|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| logoKM | logoKM | | POLITECHNIKA ŚLĄSKA  WYDZIAŁ ELEKTRYCZNY  KATEDRA MECHATRONIKI | logoKM |
| Instrukcja do ćwiczenia laboratoryjnego | | | | |
| Przedmiot: | | Przetwarzanie i Wizualizacja Danych Pomiarowych | |  |
| Symbol ćwiczenia: | | PiWDP2 | |  |
| Tytuł ćwiczenia: | | **Podstawy obsługi środowiska, tworzenia i debuggowania programów i podprogramów** | | |

**SPIS TREŚCI**

|  |  |  |
| --- | --- | --- |
|  | **SPIS RYSUNKÓW** | 2 |
| 1. | **CELE ĆWICZENIA** | 3 |
| 2. | **WPROWADZENIE** | 3 |
| 2.1. | **Podstawowe wiadomości o środowisku LabVIEW** | 3 |
| 2.2. | **Program narzędziowy MAX** | 6 |
| 2.3. | **Konfiguracja zadania pomiarowego za pomocą kreatora** | 9 |
| 3. | **LABORATORYJNE STANOWISKO BADAWCZE** | 10 |
| 3.1. | **Obiekt badany** | 10 |
| 3.2. | **Urządzenia dodatkowe** | 10 |
| 3.3. | **Oprogramowanie** | 10 |
| 4. | **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI** | 10 |
| 5. | **PRZYKŁAD REALIZACJI ZADANIA – Przygotowanie symulowanej karty DAQ** | 11 |
| 6. | **RAPORT** | 12 |
| 7. | **PYTANIA** | 12 |

# SPIS RYSUNKÓW

|  |  |  |
| --- | --- | --- |
| 1. |  |  |
| 2. |  |  |
| 3. |  |  |
| 4. |  |  |
| 5. |  |  |
| 6. |  |  |
| 7. |  |  |
|  |  |  |
|  |  |  |

* wyszukiwanie węzłów i obiektów panelu (funkcje Quick Drop **Ctrl**+**Space**)
* funkcje Panelu Czołowego i Diagramu Kodu
* pomoc w LabVIEW
* skróty klawiaturowe (**Ctrl**+**E**, **Ctrl**+**B**)
* pomoc kontekstowa
* okno help
* explain error
* podstawy „dataflow” – kod pojedynczy i równoległy (Highlight execution)
* testowanie działania aplikacji (podgląd, praca krokowa, breakpoint)
* tworzenie i opisywanie kodu
* znaczące nazwy plików VI
* ikony
* nazwy terminali we/wy
* domyśla wartość kontrolki
* nazwy przewodów (labels)
* wolne etykiety (free labels)
* zakładki **#nazwa** (tylko w Block Diagram), wyszukiwanie zakładek   
  **> View -> Bookmark Manager**
* tworzenie subVI
* z pustego VI
* **> Edit -> Create SubVI**
* z szablonu

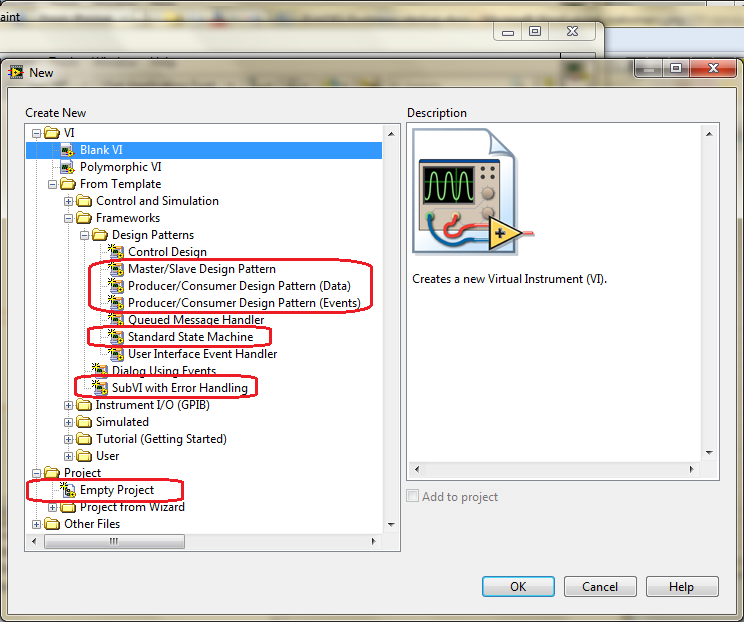
1. **CELE ĆWICZENIA**

Celem ćwiczenia jest zapoznanie się z podstawową obsługą środowiska LabVIEW:

* Zapoznanie się z metodami wyszukiwania węzłów diagramu oraz komponentów panelu,
* Zapoznanie się z funkcjami Panelu Czołowego oraz Diagramu kodu,
* Zapoznanie się z mechanizmami wspierającymi pracę programisty,
* Zapoznanie się z ideą wykonywania kodu graficznego – „dataflow”,
* Zapoznanie się z metodami debuggowania aplikacji i subVI,
* Zapoznanie się z podstawowymi technikami tworzenia i opisywania kodu.

1. **WPROWADZENIE**
   1. **Rozpoczynanie pracy w środowisku LabVIEW**

Pracę w środowisku LabVIEW można zacząć na kilka sposobów. Można utworzyć nowy pusty plik (File 🡪 New VI), jednak można w znaczy sposób przyspieszyć pracę korzystając z szablonów (File 🡪 New…). Szablony startowe w LabVIEW zawierają gotowe, najczęściej używane struktury programistyczne. Do najważniejszych z nich można zaliczyć: szablon projektu, strukturę podprogramu z funkcją obsługi błędów (SubVI with terror handling), struktury wielopętlowe oraz strukturę maszyny stanów.

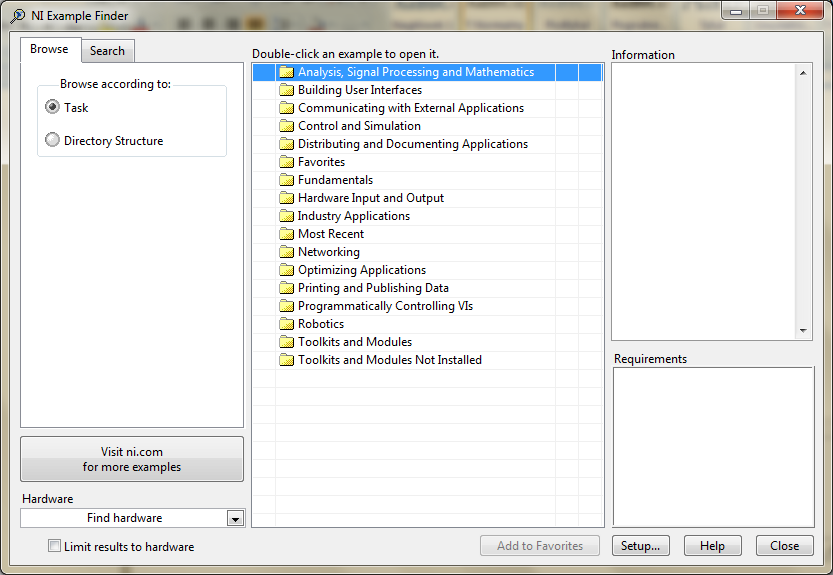


1. Widok okna wyboru szablonu z zaznaczonymi najważniejszymi strukturami.

Zaznaczone na Rys. 1 szablony będą omawiane w trakcie zajęć. Szablony są to gotowe fragmenty plików zawierające określoną strukturę programistyczną oraz najważniejszą funkcjonalność. Niektóre szablony mogą zostać uruchomione bez żadnych modyfikacji. Wszystkie szablony wyposażone są w dokładne opisy działania i instrukcje dla programisty.

Inną metodą przyspieszającą pracę w LabVIEW są przykłady. Wyszukać ich można za pomocą wyszukiwarki przykładów uruchamianej za pomocą:

Help 🡪 Find Examples…

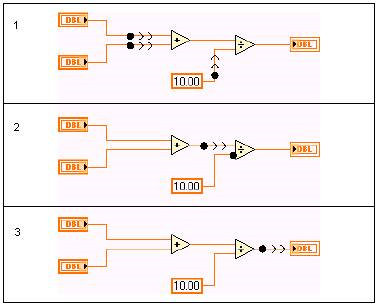


1. Okno wyszukiwarki przykładów.

W przeciwieństwie do szablonów wszystkie przykłady są gotowymi, uruchamialnymi aplikacjami LabVIEW. Zarówno wszystkie szablony jak i przykłady mogą być użyte bez dodatkowych opłat w pracy programisty w ramach licencji (***licencja akademicka nie umożliwia komercyjnego wykorzystania środowiska LabVIEW***).

* 1. **Dataflow - idea programowania graficznego**

Bardzo istotną sprawą jest zrozumienie mechanizmu kolejności wykonywania kodu w LabVIEW. W tekstowych językach strukturalnych kolejność wykonywania kodu jest determinowana kolejnymi liniami od góry do dołu. W LabVIEW kolejność wykonywana kodu jest określana przepływem danych **DATAFLOW**. Elementami wprowadzającymi dane są terminale wejściowe – są to kontrolki, stałe lub węzły nie posiadające tuneli wejściowych (np. węzeł Random), dalej dane przechodzą przez kolejne węzły zgodnie ze strukturą połączeń, ostatnimi elementami są wskaźniki prezentujące wynik działania kodu. Przykład działania dataflow z rozbiciem na kolejne kroki wykonywanego kodu został przedstawiony na Rys. 3.



1. Dataflow – kolejność wykonywania kodu w LabVIEW.

Kolejność wykonywania programu w LabVIEW jest określona następującymi zasadami:

* Przepływ danych odbywa się od lewej do prawej strony (kontrolki umieszczamy na lewo a indykatory na prawo kodu),
* Przepływ danych odbywa się od kontrolek do indykatorów,
* Węzeł wykona swoje zadanie jeżeli otrzyma WSZYSTKIE wymagane dane wejściowe,
* Wszystkie węzły, które otrzymują dane w tym samym kroku wykonują się „równolegle”,
* Niezależne kody programu wykonują się „równolegle”,
  1. **Tworzenie podprogramów – SubVI**

Bardzo istotna w LabVIEW jest modułowość pisanego kodu. Jest ona nierozłączna z dobrymi praktykami programowania w LabVIEW. Do dobrych praktyk zaliczyć można: czytelność i skalowalność. Modułowość kodu bazuje na zamykaniu fragmentów kodu w podprogramach celem późniejszego ich zastosowania w różnych miejscach aplikacji, w LabVIEW podprogram nazywa się **subVI**. Każdy plik vi może być użyty w innym pliku jako subVI. Pamiętać należy, że przekazywanie danych do i z subVI odbywa się za pomocą panelu połączeń (patrz instrukcja do ćwiczenia 1). Tworzenie podprogramów można wykonać na trzy sposoby:

* Tworzyć nowy czysty plik vi i zaimplementować w nim strukturę subVI,
* Utworzyć nowy subVI z szablonu: SubVI with error handling (),
* Gotowy fragment kodu zaznaczyć i zamienić na subVI: Edit 🡪 Create SubVI,

Podstawowa oraz inne dostępne struktury panelu połączeń zostały przedstawione na Rys. 4. Pamiętać należy, że wszystkie komponenty pulpitu, które mają przenosić dane do nadrzędnego vi musza być podpięte do panelu terminali.

|  |  |
| --- | --- |
| a) | b) |
| 1. Struktury panelu połączeń (a) domyślna, (b) inne dostępne. | |

Wewnątrz struktury połączeń wybrane terminale zarezerwowane są dla sygnałów błędu oraz referencji.



1. Domyślne terminale dla klastra błędu oraz referencji.

Terminale wejściowe posiadają różne priorytety podłączenia. Mogą przyjmować następujące:

* Recommended – domyślny priorytet, oznaczony cienką czarną linią,
* Required – wejście wymagane, jeżeli do tego terminalu nie jest podpięty przewód kompilator zgłasza błąd kompilacji,
* Optional – wejście opcjonalne – oznaczone kolorem szarym.

Typ wejścia zmienia się klikając:

PPM 🡪 This Connection Is 🡪 Required/Recommended/Optional

Ponadto dla wszystkich wejść można ustalić wartości domyślne dla danej kontrolki:

(na kontrolce wpisać wartość, która ma być domyślna a następnie)

PPM 🡪 Data Operations 🡪 Make Current Value Default

lub domyślnie dla wszystkich kontrolek na panelu:

Edit 🡪 Make Current Values Default

* 1. **Dokumentacja kodu**

Dsd

* 1. **Debuggowanie i wyszukiwanie błędów w programie**

1. **LABORATORYJNE STANOWISKO BADAWCZE**
   1. **Obiekt badany**

- Środowisko programistyczne LabVIEW,

* 1. **Urządzenia dodatkowe**

- brak,

* 1. **Oprogramowanie**
* LabVIEW 2013 lub nowszy

1. **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI**

Kolejne kroki do wykonania podczas zajęć:

- uruchomić nowy plik vi (File 🡪 New VI lub Ctrl+N)

* Zapoznanie się z metodami wyszukiwania węzłów diagramu oraz komponentów panelu:

- zidentyfikować okno panelu oraz okno diagramu,

- przełączyć kilka razy między oknami za pomocą odpowiedniego skrótu klawiaturowego (patrz poprzednia instrukcja) (………………… można wpisać skrót)

- za pomocą palety funkcji (PPM) na panelu umieścić dowolną kontrolkę typu string

- skopiować kontrolkę używając Ctrl+LPM

- zmienić typ skopiowanego terminalu z kontrolki na terminal

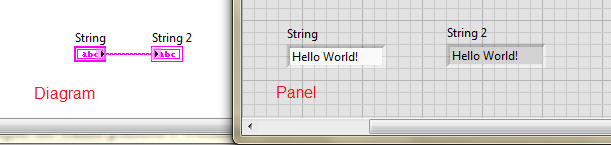
PPM 🡪 Change to Indicator

- sprawdzić reprezentacje terminali w oknie diagramu

- połączyć za pomocą przewodu kontrolkę ze wskaźnikiem

- wrócić do Panelu i w kontrolce wpisać wybrany tekst np. „**Hello World!**”

- uruchomić aplikację (Ctrl+R)



- przejść do okna panelu i uruchomić **Quick-Drop** (Ctrl+Space)

- wyszukać kontrolkę typu Numeric i umieścić na panelu

- zmienić nazwę kontrolki na „a”

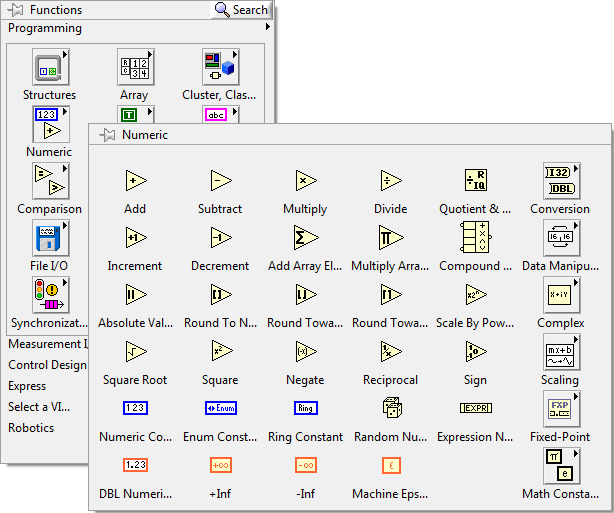
- wybraną metodą (kopiowanie, Quick-Drop, paleta kontrolek) umieścić dodatkowe 3 kontrolki typu Numeric

- pozmieniać nazwy: „a”, „b”, „x”, „y”,

- kontrolkę „y” zamienić a wskaźnik,

- przejść do okna diagramu

- w palecie funkcji znaleźć funkcje matematyczne



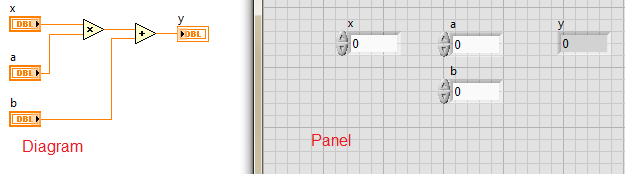
- sprawdzić i zapamiętać nazwę węzła mnożenia

- wybrać i umieścić na diagramie węzeł dodawania

- za pomocą Quick drop dodać węzeł mnożenia

- połączyć elementy diagramu aby uzyskać funkcję liniową „y=a\*x+b”

- uporządkować obiekty w obydwu oknach (np. jak na rysunku)



- zadać wartości w kontrolkach i uruchomić aplikację

- zapisać plik pod nazwą „f\_liniowa.vi”

- zamknąć plik,

* Zapoznanie się z mechanizmami wspierającymi pracę programisty,
* Zapoznanie się z ideą wykonywania kodu graficznego – „dataflow”,
* Zapoznanie się z metodami debuggowania aplikacji i subVI,
* Zapoznanie się z podstawowymi technikami tworzenia i opisywania kodu.

- podpiąć kartę DAQ i zaczekać na jej zgłoszenie w systemie,

- uruchomić

1. **PRZYKŁAD REALIZACJI ZADANIA – Przygotowanie symulowanej karty DAQ**

W tej części zadania należy zasymulować kartę NI USB-6210.

1. **RAPORT**

Raport z przeprowadzonego ćwiczenia laboratoryjnego powinien zawierać opis kolejnych czynności wykonywanych w trakcie realizacji ćwiczenia, zrzuty ekranu dokumentujące wykonane kroki oraz zanotowane parametry konfiguracyjne kart DAQ (mogą być zawarte w tabeli)

1. **PYTANIA**
2. Opisać ideę programowania “dataflow”.

# LITERATURA

1. LabVIEW Core 1 Course manual.
2. LabVIEW Core 1 Exercise book.
3. LabVIEW Core 2 Course manual.
4. LabVIEW Core 2 Exercise book.
5. Nota katalogowa LM35

Opracowanie: Marek Kciuk

# ZADANIA DO REALIZACJI PRZED PRZYSTĄPIENIEM DO ĆWICZENIA

- przygotować tabelę przeliczników między skalami Kewina Celcjusza i Farenchaita

|  |  |  |  |
| --- | --- | --- | --- |
| Współczynniki (a,b) | Kewin, K | Celcjusz oC | Ferenchait, F |
| Kewin, K | 1, 0 |  |  |
| Celcjusz oC |  | 1, 0 |  |
| Ferenchait, F |  |  | 1, 0 |

# ZADANIA DODATKOWE PO WYKONANIU ĆWICZENIA

- zapoznać się z właściwościami (PPM 🡪 Properties) kontrolek, zwłaszcza z zakładkami:

**Appearance**, **Data type**, **Data Entry**, **Display format**, **Documentation**,

w przypadku wskaźników typu graph dodatkowo:

**Plots**, **Scales**, **Cursos**.

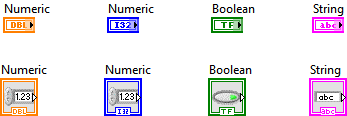
- Jakie możliwości edycji mają właściwości?

Do niektórych właściwości jest szybszy dostęp bezpośrednio po naciśnięciu PPM (np. PPM 🡪 Representation)

# ZAŁĄCZNIKI

## Wygląd wejściowych i wyjściowych komponentów pulpitu na diagramie

Komponenty pulpitu mają swoją reprezentację w oknie diagramu, niezależnie od typu oraz wyglądu na panelu w oknie diagramu może przyjąć jeden z dwóch widoków: widok ikony lub widok terminalu. Na rysunku przedstawiono obydwa widoki:



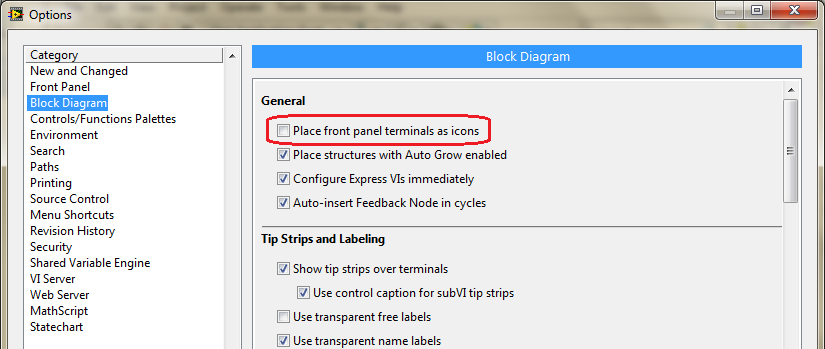
Zmianę widoku pojedynczej ikony można dokonać:

PPM 🡪 View as Icon (zaznaczyć lub odznaczyć).

Jeżeli chcemy zmienić domyślny wygląd terminalu w opcjach LabVIEW należy wybrać:

Tools 🡪 Options… 🡪 Block Diagram

(zaznaczyć lub odznaczyć opcję „**Place front panel terminals as icons**”)



## Zapisywanie plików wykonywalnych w postaci plików graficznych (snippet)

W LabWIEV istnieje możliwość tworzenia plików wykonywalnych ukrytych pod postacią pliku graficznego o rozszerzeniu PNG. Plik taki w swojej strukturze zawiera zrzut ekranu „printscreen” fragmentu kodu z diagramu oraz kod wykonywalny w postaci subVI. Próba podwójnego kliknięcia spowoduje uruchomienie domyślnego programu graficznego i wyświetlenie zawartości części graficznej pliku. Umieszczenie pliku na stronie internetowej także skutkować będzie wyświetleniem części graficznej (patrz link niżej). Uruchomienie tego typu pliku polega na przeciągnięciu go do okna diagramu metodą Drag & Drop.

Przygotowanie graficznego pliku wykonywalnego:

- zaznaczyć interesujący fragment kodu,

- Edit 🡪 Create snippet from selection 🡪 zapisać plik z rozszerzeniem png.

Zaletą tego typu rozwiązania w porównaniu do klasycznych printscreenów jest fakt, że w przypadku pętli warunkowych cały kod (z wszystkimi warunkami) zostanie zapisany w części uruchamialnej. Klasyczny printscreen wymaga wykonania zrzutu dla każdego warunku Case osobno.

Dodatkowe informacje oraz przykładowe snippety:

http://www.ni.com/tutorial/9330/en/