|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| logoKM | logoKM | | POLITECHNIKA ŚLĄSKA  WYDZIAŁ ELEKTRYCZNY  KATEDRA MECHATRONIKI | logoKM |
| Instrukcja do ćwiczenia laboratoryjnego | | | | |
| Przedmiot: | | Przetwarzanie i Wizualizacja Danych Pomiarowych | |  |
| Symbol ćwiczenia: | | PiWDP4 | |  |
| Tytuł ćwiczenia: | | **Pętle w środowisku LabVIEW** | | |

**SPIS TREŚCI**

|  |  |  |
| --- | --- | --- |
|  | **SPIS RYSUNKÓW** | 2 |
| 1. | **CELE ĆWICZENIA** | 3 |
| 2. | **WPROWADZENIE** | 3 |
| 2.1. | **Struktura wyboru - Case** | 3 |
| 2.2. | **Praca w projekcie** | 5 |
| 3. | **LABORATORYJNE STANOWISKO BADAWCZE** | 6 |
| 3.1. | **Obiekt badany** | 6 |
| 3.2. | **Urządzenia dodatkowe** | 6 |
| 3.3. | **Oprogramowanie** | 7 |
| 4. | **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI** | 7 |
| 5. | **PRZYKŁAD REALIZACJI ZADANIA – Przygotowanie symulowanej karty DAQ** | 8 |
| 6. | **RAPORT** | 9 |
| 7. | **PYTANIA** | 9 |
|  |  |  |

# SPIS RYSUNKÓW

|  |  |  |
| --- | --- | --- |
| 1. |  |  |
| 2. |  |  |
| 3. |  |  |
| 4. |  |  |
| 5. |  |  |
| 6. |  |  |
| 7. |  |  |
| 8. |  |  |
| 9. |  |  |

1. **CELE ĆWICZENIA**

Celem ćwiczenia jest zapoznanie się z działaniem i zastosowaniem pętli programowych w środowisku LabVIEW:

* Poznanie właściwości i zastosowania pętli While
* Poznanie właściwości i zastosowania pętli For
* Poznanie struktur sekwencyjnych

1. **WPROWADZENIE**

Pętle w środowisku LabVIEW są zamkniętymi obszarami otaczającymi fragment kodu, który ma być wykonywany wielokrotnie. **Indeksowanie pętli (tak samo jak indeksowanie wektorów) zaczyna się od wartości 0!** Wejścia i wyjścia pętli nazywamy tunelami odpowiednio wejściowymi i wyjściowymi. Istnieją cztery typy tuneli:

- tunel zwykły (**Last Value**) – (wejściowy) wprowadza dane do pętli, we wszystkich iteracjach wprowadza tą samą wartość, (wyjściowy) wyprowadza do dalszego fragmentu kodu wartość z ostatniej iteracji,

- tunel autoindeksowany (**Indexing**) – służy do współpracy pętli z wektorami danych, (wejściowy) w kolejnych iteracjach pętli wprowadza dane z kolejnych komórek wektora – w 0 iteracji wprowadza wartość z 0 elementu wektora, w 1 iteracji wprowadza wartość z 1 elementu wektora itd. (wyjściowy) tworzy wektor zawierający kolejne wartości obliczone w kolejnych iteracjach pętli, rozmiar wektora jest równy liczbie iteracji pętli,

- tunel łączący (**Concatenating**) – służy do uzupełniania wektora kolejnymi komórkami, istnieje tylko jako tunel wyjściowy,

- rejestr przesuwny (**Shift Register**) – tunel przenoszący wartość z poprzedniej iteracji do aktualnej, może być inicjowany wprowadzając daną do pierwszej iteracji, w przypadku, gdy nie jest inicjowany wprowadza domyślną lub zapamiętaną wartość, jako jedyny pojawia się po obydwu stronach pętli, może zostać rozszerzony, żeby „pamiętać” wartości z więcej niż jednej iteracji.

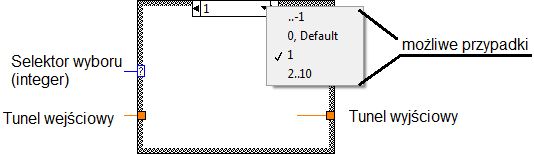
W przypadku pierwszych trzech tuneli (wyjściowych) można dodatkowo uaktywnić opcję warunkową (**Conditional**). W tym przypadku tylko wartości, dla których boolowski warunek jest spełniony zostają wyprowadzone.

* 1. **Pętla While**

Petla **While** jest główną pętlą programu, wykonuje się aż do spełnienia warunku wyjścia. Warunek wyjścia jest typu boolowskiego i może być ustawiony jako „Stop if True” lub jako „Continue if True”. Na Rys. 1 Przedstawiono wygląd pętli While.

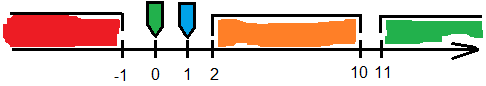
1. Pętla While.

Przypadek użycia struktury Case z selektorem typu integer został przedstawiony  
na Rys. 2. Dla liczb całkowitych można tworzyć przypadki dla konkretnych wartości (0, 1), zamkniętych przedziałów wartości [2..10], oraz otwartych przedziałów wartości (-∞..-1].



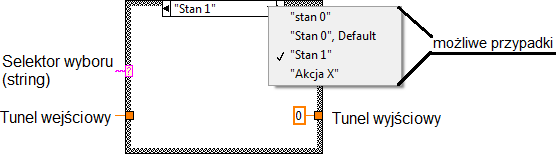
1. Struktura Case z selektorem typu integer.

Dla przypadków nieokreślonych wykonywany jest warunek oznaczony jako domyślny (**Default**). Przypadki takie na Rys. 2 to zakres liczb powyżej 10. Zakresy liczbowe  
dla poszczególnych przypadków zostały przedstawione na



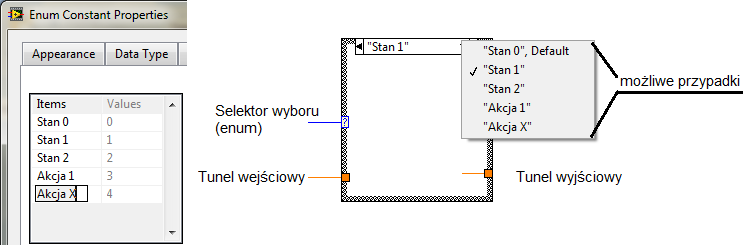
1. Przedziały zakresów liczbowych dla struktury Case z warunkami z Rys. 2.

Przypadek użycia selektora tekstowego (**string**) został przedstawiony na Rys. 4. LabVIEW rozróżnia wielkość liter więc przypadki „*stan 0*” oraz „*Stan 0*” to dwa różne stany. Należy zwrócić szczególną uwagę, jeżeli teksty podawane na selektor nie są zunifikowane (np. podawane przez użytkownika).



1. Struktura Case z selektorem typu string.

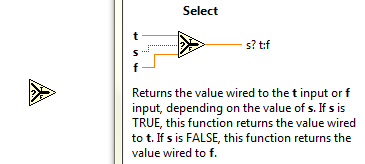
Przypadek użycia selektora **Enum**. Zmienna tego typu przypisuje kolejnym wartościom typu integer nazwy, jednak środowisko LabVIEW obsługuje go jako tryb numeryczny i umożliwia wykonywanie operacji arytmetycznych. Na Rys. 5 przedstawiono strukturę **Case** z selektorem typu enum, którego wartości także znajdują się na rysunku.



1. Struktura Case z selektorem typu enum.

Tunele wyjściowe muszą mieć wprowadzone dane we wszystkich przypadkach. Jednak istnieje opcja na tunelu wyjściowym PPM 🡪 **Use Default If Unwired**, która wypełnia wartościami domyślnymi przypadki nieobsłużone. W przypadku zmiennych liczbowych wartością domyślną jest 0. Nie jest to opcja zalecana, powoduje zmniejszenie czytelności kodu. Może także powodować niespodziewane błędy, domyślną wartością dla referencji   
do pliku jest pusta referencja.

W przypadku prostych wyborów dwustanowych warto zamiast struktury **Case** stosować węzeł wyboru **Select** (Rys. 6). Znajduje się on w zakładce  
Programming 🡪 Comparison.



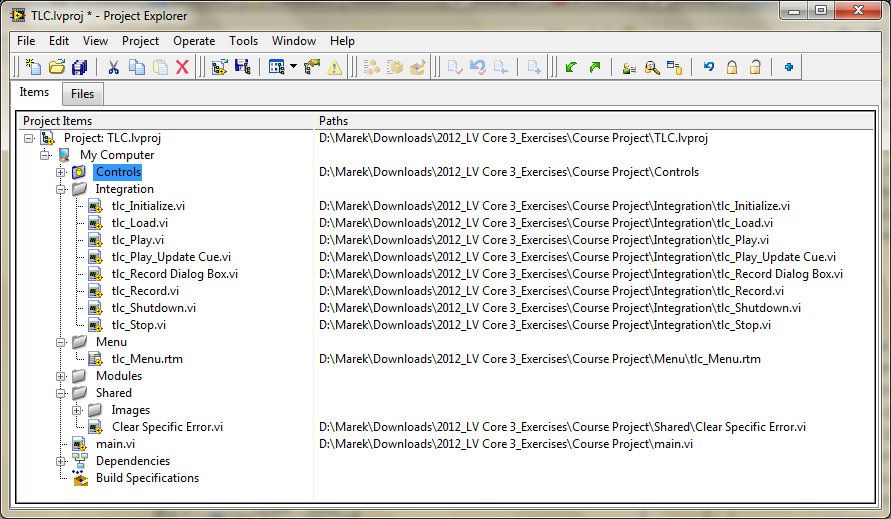
1. Węzeł Select – wygląd oraz opis w pomocy kontekstowej.
   1. **Praca w projekcie**

W trakcie tworzenia aplikacji w środowisku LabVIEW praca z zastosowaniem projektu w niektórych przypadkach jest wymagana. Przykłady, w których wymagana jest praca w projekcie:

- kompilacja programu do pliku uruchamialnego (exe),

- praca z systemami czasu rzeczywistego (RT) oraz modułami FPGA.

Plik projektu grupuje pliki wchodzące w skład projektu. W skład projektu mogą wchodzić pliki vi (programy oraz subVI), biblioteki plików llb, pliki obiektowe  
(dla programowania obiektowego w LabVIEW) oraz inne typy plików (np. dokumentacja w postaci dowolnych formatów plików, dodatkowe pliki graficzne itd.). Pliki są skatalogowane w wirtualnych katalogach, których struktura jest tworzona przez programistę i jest niezależna od struktury plików na komputerze. Przykładowe okno projektu zostało przedstawione na Rys. 7. Widok drzewa katalogów (kolumna: **Path**) uaktywnia się   
wybierając Project 🡪 Show Item Paths.



1. Widok okna projektu LabVIEW.

W strukturze wirtualnego drzewa katalogów można umieścić katalogi z dysku na dwa sposoby:

* Add 🡪 Folder (Snapshot) – tworzy gałąź w wirtualnym drzewie na podstawie istniejącej struktury katalogu na dysku, gałąź nie jest aktualizowana w przypadku zmian w katalogu na dysku (ikona wirtualnego folderu) Rys. 8b,
* Add 🡪 Folder (Auto-populating) – tworzy gałąź w wirtualnym drzewie na podstawie istniejącej struktury katalogu na dysku, gałąź jest aktualizowana w przypadku zmian  
  w katalogu na dysku (niebieska ikona folderu z żółtym walcem) Rys. 8c.

|  |  |  |
| --- | --- | --- |
| a) | b) | c) |
| 1. Widok: opcji dodawania w drzewie (a), wirtualnego folderu (b),  folderu typu auto-populating (c). | | |

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)

* Poznanie właściwości i zastosowania struktury wyboru Case:

- na podstawie tabeli konwersji temperatury przygotowanej po 1 ćwiczeniu przygotować plik subVI wykonujący konwersję wartości temperatur między wszystkimi skalami (jako wejście selektora wybrać typ **Enum**),

- ustawić domyślną konwersję ze skali Celsjusza na skalę Kelwina,

- sprawdzić działanie pliku,

- zapisać plik pod nazwą „*temp konwersja.vi*”,

- zamknąć plik,

* Praca w projekcie

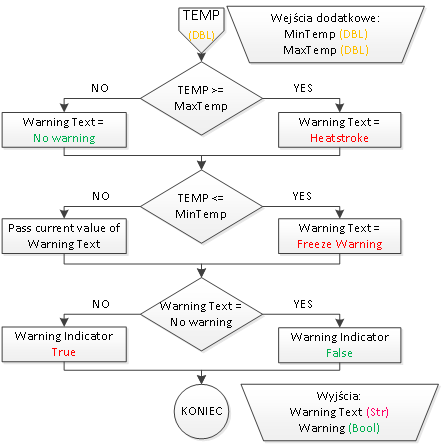
- utworzyć nowy pusty projekt,

- zapisać projekt w nowym katalogu „*Stacja pogodowa/System pomiarowy DAQ*”,

- dodać nowy wirtualny katalog „*subVI*”,

- w katalogu „*subVI*” umieścić plik „*temp konwersja.vi*”

- przygotować plik realizujący decyzyjność działania aplikacji zgodnie z podanym niżej schematem blokowym



1. Schemat blokowy pliku „temp limits.vi”.

- dodać notatkę „***#todo*** *przekształcić w subVI*”,

- dodać notatkę „***#todo*** *zabezpieczyć przed limit min > limit max*”

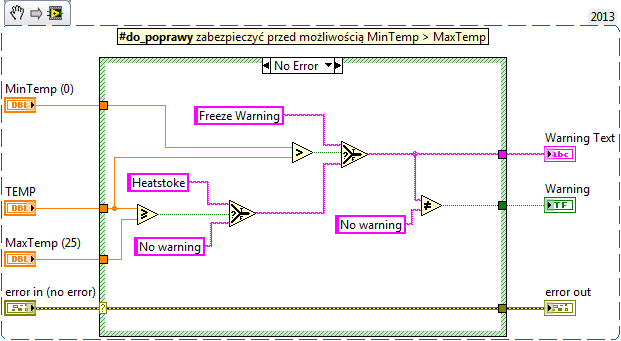
- sprawdzić działanie pliku,

- zapisać plik pod nazwą „*temp limits.vi*”,

- umieścić plik w katalogu „*subVI*”,

1. **PRZYKŁAD REALIZACJI ZADANIA – Plik „*temp limits.vi*”**

Przykładowa realizacja subVI „*temp limits.vi*” z zastosowaniem węzłów **selektor** została przedstawiona na Rys. 10.



1. Przykładowa realizacja pliku subVI - diagram.



1. Przykładowa realizacja pliku subVI - ikona i panel połączeń.

Kontrolki **Min Temp** oraz **Max Temp** ustawiono jako opcjonalne (**Optional**), wartości domyślne podano w nawiasach, kontrolka **TEMP** ustawiona jako wymagana (**Required**). Dla przypadku Error wskaźniki przyjmują następujące wartości: **Warning** – „***True***”, **Warning Text** – „***Error in***”.

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. Jaki będzie stan operacji: „Stan 1” + „Akcja 1” dla zmiennej typu enum na Rys. 5?
3. Wymienić zalety stosowania projektu w LabVIEW.

# 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 Kelwina, Celsjusza i Fahrenheita

|  |  |  |  |
| --- | --- | --- | --- |
| Współczynniki (a,b) | Kelwin, K | Celsjusz, oC | Fahrenheit, oF |
| Kelwin, K | 1, 0 |  |  |
| Celsjusz, oC |  | 1, 0 |  |
| Fahrenheit, oF |  |  | 1, 0 |

# ZADANIA DODATKOWE PO WYKONANIU ĆWICZENIA

- uzupełnić plik „*temp limits.vi*” zgodnie z konwencją plików subVI

- uzupełnić plik „*temp limits.vi*” aby zabezpieczyć przed odwrotnym podpięciem wejść ograniczających zakres dopuszczalnej temperatury **Temp min** oraz **Temp max**.

# ZAŁĄCZNIKI

## Linkowanie tuneli wejściowych z wyjściowymi

Często zdarza się,…

## Pliki polimorficzne

Pliki polimorficzne składają się z wielu