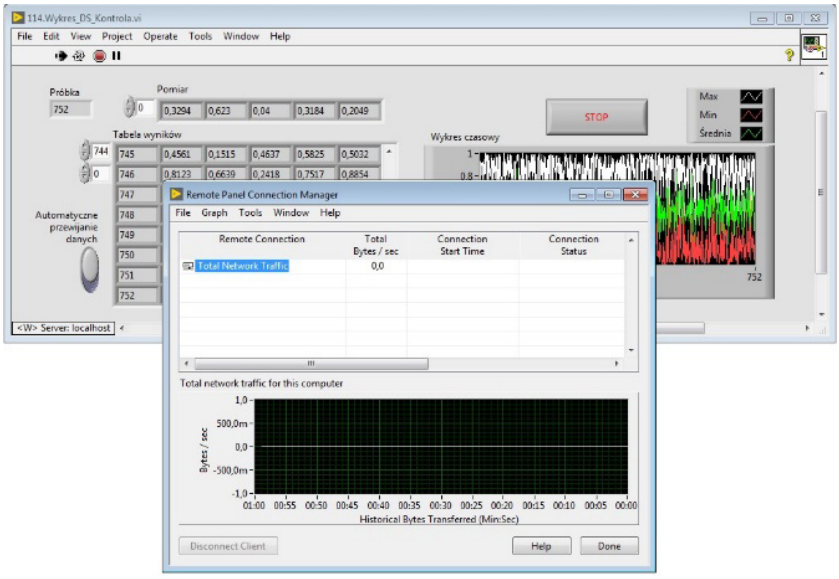
Rys. 9.18. Okna konfiguracyjne Web Serwer

Procedura publikowania front panelu aplikacji w przeglądarce internetowej związana jest z wywołaniem w środowisku LabVIEW funkcji Publishera www (rys. 9.19).



Rys. 9.19. Wywołanie funkcji użycia Publishera

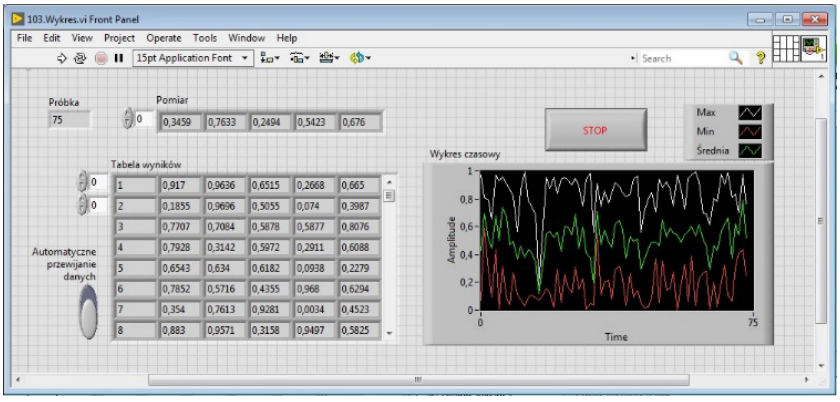
Spośród zasobów środowiska LabVIEW można również wykorzystać funkcję monitorowania klientów zdalnego pulpitu publikowanego w sieci www. W tym celu należy uruchomić funkcję Remote Panel Connection Manager (rys. 9.20).



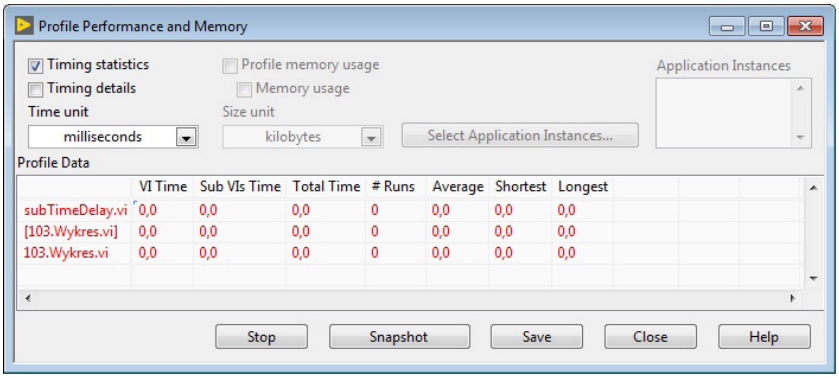
Rys. 9.20. Monitorowanie klientów (zakładka Tools /Remote Panel Connection Manager)

9.2.3. Planowanie aplikacji i generowanie plików wykonywalnych

Na prawidłowe funkcjonowanie programu duży wpływ ma odpowiednia organizacja procesu przetwarzania danych. Udostępnianie informacji na temat szybkości wykonania (zajętego czasu) i wymaganej/używanej pamięci dostępne są w oknie dialogowym Profile Performance and Memory zakładka Tools /Profile. Na rysunku 9.21 przedstawiono front panel aplikacji, natomiast na rysunku 9.22 okno dialogowe dotyczące bieżącego użycia pamięci przez tę aplikację.



Rys. 9.21. Front panel aplikacji



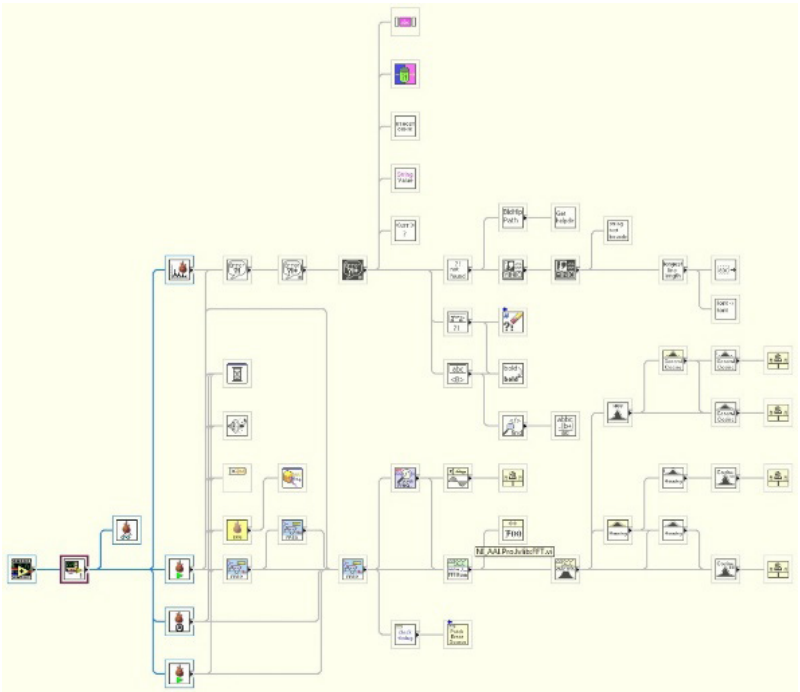
Rys. 9.22. Okno dialogowe Profile Performance and Memory

Ważną częścią procesu tworzenia aplikacji złożonej z wielu węzłów jest określanie powiązań oraz hierarchii poszczególnych elementów programu. W środowisku LabVEW możliwe jest poznanie powiązań i wzajemnej hierarchii elementów systemu poprzez użycie polecenia VI Hierarchy z menu View. Struktura powiązań dla programu przedstawionego na rysunku 9.23 została zaprezentowana na rysunku 9.24.

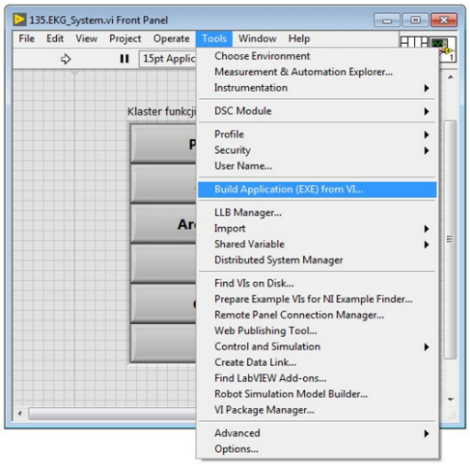
Rys. 9.23. Panel czołowy programu

Końcowym etapem budowy komputerowego systemu kontrolno-pomiarowego jest przygotowanie aplikacji z plikiem wykonywalnym. Takie rozwiązanie uniezależnia użytkownika od zainstalowania środowiska programistycznego, a programiście daje możliwość budowy zamkniętego programu obsługiwanego z wybranej platformy sprzętowej.

Wywołanie funkcji (narzędzia) „Build Application (EXE) from VI...” do tworzenia aplikacji z plikiem wykonywalnym .exe zostało przedstawione na rysunku 9.25.

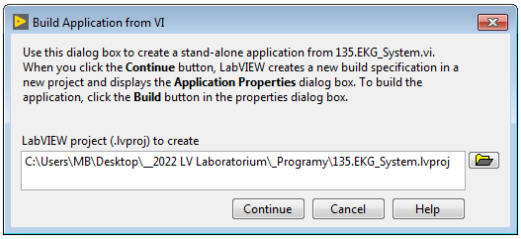


Rys. 9.24. Struktura powiązań programu w środowisku LabVIEW

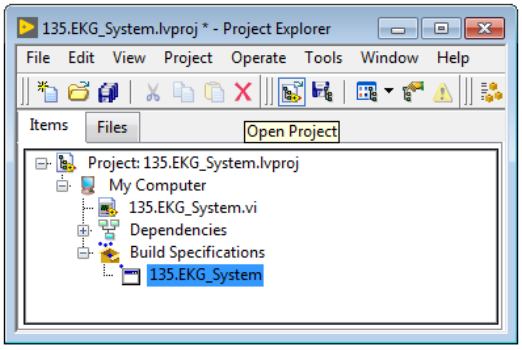


Rys. 9.25. Lokalizacja polecenia „Build Application (EXE) from VI...”

Uruchomienie edytora tworzenia aplikacji rozpoczyna się od okna komunikacyjnego zawierającego informację o lokalizacji pliku projektu z rozszerzeniem .lvproj (rys. 9.26) i wygenerowaniem pliku projektu oraz otworzeniem okna Project Explorer (rys. 9.27).



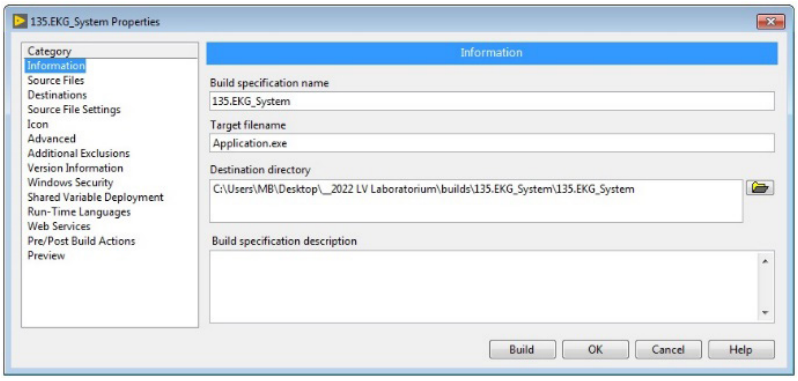
Rys. 9.26. Ono dialogowe z generatorem nazwy projektu



Rys. 9.27. Okno Project Explorer z przykładową strukturą projektu

Tworzenie aplikacji z plikiem wykonywalnym wiąże się z określeniem:

- nazwy biblioteki/pliku wykonywalnego (rys. 9.28),
- przygotowaniem ikony dla programu .exe (może być inna niż ikona .vi),
- lokalizacji katalogu docelowego i roboczego.

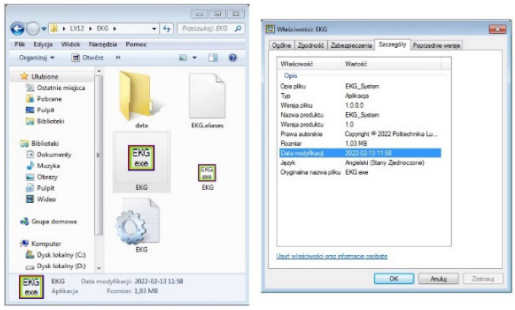


Rys. 9.28. Jedno z okien konfiguratora z polami dialogowymi służącymi do określania danych plikowych tworzonego programu z plikiem wykonywalnym

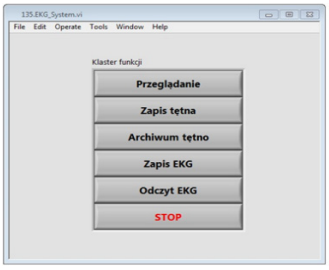
Rezultat przeprowadzonych czynności to aplikacja, funkcjonująca niezależnie od zainstalowanego środowiska LabVIEW, z plikiem wykonywalnym .exe oraz z plikami zawierającymi niezbędne sterowniki i biblioteki.

Zbudowany program zazwyczaj wyróżnia:

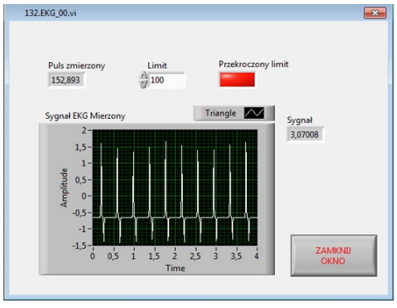
- katalog z plikami aplikacji (rys. 9.29),
- okno właściwości pliku .exe,
- okna aplikacji zawierające front panele programu głównego (rys. 9.30) oraz programów wywoływanych przez program główny (rys. 9.31).



Rys. 9.29. Katalog z plikami programu oraz właściwości pliku .exe aplikacji



Rys. 9.30. Front panel programu głównego aplikacji



Rys. 9.31. Front panel program wywoływanego z okna programu głównego

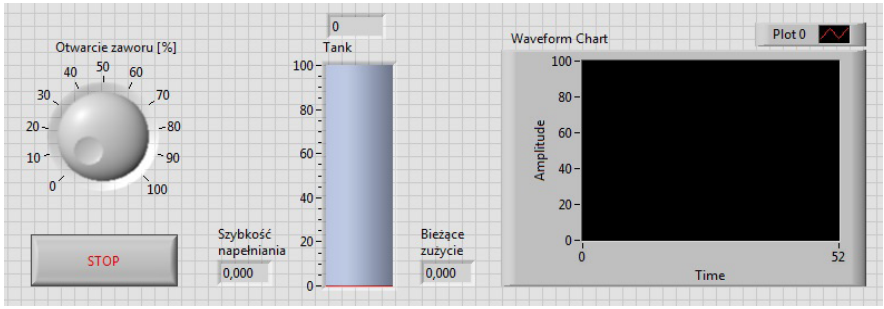
9.3. Zadania

9.3.1. Kontrola poziomu zapełnienia zbiornika. Zdalny nadzór nad aplikacją z poziomu przeglądarki internetowej

Cel zadania: opracowanie programu, który, dzięki zastosowaniu węzłów właściwości, informuje o aktualnym poziomie zapełnienia zbiornika przez zmianę koloru, a w przypadku stanów niebezpiecznych – przez miganie. Demonstracja możliwości użycia zdalnego dostępu do panelu programu LabVIEW za pomocą przeglądarki internetowej.

Zakres zadania: wykorzystanie węzłów właściwości do kontroli zarządzania wybranymi opcjami i funkcjonalnościami obiektów umieszczonych na front panelu. Użycie narzędzia „Web Publishing Tool" do uzyskania przez użytkownika zdalnego dostępu do panelu czołowego programu za pomocą przeglądarki internetowej i identyfikatora URL.

1. **Panel czołowy.** W trakcie ćwiczenia zostanie przygotowany program analizujący stan zapełnienia zbiornika.
2. Uruchomić program LabVIEW i otworzyć nowy plik programu.
3. Zbudować front panel programu tożsamy z przedstawionym na rysunku.



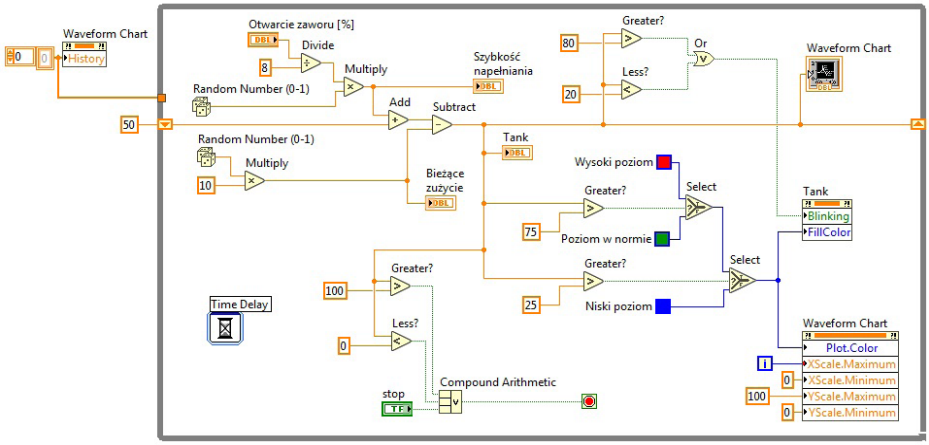
4. W celu wykonania front panelu zgodnego z założeniami programu należy zamieścić:
- kontrolkę numeryczną typu Knob. Kontrolce nadać nazwę „Otwarcie zaworu [%]",
- przycisk typu Stop Button. Ukryć jego etykietę (Label),
- wskaźnik numeryczny typu Tank. Wyświetlić dodatkowo jego Digital Display (użyć funkcji dostępnej w menu kontekstowym),
- dwa wskaźniki numeryczne Numeric Indicator. Nadać im nazwy: „Szybkość napełniania" i „Bieżące zużycie",
- wykres typu Waveform Chart.

5. Zapisać plik programu pod nazwą 091_Zbiornik.vi.

6. **Budowa schematu blokowego.** Wstawić w oknie schematu blokowego wymagane elementy. W tym celu pusty schemat blokowy należy uzupełnić o:

- pętlę While,
- przycisk [STOP] (STOP Button),
- dwa generatory liczb losowych Random Number (0–1),
- dwanaście stałych liczbowych Numeric Constant. Ustawić wartości w poszczególnych stałych w sposób analogiczny jak na rysunku,
- moduł dodawania Add,
- moduł odejmowania Subtract,
- dwa moduły mnożenia Multiply,
- moduł dzielenia Divide,
- moduł funkcyjny Compound Arithmetic. Ustawić w nim opcję realizacji sumy logicznej Or.
- cztery moduły większości Greather?,
- dwa moduły mniejszości Less?,
- moduł sumy logicznej Or,
- moduł funkcji wyboru Select,
- trzy stałe koloru Color Box Constant,
- moduł opóźnienia czasowego Time Delay. Ustawić wartość opóźnienia na jedną sekundę,
- węzeł właściwości obiektu Wykres czasowy. W tym przypadku ustawić właściwość History Data. Wybierając z menu kontekstowego węzła właściwości Wykresu czasowego funkcję Change All To Write zmienić sposób funkcjonowania węzła. Do terminala wejściowego właściwości History Data wygenerować stałą (polecenie Create /Constant z menu kontekstowego),
- węzeł właściwości obiektu Waveform Chart. Wybrać z listy właściwość Plot / Plot Color. Zmienić rozmiar węzła tak, aby pokazywał pięć (terminale) właściwości. Klikając w kolejne pole, wybrać z listy właściwości: Xscale /Range /Maximum, Xscale /Range /Minimum, YScale /Range /Maximum, YScale /Range/ Maximum. Wybierając z menu kontekstowego węzła właściwości tablicy Tabela wyników funkcję Change All To Write zmienić sposób funkcjonowania węzła.

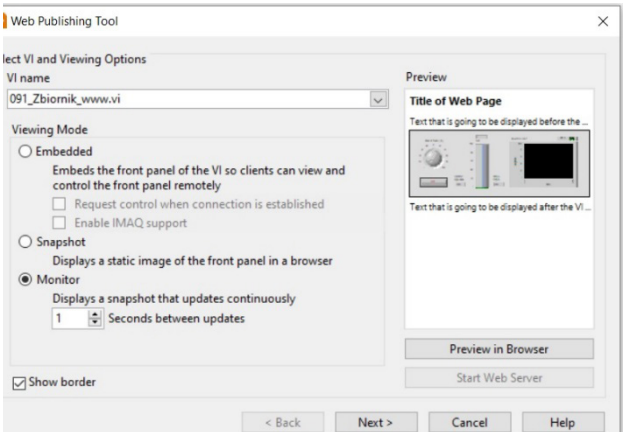
7. Rozmieścić poszczególne elementy na schemacie blokowym w sposób przejrzysty oraz wykonać stosowne połączenia analogicznie jak na rysunku.



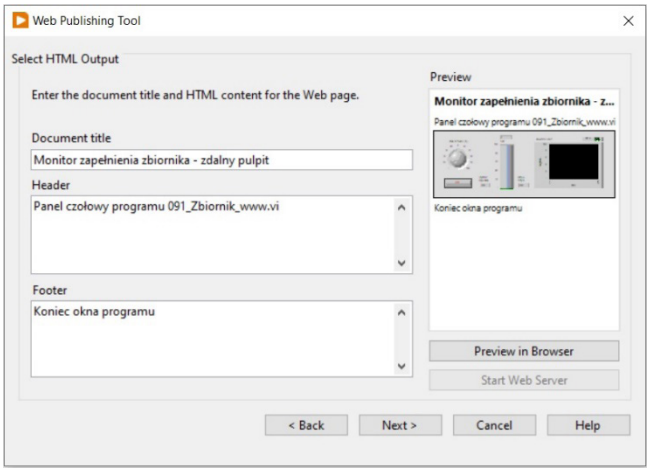
8. Sprawdzić, czy program nie zawiera błędów oraz ocenić poprawność wykonania schematu blokowego i rozmieszczenia elementów.
9. Zapisać wprowadzone czynności w pliku pod nazwą (091_Zbiornik,vi).

10. **Uruchamianie programu.** Przejść do okna panelu czołowego i uruchomić program.
11. Obserwować funkcjonowanie programu. Należy zwrócić uwagę na zmiany zachodzące w sposobie wyświetlania danych oraz w wyglądzie wybranych elementów przy osiągnięciu zadeklarowanych poziomów napełnienia zbiornika.
12. Program po pewnym czasie, po osiągnięciu pełnego napełnienia, zostanie wyłączony automatycznie. Zatrzymanie programu może się odbyć również poprzez użycie przycisku [STOP].
13. Po zatrzymaniu funkcjonowania programu zamknąć wszystkie aktywne okna programu LabVIEW.

14. **Konfiguracja serwera WWW.** Otworzyć ponownie plik programu 091_Zbiornik.vi i zapisać go pod nową nazwą 091_Zbiornik_www.vi.
15. W aktywnym oknie front panelu z menu Tools wybrać polecenie Web Publishing Tool.
16. W oknie dialogowym sprawdzić, czy przycisk Start Web Serwer jest akywny. Jeżeli tak, należy go wybrać.
17. W oknie konfiguracyjnym Select VI and Viewing Options narzędzia Web Publishing Tool wybrać z listy rozwijanej VI name plik 091_Zbiornik_www.vi (nazwa bieżącego programu), a następnie w polu Viewing Mode wybrać opcję Monitor.
18. Sprawdzić, czy wskaźnik Seconds between updates ma wartość 1 (ewentualnie zadeklarować wartość). Okno konfiguracyjne Select VI and Viewing Options narzędzia Web Publishing Tool przedstawiono na rysunku.

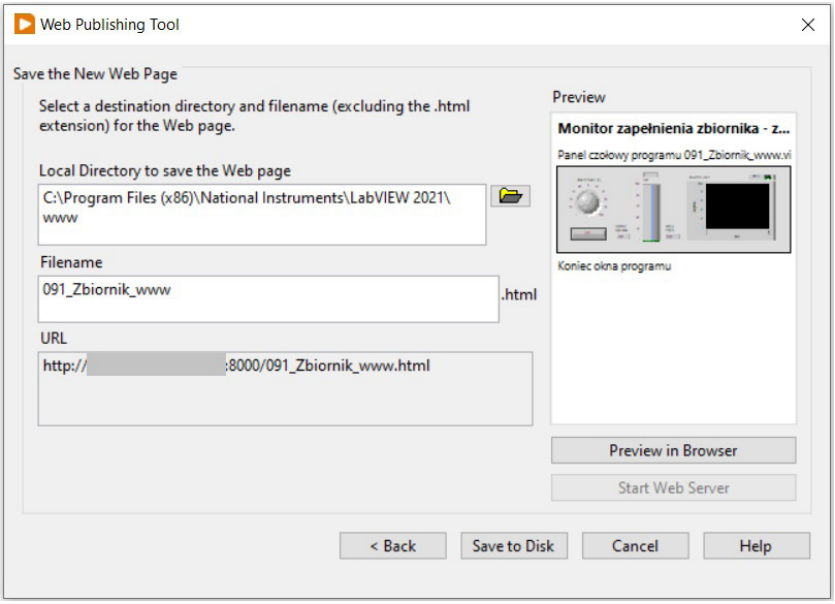


19. Wybrać przycisk Next i przejść do kolejnego okna konfiguracyjnego narzędzia Web Publishing Tool o nazwie Select HTML Output.
20. W oknie konfiguracyjnym Select HTML Output można dla dokumentu HTML ustawić tytuł i treść, które znajdą się na zdalnym front panelu w przeglądarce internetowej. W trzech polach tekstowych można ustawić następujące komunikaty: tytuł (Document title), nagłówek (Header) oraz stopkę (Footer). W polu Document title wpisz tekst: „Monitor zapełnienia zbiornika – Zdalny pulpit", w polu Header wpisać tekst: „Panel czołowy programu 091_Zbiornik_www.vi", a w polu Footer wpisać tekst: „Koniec okna programu". Okno konfiguracyjne Select HTML Output z zamieszczonymi komunikatami przedstawiono na rysunku.

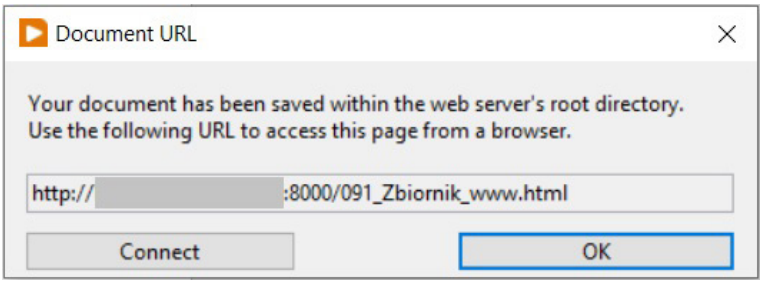


21. Wybrać przycisk Next i przejść do kolejnego okna konfiguracyjnego narzędzia Web Publishing Tool o nazwie Save the New Web Page.

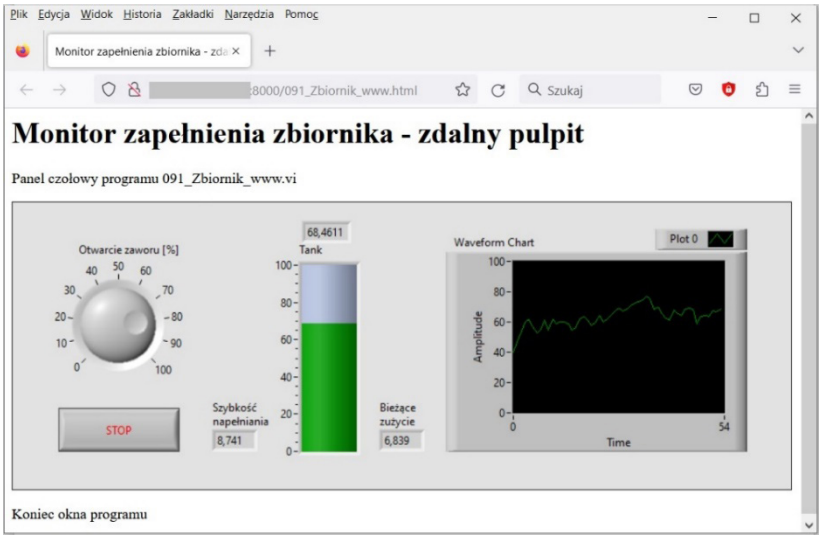
22. Okno konfiguracyjne Save the New Web Page służy do określenia katalogu docelowego i nazwy dla strony internetowej związanej z prezentacją zdalnego pulpitu programu w przeglądarce internetowej. W polach tekstowych Local Directory to save the Web page oraz Filname należy pozostawić automatyczne wygenerowane ustawienia. Okno konfiguracyjne Save a New Web Page, z przykładowymi ustawieniami katalogu docelowego i nazwy dla strony internetowej, przedstawiono na rysunku.



23. W ćwiczeniu mającym na celu powiązanie zdalnego pulpitu z działającym programem posłuży adres URL. Zanotuj wygenerowany w polu tekstowym URL adres URL. Będzie on potrzeby na późniejszym etapie ćwiczenia.
24. Za pomocą przycisku Save to Disk zapisać dokonane ustawienia (jeżeli pojawi się monit, że plik o danej nazwie już istnieje i pytanie o jego nadpisanie, należy potwierdzić chęć nadpisania). Zapisanie pliku na dysku komputera umożliwi późniejszą prezentację zdalnego pulpitu. Po kliknięciu przycisku Save to Disk otworzy się okno dialogowe o nazwie Document URL (patrz rysunek). W oknie tym zamieszczone są informacje o adresie URL, jakich należy użyć w celu dostępu do zdalnego pulpitu aplikacji z poziomu przeglądarki internetowej. Użycie przycisku Connect umożliwia automatyczne otworzenie zdalnego pulpitu w przeglądarce internetowej. Użycie przycisku [OK] zatwierdza proces tworzenia zdalnego pulpitu za pomocą narzędzia Web Publishing Tool. Należy użyć najpierw przycisku Connect, a po otwarciu przeglądarki internetowej z widocznym oknem programu 09_Zbiornik_www.vi użyć przycisku [OK].



25. Zapisać dokonane zmiany w programie pod bieżącą nazwą (091_Zbiornik_www.vi).
26. Otworzyć okno przeglądarki internetowej i wprowadzić do niej adres URL programu. W oknie przeglądarki internetowej zostanie wyświetlone okno front panelu programu 091_Zbiornik_www.vi wraz z wprowadzonymi wcześniej informacjami.



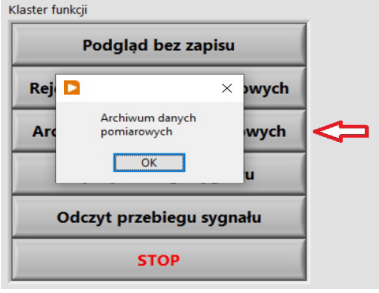
27. Uruchomić w środowisku LabView program 091.Zbiornik_www.vi. Obserwować okno przeglądarki internetowej. Powinien być w nim widoczny panel czołowy programu i powinna następować aktualizacja prezentowanych danych zgodna ze zmianami na front panelu programu głównego (w razie potrzeby użyć przycisku Odświeżania w przeglądarce internetowej).
28. Zakończyć działanie programu przyciskiem [STOP], a następnie zamknąć wszystkie okna programu.
29. Zamknąć aktywne okna przeglądarki internetowej.

9.3.2. Klaster Menu. Szkielet programu zarządzania

Cel zadania:	sprawdzenie gotowego programu maszyny stanu z zastosowaniem logicznego klastra sterującego do zarzadzania rozbudowaną aplikacją. Zbudowane menu umożliwia wywoływanie podprogramów realizujących funkcje związane z przetwarzaniem, archiwizacją i odczytem danych pomiarowych lub symulowanych.
Zakres zadania:	zapoznanie z pracą programu bazującego na maszynie stanu. Sprawdzenie działania aplikacji. Opanowanie procesu kontroli funkcjonowania algorytmów sterowania.

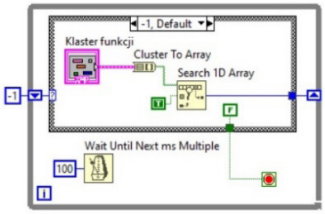
1. **Panel czołowy.** Odnaleźć na dysku i uruchomić program 192_Menu.vi (który jest jednym z gotowych programów). Panel czołowy otworzonego programu powinien być analogiczny z przedstawionym na rysunku.

2. Uruchomić program i sprawdzić, czy w reakcji na naciśnięcie przycisków pojawia się adekwatne działanie. W przypadku „czarnych" przycisków powinno to być wygenerowanie okna z komunikatem tekstowym (rysunek). W przypadku naciśnięcia przycisku „STOP" aplikacja powinna zakończyć swoje działanie.

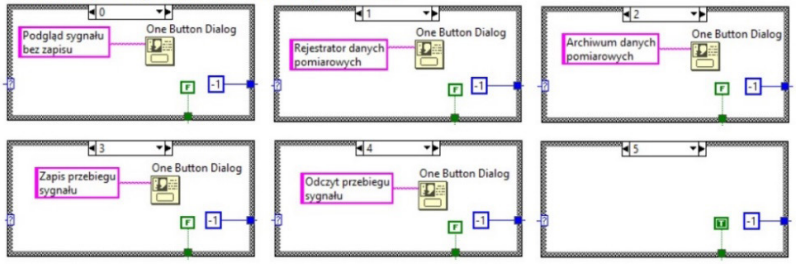


3. Jeżeli program funkcjonuje prawidłowo, należy zapisać go pod nową nazwą 092_Menu.vi. i przejść do analizy funkcjonowania algorytmu sterującego.

4. **Budowa i funkcjonowanie programu.** Uaktywnić okno schematu blokowego. Schemat blokowy programu został przedstawiony na rysunku.



Alternatywne przypadki struktury wyboru



5. Rozmieścić na ekranie monitora okna front panelu i schematu blokowego tak, żeby widoczne były wszystkie ich elementy.
6. Włączyć funkcję Highlight Execution i uruchomić działanie programu w trybie Run. Użyć wszystkich przycisków dostępnych na front panelu. Obserwować informacje generowane w oknie schematu blokowego i drogi przepływu strumienia danych w programie.
7. Po dokonaniu analizy zamknąć wszystkie aktywne okna LabVIEW.

9.3.3. Programy użytkowe systemu zarządzania danymi pomiarowymi

Cel zadania:	zapoznanie z działaniem przykładowych programów realizujących założone procedury i czynności.
Zakres zadania:	sprawdzenie funkcjonalności programów, które będą użyte jako Sub-VI w aplikacji głównej.

1. **Czynności kontrolne i sprawdzające.** Odnaleźć na dysku i uruchomić program 193_Paliwo_00.vi. Sprawdzić jego funkcjonowanie. Plik zawiera aplikację do zarządzania procesem napełniania zbiornika i kontrolowania go.
2. Jeżeli program funkcjonuje prawidłowo, zapisać go pod nową nazwą 093_Zbiornik_00.vi.

3. Odnaleźć na dysku i uruchomić program 193_Paliwo_01.vi. Sprawdzić jego funkcjonowanie. Plik zawiera aplikację do zarządzania procesem napełniania zbiornika i kontrolowania go oraz zapisuje w formie raportu (plik tekstowy) dane pomiarowe.
4. Jeżeli program funkcjonuje prawidłowo, zapisać go pod nową nazwą 093_Zbiornik_01.vi.
5. Odnaleźć na dysku i uruchomić program 193_Paliwo_02.vi. Sprawdzić jego funkcjonowanie. Plik to aplikacja umożliwiającą odczytanie danych z plików tekstowych zawierających informację o dokonanych wcześniej pomiarach (raporty tekstowe).
6. Jeżeli program funkcjonuje prawidłowo, zapisać go pod nową nazwą 093_Zbiornik_02.vi.
7. Odnaleźć na dysku i uruchomić program 193_Paliwo_03.vi. Sprawdzić jego funkcjonowanie. Plik zawiera aplikację do zarządzania procesem napełniania zbiornika i kontrolowania go oraz zapisuje do pliku binarnego przebieg sygnału EKG.
8. Jeżeli program funkcjonuje prawidłowo, zapisać go pod nową nazwą 093_Zbiornik_03.vi.
9. Odnaleźć na dysku i uruchomić program 193_Paliwo_04.vi. Sprawdzić jego funkcjonowanie. Plik zawiera aplikację umożliwiającą wyświetlenie na wykresie wcześniej zapisanych w plikach binarnych przebiegów sygnału EKG.
10. Jeżeli program funkcjonuje prawidłowo, zapisać go pod nową nazwą 093_Zbiornik_04.vi.
11. Upewnić się, że wszystkie programy zostały prawidłowo zapisane.
12. Zamknąć wszystkie aktywne okna programów środowiska LabVIEW.

9.3.4. Budowa aplikacji głównej

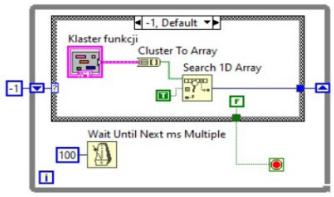
Cel zadania: łączenie programów SubVI ze szkieletem aplikacji głównej. Budowa rozbudowanych systemów kontrolno-pomiarowych służących do monitorowania i diagnostyki procesów technicznych. Przygotowanie pliku VI zawierającego aplikację końcową, realizującą określone cele użytkownika dotyczące funkcjonowania programu, służące do generowania sygnału, jego przetwarzania oraz rejestracji i prezentacji danych zapisanych na dysku.

Zakres zadania: uzupełnianie szkieletu aplikacji głównej przez programy realizujące określone zadania kontrolno-pomiarowe.

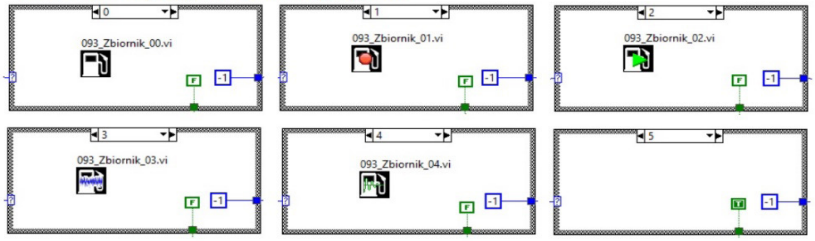
1. **Czynności wstępne.** Odnaleźć na dysku i uruchomić program 092_Menu.vi. Sprawdzić poprawność jego funkcjonowania.
2. Jeżeli program funkcjonuje prawidłowo, zapisać go pod nową nazwą 094_Aplikacja.vi.
3. Przejść do edycji schematu blokowego. Okno front panelu pozostaje bez zmian.

4. **Tworzenie nowego schematu blokowego.** Otworzyć okno schematu blokowego. Usunąć z ramek poszczególnych przypadków funkcje okna dialogowego One Button Dialog wraz z podłączonymi do nich stałymi łańcuchowymi zawierającymi komunikaty tekstowe.
5. Wstawić w obszar poszczególnych przypadków programy związane z realizacją założonych procesów:
- do ramki przypadku 0 wstaw pogram 093_Zbiornik_00.vi,
- do ramki przypadku 1 wstaw pogram 093_Zbiornik_01.vi,
- do ramki przypadku 2 wstaw pogram 093_Zbiornik_02.vi,
- do ramki przypadku 3 wstaw pogram 093_Zbiornik_03.vi,
- do ramki przypadku 4 wstaw pogram 093_Zbiornik_04.vi.

6. Po wykonaniu przedstawionych czynności schemat blokowy programu powinien być tożsamy z przedstawionym na rysunku.



Alternatywne przypadki struktury wyboru



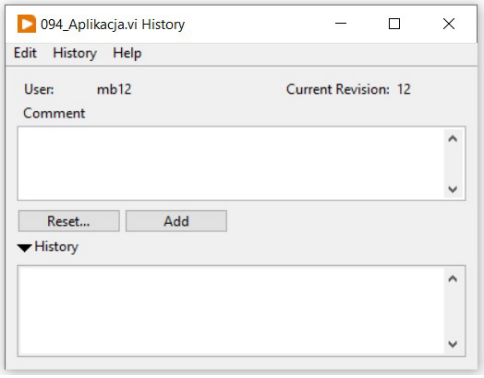
7. Przed uruchomieniem programu skonfiguruować właściwości wywołania wszystkich wstawionych do aplikacji podprogramów. Podprogramy te powinny mieć aktywne opcje Show front panel when called oraz Close afterwards if oryginaly closed (opcje te można ustawić w oknie dialogowym SubVI Nod Setup, wywoływanym poleceniem „SubVI Node Setup...” z menu kontekstowego poszczególnych podprogramów).
8. **Uruchomienie programu i kontrola funkcjonowania.** Uruchomić program i skontrolować poprawność jego funkcjonowania. Sprawdzić działanie programu w zakresie wywoływania poszczególnych procedur programowych i realizacji przez nie założonych funkcji. Pamiętać, aby otwierające się okna podprogramów zamykać stosownymi przyciskami, znajdującymi się na ich front panelach.
9. Utworzyć ikonę programu. Wykonać ikonę zbliżoną wyglądem do przedstawionej na rysunku.

10. Sprawdzić kilkukrotnie działanie aplikacji, wywołując wszystkie jej procedury.
11. Zapisać wprowadzone zmiany w programie w bieżącym pliku (094_Aplikacja.vi).

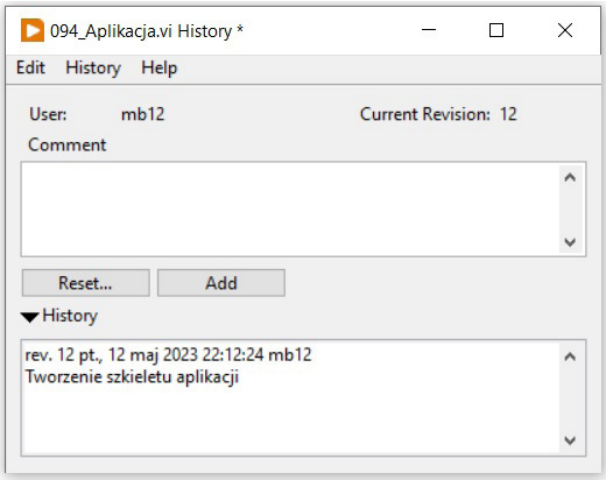
9.3.5. Narzędzia do zarządzania projektami w środowisku LabVIEW

Cel zadania:	poznanie wybranych wbudowanych funkcji w środowisku LabVIEW ułatwiających obsługę projektów tworzenia aplikacji.
Zakres zadania:	praktyczne wykorzystanie funkcji VI Revision History oraz VI Hierarchy.

1. Otworzyć plik programu 094_Aplikacja.vi. Upewnić się, że nie ma innych otwartych programów w środowisku LabVIEW.
2. W tym ćwiczeniu panel czołowy i blok diagram programu nie będą podlegać modyfikacjom. Sprawdzone będą funkcje ułatwiające obsługę tworzonych aplikacji. Tymi funkcjami będą VI Revision History... oraz VI Hierarchy.
3. **Zapoznanie z historią zmian – VI Revision History.** Z menu Edit wybrać polecenie „VI Revision History...". Funkcja ta umożliwia wyświetlenie okna z komentarzami tekstowymi użytkownika dotyczącymi wprowadzanych w programie zmian. Na bieżącym etapie otworzone okno nie powinno zawierać żadnych komentarzy i powinno być analogiczne do przedstawionego na rysunku. Jeżeli w polu tekstowym History znajdują się jakieś komentarze, to należy je usunąć. W tym celu należy skorzystać z przycisku Reset..., a następnie w wyświetlonym oknie dialogowym Reset Revision History zatwierdzić operację usuwania wpisów przyciskiem [OK].

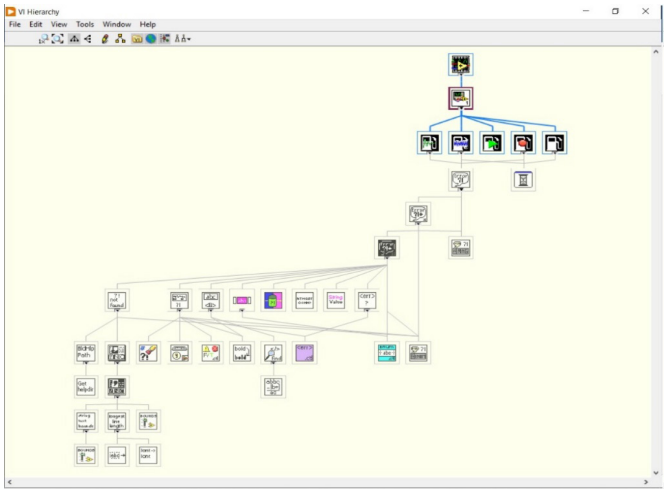


4. Wprowadzić w polu komentarzy Comment tekst „Tworzenie szkieletu aplikacji”, a następnie wybrać przycisk dodawania (Add). Komentarz „Tworzenie szkieletu aplikacji” powinien zostać wyświetlony na liście komentarzy w dolnej części okna, w polu tekstowym History wraz z numerem, datą i czasem oraz nazwą użytkownika.



5. Zamknąć okno historii komentarzy.

6. **Przeglądanie hierarchii elementów projektu.** Z menu View wybrać polecenie VI Hierarchy. Funkcja ta umożliwia wyświetlenie okna z hierarchicznie połączonymi składnikami programu (patrz rysunek).
7. Przetestować sposób rozwijania i zwijania połączeń w widoku hierarchicznym. Zwrócić uwagę, że kliknięcie małej czerwonej lub czarnej strzałki w węzłach hierarchii powoduje rozwinięcie lub zwinięcie wybranej gałęzi. Jeżeli przy (obok) ikonach obiektów występują czerwone strzałki, to oznaczają one, że obiekty te mogą wywoływać jeden i więcej podprogramów. Niebieskie strzałki przy obiektach wskazują, że obiekt wywoływany jest z kilku lokalizacji, ale nie wszystkie wywołania są w bieżącej chwili prezentowane w oknie.
8. Skontrolować sposób działania przycisków paska narzędzi. Zwrócić uwagę na zmiany dokonywane za pomocą dwóch przycisków układu Layout (Vertical Layout oraz Horizontal Layout). Sprawdzić możliwości własnej edycji obrazu widoku hierarchicznego, w tym zmiany położenia ikon (przeciąganie ikon kursorem myszki) oraz prezentację elementów za pomocą trzech przycisków zawartości Include (Include VI Lib, Include Globals oraz Include Typ Def.).
9. Wykorzystać przycisk Redo Layout do „przerysowania” wyświetlanego schematu, w celu ograniczenia obszaru rysunku i minimalizacji przecięć połączeń w widoku hierarchicznym programu.



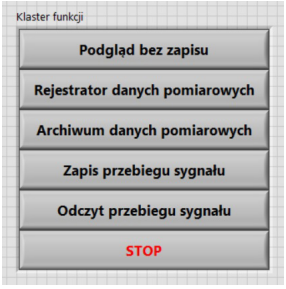
10. Aby zapoznać się z danym podprogramem i wyświetlić jego panel czołowy, kliknąć dwukrotnie w ikonę dowolnego podprogramu.
11. Zamknąć otworzone okno podprogramu bez zapisu zmian, a następnie zamknąć okno VI Hierarchy programu 094_Aplikacja.vi.

9.3.6. Generowanie pliku wykonywalnego

Cel zadania: przygotowanie pliku wykonywalnego (.exe) oraz plików aplikacji przeznaczonej dla użytkownika końcowego. Efektem procesu kompilowania programu będzie stworzenie niezależnej od platformy LabVIEW aplikacji funkcjonującej w środowisku Windows.

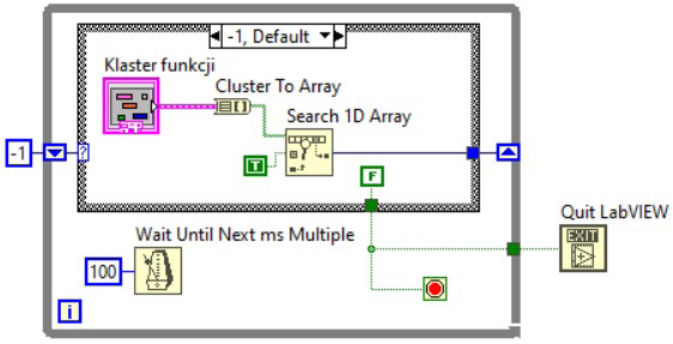
Zakres zadania: wykorzystanie funkcji służącej do budowy zarządzania projektem. Użycie narzędzia Build Application (EXE) from VI.

1. **Czynności wstępne.** Otworzyć plik programu 094_Aplikacja.vi. Panel czołowy aplikacji powinien być analogiczny do przedstawionego na rysunku.



2. Za pomocą narzędzia VI Properties skonfigurować program do pracy w trybie aplikacji pierwszoplanowej (menu kontekstowe programu, następnie polecenie File/VI Properties…, w oknie Category wybrać Window Appearance, a potem opcję Top-level application window). Takie ustawienie zapewni profesjonalny wygląd programowi uruchamianemu z pliku wykonywalnego.
3. Zapisać plik programu pod nową nazwą 096_System.vi i przejść do edycji schematu blokowego.

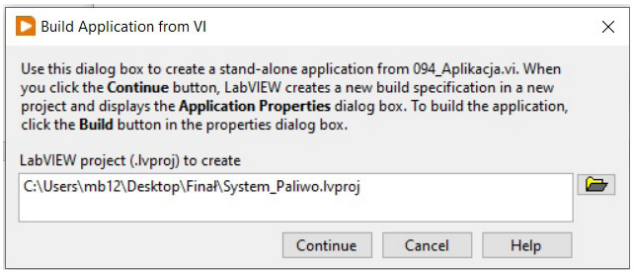
4. **Schemat blokowy.** Schemat blokowy uzupełnić o funkcję Quit LabVIEW (paleta Functions /Programming /Application Control). Wywołanie tej funkcji umożliwi automatyczne zamknięcie aplikacji po naciśnięciu przycisku [STOP].
5. Uzupełnić schemat blokowy o brakujące połączenie tak, żeby uzyskać efekt zgodny z przedstawionym na rysunku. Wprowadzone zmiany umożliwiają po zakończeniu działania aplikacji zamknięcie aktywnych okien programu i opuszczenie środowiska LabView.
6. Zapisać aktualną postać programu w pliku pod bieżącą nazwą (096_System.vi).



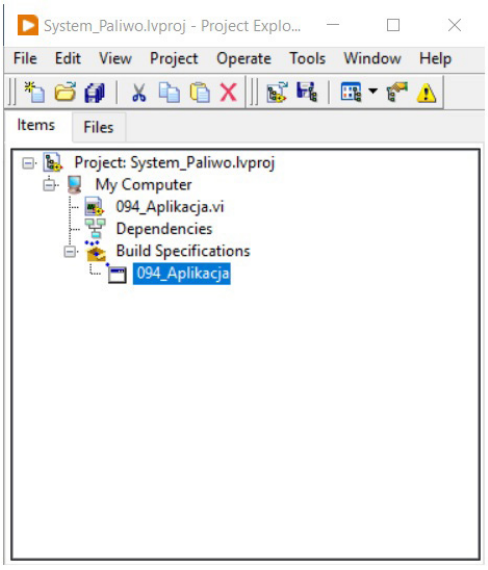
7. **Kontrola funkcjonowania programu.** Przejść do panelu czołowego programu i go uruchomić. Następnie wybrać przycisk [STOP]. Program zatrzyma się, a dodatkowo zostanie wyłączone całe środowisko LabVIEW.
8. Uruchomić ponownie program LabVIEW i otworzyć program 096_System.vi.

9. **Tworzenie pliku wykonywalnego.** Przed przystąpieniem do tworzenia aplikacji EXE z programu VI sprawdzić poprawność funkcjonowania programu, używając wszystkich dostępnych opcji. Należy zwrócić szczególną uwagę na ustawienia opcji dotyczące wywoływania wstawionych podprogramów. Ważne jest, aby podprogramy miały aktywne opcje Show front panel when called oraz Close afterwards if oryginaly closed (opcje te można ustawić w oknie dialogowym SubVI Nod Setup danego podprogramu).

10. Z menu Tool wybrać polecenie Build Application (EXE) from VI.... Funkcja ta umożliwia przejście do edytora aplikacji z pliku VI. Uruchomienie edytora tworzenia aplikacji rozpoczyna się od okna komunikacyjnego (patrz rysunek), zawierającego informację o lokalizacji pliku projektu (rozszerzenie pliku: .lvproj). Wprowadzić nazwę pliku projektu jako System_Paliwo.lvproj z lokalizacją docelową we własnym katalogu roboczym. Zatwierdzić zamiar tworzenia pliku projektu przyciskiem Continue.



11. Po chwili zostanie wyświetlone okno Project Explorer (patrz rysunek) zawierające informacje o tworzonym pliku projektu System_Paliwo.lvproj.

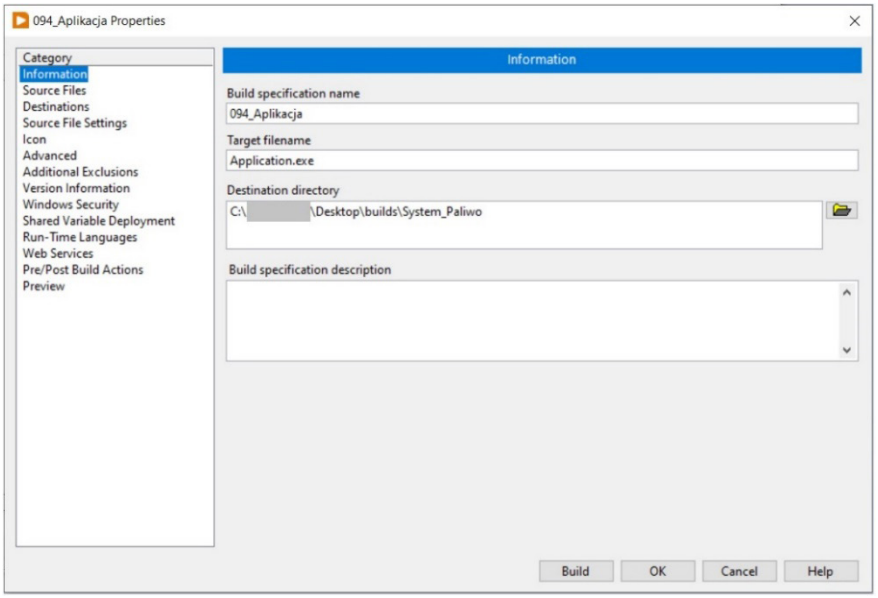


12. Następnie zostanie uruchomiony konfigurator aplikacji umożliwiający ustawienie żądanych właściwości dla tworzonej aplikacji. W oknie dialogowym Properties wymieniono szereg nazw zakładek konfiguracyjnych (Category), wywołujących

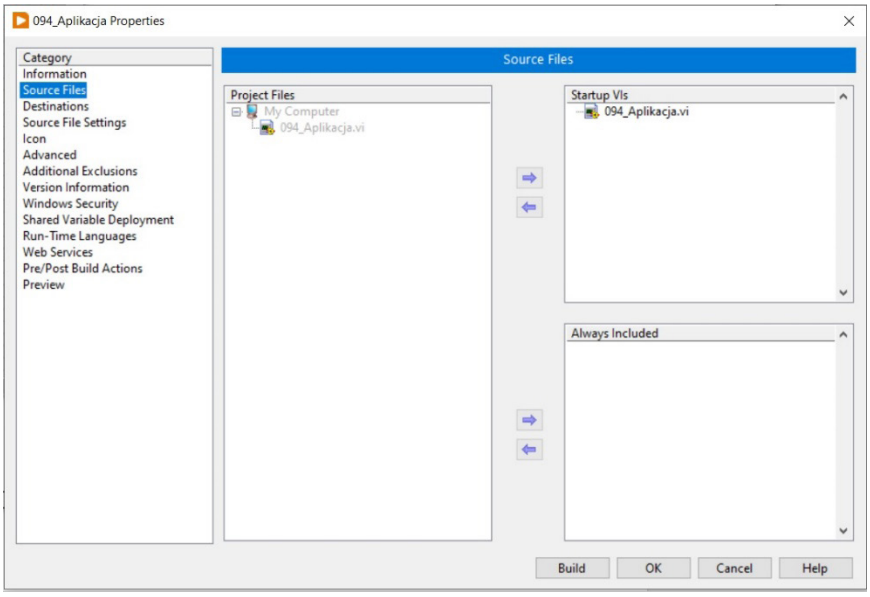
odpowiednie zakładki z polami ustawień dla tworzonej aplikacji, związanymi tematycznie z daną kategorią ustawień. Najważniejsze kategorie ustawień to:

- Information,
- Source Files,
- Destination,
- Icon,
- Version Information.

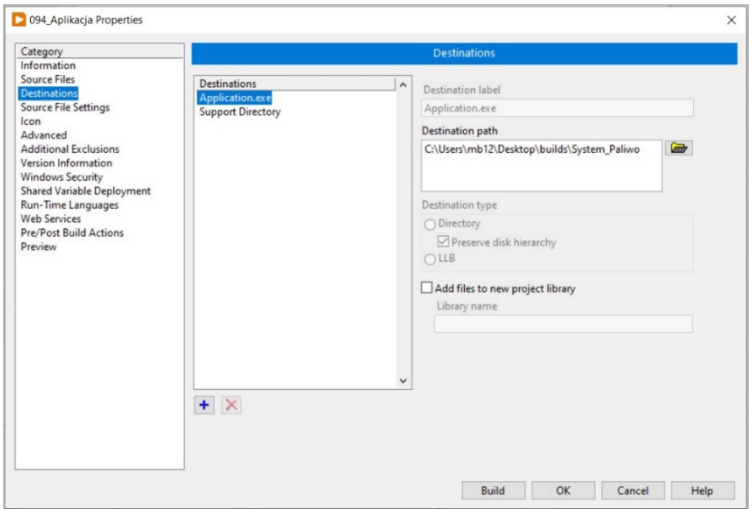
13. Okno konfiguracyjne Information ma postać przedstawioną na rysunku i umożliwia określenie nazwy dla tworzonej aplikacji oraz nazwy i lokalizacji dla jej pliku docelowego. Możliwe jest również wstawienie opisu dla budowanej aplikacji. Należy uzupełnić okna dialogowe zgodnie z wzorcem z rysunku.



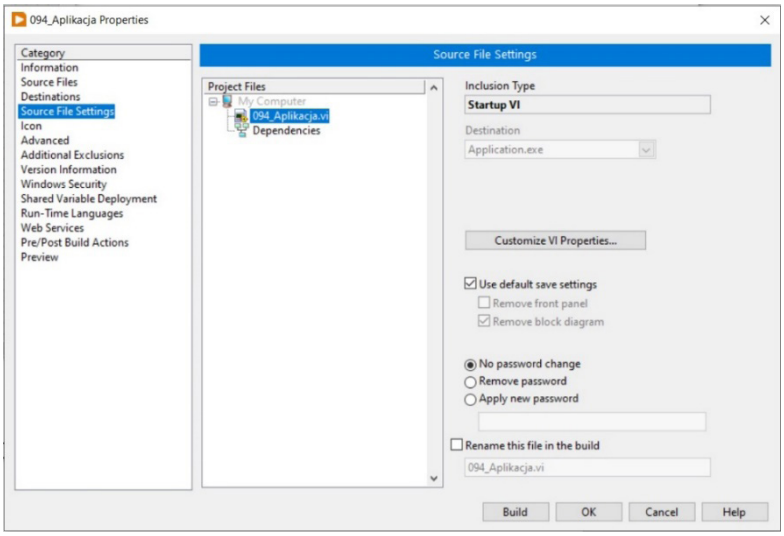
14. Okno konfiguracyjne Source Files ma taką postać jak na rysunku i umożliwia określenie plików źródłowych, jakie zawierać ma tworzona aplikacja. Należy zapoznać się z wygenerowanymi ustawieniami.



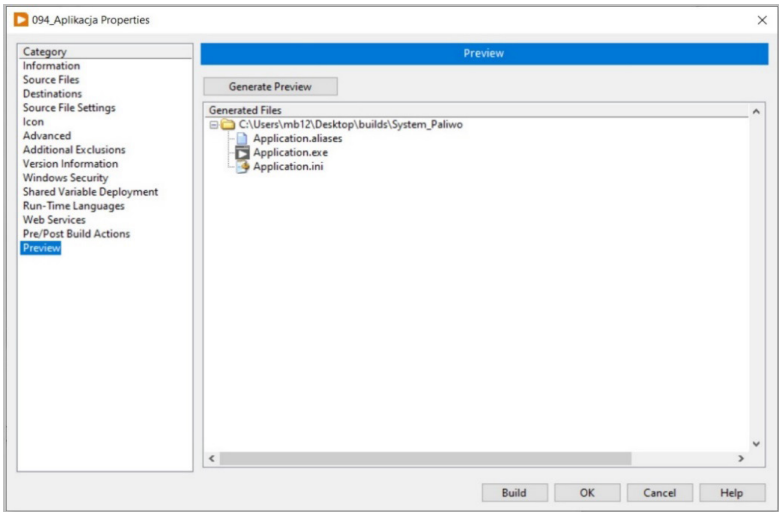
15. Okno konfiguracyjne Destinations ma postać przedstawioną na rysunku i umożliwia określenie przeznaczenia projektu (aplikacja exe lub katalog wsparcia) oraz określenie pliku docelowego dla tworzonej aplikacji. Należy zapoznać się z wygenerowanymi ustawieniami.



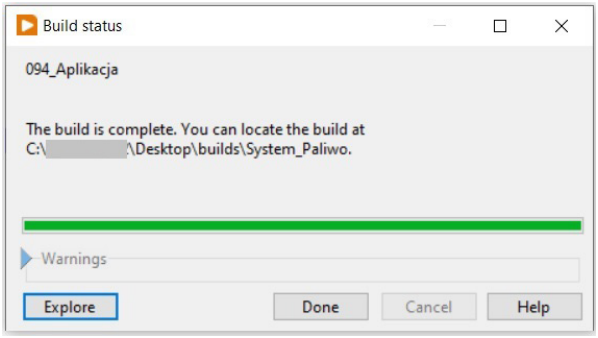
16. Okno konfiguracyjne Source File Settings ma postać analogiczną do tej na rysunku i umożliwia m.in. ustawienia zabezpieczeń programu. Należy zapoznać się z wygenerowanymi ustawieniami.



17. Okno konfiguracyjne Icon umożliwia określenie typu oraz edycję znaku graficznego (ikony) dla tworzonej aplikacji (pliku .exe).
18. Okno konfiguracyjne Version Information pozwala na określenie informacji dotyczących projektów rozwijanych, w tym numeru wersji, opisu przeznaczenia programu i edycji metryczki dla aplikacji.
19. Okno konfiguracyjne Preview (patrz rysunek) umożliwia zapoznanie się z zawartością katalogu z wygenerowanym programem użytkowym. Należy zapoznać się z wygenerowanymi informacjami.



20. Należy przejrzeć pozostałe okna konfiguracyjne i zwrócić uwagę na zakres możliwych zmian i ustawień właściwości dla tworzonej aplikacji. Zachować automatycznie wygenerowane ustawienia w oknach konfiguracyjnych.
21. Użyć przycisku Build w celu zainicjowania procesu tworzenia plików autonomicznie działającej aplikacji. Proces tworzenia aplikacji zakończyć przyciskiem Done w wyświetlonym oknie dialogowym Build status (patrz rysunek).



22. Zamknąć okno Project Explorer aplikacji System_Paliwo.lvproj (z ewentualnym potwierdzeniem zapisu zmian w pliku projektu), a następnie zamknąć wszystkie aktywne okna programów w środowisku LabVIEW.
23. Odszukać we właściwym katalogu z programami (zgodnie z zadeklarowaną lokalizacją) plik Aplikacja.exe i uruchomić go. Skontrolować funkcjonowanie aplikacji, wywołując jej wszystkie procedury programowe. Zakończyć pracę aplikacji za pomocą przycisku [STOP].
24. Zamknąć wszystkie aktywne okna programów środowiska LabVIEW.

9.4. Pytania kontrolne

1. Jakie są cechy wspólne i różnice pomiędzy komunikacją prowadzoną w standardach RS-232 a standardach RS-485?
2. Jakie są najpopularniejsze mechanizmy (standardy, sposoby, protokoły) wymiany danych pomiędzy aplikacjami i systemami, wspierane przez środowisko LabVIEW?
3. Jakie efekty uzyskuje się za pomocą narzędzia Web Publishing Tool?
4. Jakie dodatkowe czynności wymagane są do kontrolowania aplikacji za pomocą protokołu http?
5. W jaki sposób (i przy pomocy jakich narzędzi) wspiera się rozwiązywanie problemów z komunikacją pomiędzy aplikacjami bazującymi na wielu podprogramach?

10. Wyrównywanie zaległości i ocena postępów

10.1. Cel zajęć

Rozdział odnosi się do przeprowadzania ćwiczeń w trybie spotkań laboratoryjnych. Podczas takich podsumowujących zajęć zostają wykonane działania wyrównawcze, umożliwiające realizację pełnego procesu kształcenia i wszystkich zadań przewidzianych do samodzielnego wykonania.

Celem zajęć w szczególności jest:

- podsumowanie stanu realizacji wszystkich celów przedmiotu i efektów uczenia się,
- wykonanie dodatkowych projektów zadaniowych dla studentów chcących podnieść swoje umiejętności,
- przeprowadzenie testu wiedzy z zakresu programowania w środowisku LabVIEW,
- ocena zdobytych umiejętności.

10.2. Warunki uzyskania zaliczenia

Warunkiem zaliczenia jest opracowanie wszystkich wymaganych przez prowadzącego aplikacji w postaci „zbliżonej" do wzorcowej, przy częstym korzystaniu z pomocy wykładowcy. Dopuszczalna jest dysfunkcjonalność pojedynczych programów. Natomiast warunkiem otrzymania maksymalnej oceny jest samodzielne opracowanie obligatoryjnych aplikacji:

- z pełną funkcjonalnością programów,
- przy braku pomocy ze strony wykładowcy,
- w czasie trwania zajęć,
- na wysokim poziomie graficznym.

W celu przedstawienia obiektywnej oceny wykładowca powinien prowadzić bieżące notatki, pozwalające na określenie postępów osiąganych w trakcie każdych zajęć laboratoryjnych.

11. Bibliografia

[1] Bitter R., Mohiuddin T., Nawrocki M., *LabVIEW: advanced, programming, technique,* Wydawnictwo CRC Press Taylor & Francis Group, Boca Raton––London–New York 2007.

[2] Chruściel M., *LabVIEW w praktyce*, Wydawnictwo BTC, Legionowo 2008.

[3] Daciuk J., *Projektowanie interfejsu człowiek-komputer*, [dostęp: 17.12.2022]. Dostępny w internecie: https://eti.pg.edu.pl/documents/176468/27264685/ocena.pdf.

[4] Gołębiowski J., *Laboratorium komputerowych systemów pomiarowych*, Wydawnictwo Politechniki Łódzkiej, Łódź 2004.

[5] Gook M., *Interfejsy sprzętowe komputerów PC*, Wydawnictwo Helion, Gliwice 2005.

[6] Juszczuk P., *GUI – projektowanie interfejsów*, [dostęp: 11.01.2023]. Dostępny w internecie: http://www.pjuszczuk.pl/wp-content/uploads/2016/12/Prezentacja.pdf.

[7] Kościelny J. M., *Diagnostyka zautomatyzowanych procesów przemysłowych*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001.

[8] Kruczkiewicz Z., *Projektowanie interfejsów graficznych aplikacji (GUI)*, [dostęp: 22.03.2023]. Dostępny w internecie: http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/ wyklady/analizasi/interfejsuzytkownika.pdf.

[9] National Instruments, *Configuring Software and Hardware Firewalls to Support NI Products*, [dostęp: 22.02.2023]. Dostępny w internecie: https://www.ni.com/pl-pl/support/documentation/supplemental/10/configuring-software-and-hardware-firewalls-to-support-national-.html.

[10] National Instruments, *Data Acquisition (DAQ)*, [dostęp: 15.05.2023]. Dostępny w internecie: https://www.ni.com/pl-pl/shop/data-acquisition.html.

[11] National Instruments, *DataSocket Transfer Protocol (dstp) Overview*, [dostęp: 24.04.2023]. Dostępny w internecie: https://www.ni.com/pl-pl/innovations/white-papers/06/datasocket-transfer-protocol--dstp--overview.html.

[12] National Instruments, *Determining When to Use Sequence Structures*, *System pomocy (Help)* środowiska *LabVIEW*, National Instruments, 2022.

[13] National Instruments, *Development Guidelines*, VII, National Instruments, 2000.

[14] National Instruments, *G Programming Reference Manual, BridgeVIEW and LabVIEW*, National Instruments, 2008.

[15] National Instruments, *Grouping Data with Arrays and Clusters*, *System pomocy (Help)* środowiska *LabVIEW*, National Instruments, 2022.

[16] National Instruments, *LabVIEW Express Basics Interactive Training*, National Instruments, 2008.

[17] Nawrocki W., *Komputerowe systemy pomiarowe*, WKiŁ, Warszawa 2002.

[18] Niziański S., Michalski R., *Diagnostyka obiektów technicznych,* Wydawnictwo i Zakład Poligrafii Instytutu Technologii Eksploatacji, Radom 2002.

[19] *Practical Applications and Solutions Using LabVIEW Software,* ed. by Folea S., InTech, 2011.
[20] Świsulski D., *Systemy pomiarowe – Laboratorium*, Wydawnictwo Politechniki Gdańskiej, Gdańsk 2001.
[21] Tłaczała W., Środowisko *LabVIEW w eksperymencie wspomaganym komputerowo*, Wydawnictwo WNT, Warszawa 2002.