|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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. | **Pętla While** | 4 |
| 2.2. | **Pętla FOR** | 4 |
| 2.3. | **Struktury sekwencyjne** | 5 |
| 2.4. | **Klaster błędu** | 6 |
| 3. | **LABORATORYJNE STANOWISKO BADAWCZE** | 7 |
| 3.1. | **Obiekt badany** | 7 |
| 3.2. | **Urządzenia dodatkowe** | 7 |
| 3.3. | **Oprogramowanie** | 7 |
| 4. | **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI** | 7 |
| 5. | **PRZYKŁAD REALIZACJI ZADANIA – średnia biegnąca** | 8 |
| 6. | **RAPORT** | 9 |
| 7. | **PYTANIA** | 9 |

# SPIS RYSUNKÓW

|  |  |  |
| --- | --- | --- |
| 1. | Pętla While w podstawowej strukturze programu. | 4 |
| 2. | Pętla FOR: widok ogólny (a), generacja 10-cio elementowego wektora o losowych wartościach (b). | 5 |
| 3. | Widok struktur sekwencyjnych: płaskiej (a) oraz nakładanej (b). | 5 |
| 4. | Pomiar czasu trwania programu generującego wektor 100 000 losowych próbek. | 6 |
| 5. | Wskaźnik błędu (a), stała błędu rozłożona na składowe (b), informacja o typie danych klastra błędu (c). | 6 |
| 6. | Aplikacja pomiarowa, której wykonanie jest sterowane za pomocą klastra błędu. | 7 |
| 7. | Widok programu do akwizycji danych. | 7 |
| 8. | Zastosowanie i widok wskaźnika *XY Graph* | 8 |
| 9. | Program do liczenia średniej biegnącej. | 9 |
| 10. | Widok okna programu do liczenia średniej biegnącej | 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
* Poznanie najważniejszych informacji nt. klastra błędu

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,

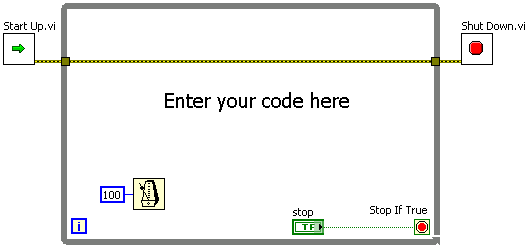
- tunel łączący (**Concatenating**) – służy do uzupełniania wektora kolejnymi komórkami, istnieje tylko jako tunel wyjściowy,

- 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”. Pętla **While** wykonuje się zawsze co najmniej raz – warunek wyjścia sprawdzany jest wewnątrz pętli. Na Przedstawiono wygląd pętli While w podstawowej strukturze programu.



1. Pętla While w podstawowej strukturze programu.

Pętlę **While** stosuje się najczęściej jako główną pętlę programu, warunek zatrzymania jest wywoływany za pomocą przycisku kończącego aplikację (*stop*), może także być zatrzymywana np. w wyniku błędu. Pętlę **While** z niezainicjowanym rejestrem przesuwnym oraz strukturą **case** stosuje się jako tzw. *zmienną funkcyjną* (więcej informacji na temat zmiennych funkcjonalnych będzie na ćwiczeniu ze struktur programistycznych).

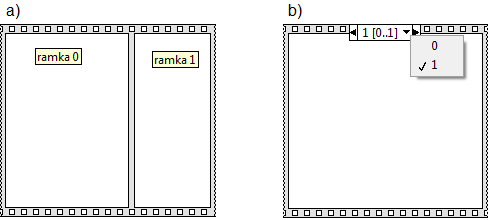
* 1. **Pętla FOR**

Podstawowym zastosowaniem pętli **FOR** są operacje na wektorach, dlatego indeksowanie pętli **FOR** oraz wektorów jest takie samo (od zera). Pętla **FOR** może nie zostać wykonana ani razu, jeżeli na wejście licznikowe zostanie podana wartość „0” lub wektor podany na wejście autoindeksujące ma rozmiar „0”. Opcjonalnym elementem pętli jest terminal warunkowy uaktywniany **PPM** 🡪 **Conditional Terminal**.

|  |  |
| --- | --- |
| a)  forloopconditionalex.bmp | b) |
| 1. Pętla FOR: widok ogólny (a), generacja 10-cio elementowego wektora o losowych wartościach (b). | |

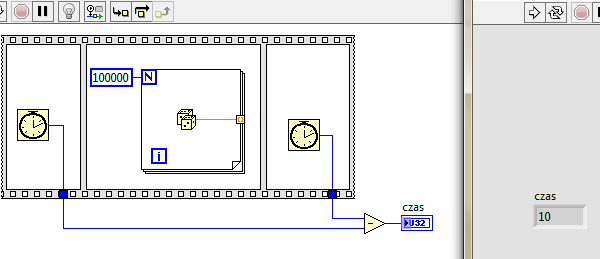
* 1. **Struktury sekwencyjne**

W LabVIEW istnieją dwie struktury sekwencyjne **Flat Sequence Structure** oraz **Stacked Sequence Structure**. Ich widok przypomina kolejne klatki filmu. Ich działanie jest identyczne, różnią się prezentacją graficzną w oknie diagramu. Struktura płaska (**Flat**) jest bardziej czytelna jednak wymaga więcej miejsca ponieważ widać wszystkie klatki. Struktura nakładana (**Stacked**) wyświetla widok tylko jednej klatki na raz, pozostałe są ukryte pod nią więc zajmuje mniej miejsca jednak kod pozostałych kratek jest ukryty, przez co jest mniej czytelny. Można bardzo szybko dokonać zmiany struktury z płaskiej na nakładaną i odwrotnie klikając **PPM** i wybierając odpowiednią opcję z menu podręcznego, struktury sekwencyjne można konwertować także na inne struktury. Struktury sekwencyjne zostały przedstawione na Rys. 3.



1. Widok struktur sekwencyjnych: płaskiej (a) oraz nakładanej (b).

Struktury sekwencyjne umożliwiają kontrolę kolejności wykonywania programu, jednak nie są zalecanym rozwiązaniem. Stosuje się je do ustalenia kolejności kroków programu gdy węzły lub subVI nie mają terminali wejściowych i nie ma możliwości ustalenia kolejności programu w klasycznym dataflow. Przykładem korzystania ze struktur sekwencyjnych jest pomiar czasu trwania fragmentu kodu lub podprogramu – Rys. 4.



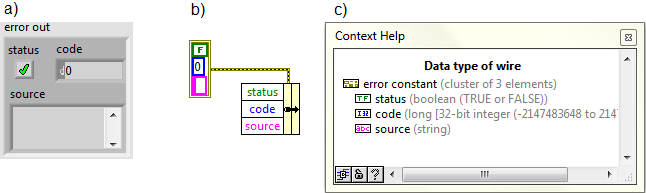
1. Pomiar czasu trwania programu generującego wektor 100 000 losowych próbek.
   1. **Klaster błędu**

Klaster błędu to struktura typu Klaster zawierająca składowe:

- **status** (typu boolean): wartość **False** oznacza brak błędu,

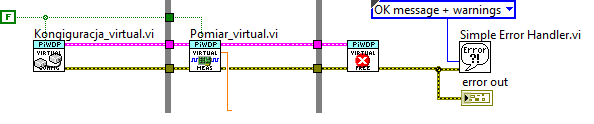
- **kod** (typu 32-integer): zawiera numer błędu, w przypadku gdy kod ma wartość niezerową a status ma wartość **True** wystąpił błąd (**error**), w przypadku gdy kod ma wartość niezerową a status ma wartość **False** wystąpiło ostrzeżenie (**warning**),

- źródło: informuje w którym subVI nastąpił błąd.



1. Wskaźnik błędu (a), stała błędu rozłożona na składowe (b), informacja o typie danych klastra błędu (c).

Klaster błędu jest preferowanym mechanizmem kontroli wykonywania programu. Stosowanie klastra błędu łączącego kolejne subVI wyłącza domyślną obsługę błędów polegającą na natychmiastowym przerwaniu działania aplikacji. W przypadku klastra błędu błąd jest propagowany przez wszystkie subVI aż do punktu, w którym program obsługuje zdarzenie. Przykład prostego programu akwizycji pracującego z prawidłowym zastosowaniem i obsługą klastra błędu przedstawiono na Rys. 6. Podczas zakończenia programu następuje obsługa błędu za pomocą subVI **Simple Error Handler.vi**.



1. Aplikacja pomiarowa, której wykonanie jest sterowane za pomocą klastra błędu.
2. **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**

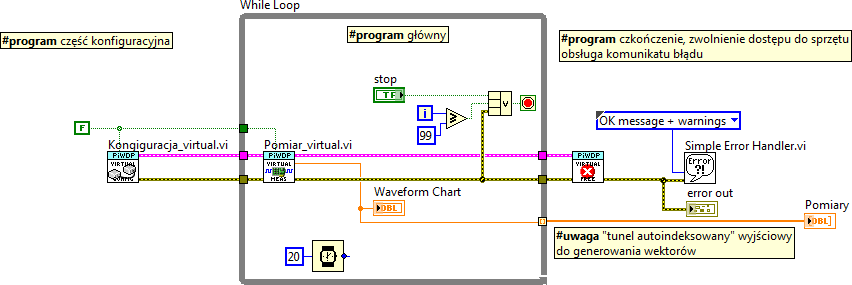
Kolejne kroki do wykonania podczas zajęć:

- uruchomić nowy projekt „*System pomiarowy DAQ*”,

- w oknie projektu utworzyć nowy plik vi na poziomie *My Computer*,

- zapisać plik pod nazwą „*main.vi*” w katalogu „*Stacja pogodowa/*”,

- uzupełnić plik zgodnie z rysunkiem,



1. Widok programu do akwizycji danych.

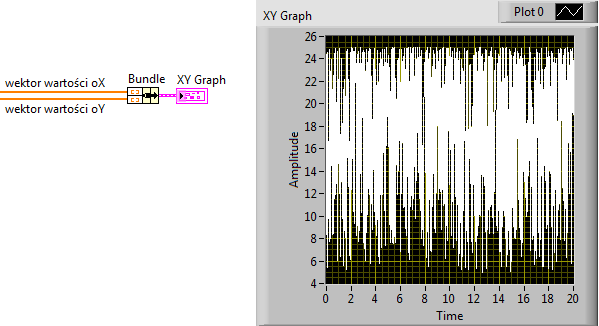
- uzupełnić program główny, aby liczył średnią biegnącą z ostatnich 5 pomiarów,

- zapisać plik,

- uzupełnić program w odpowiednim miejscu, o konwersję danych z karty na stopnie Celsjusza z zastosowaniem pliku „*temp konwersja.vi*”,

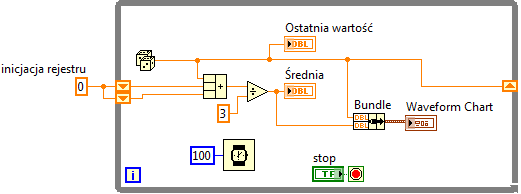
- przygotować wektor danych osi X na podstawie czasu opóźnienia w głównej pętli programu (skalowane w sekundach),

- zaprezentować wyniki za pomocą wskaźnika *XY Graph* (wartości X – podstawa czasu, wartości Y – wyniki pomiarów),

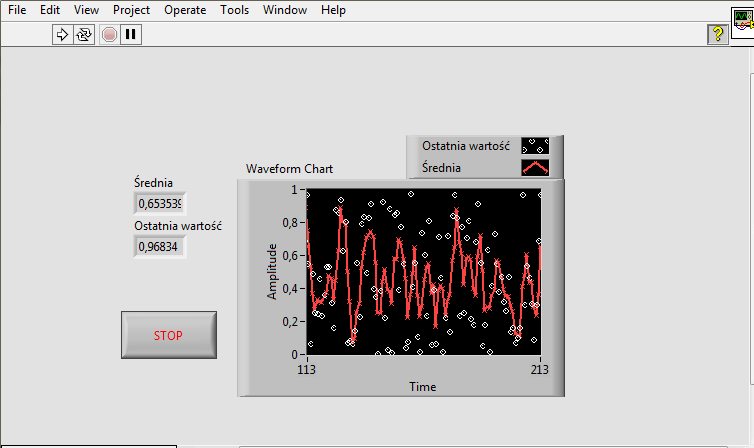


1. Zastosowanie i widok wskaźnika *XY Graph*
2. **PRZYKŁAD REALIZACJI ZADANIA – średnia biegnąca**

Realizacja zadania liczenia średniej biegnącej (**running average**) z 3 ostatnich pomiarów. Na (Program do liczenia średniej biegnącej.Rys. 9) przedstawiono przykład realizacji zadania liczenia średniej biegnącej dla trzech ostatnich wartości wylosowanych. Na (Rys. 10) przedstawiono przykładowy panel programu. Na wskaźniku typu Waveform Chart przedstawiono dwie wartości: Ostatnia wartość – w postaci białych punktów, Średnia – w postaci czerwonego przebiegu z zaznaczonymi punktami.



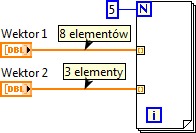
1. Program do liczenia średniej biegnącej.



1. Widok okna programu do liczenia średniej biegnącej
2. **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. Wymienić podobieństwa i różnice pętli **While** oraz **FOR**.
3. Ile razy zostanie wykonana pętla na poniższym rysunku?



1. Omówić właściwości i zastosowania klastra błedu.
2. Ile razy zostaną wykonane kody poniżej?

|  |  |
| --- | --- |
|  |  |
|  |  |

1. Jaką wartość wskaże wskaźnik:

|  |  |
| --- | --- |
| Numeric? | Value Out? |
| Iteration? | Shift Register Answer? |
| Sum po drugim uruchomieniu programu? | Value Out? |

# 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

- uzupełnienie zadań z ćwiczenia 3

# ZADANIA DODATKOWE PO WYKONANIU ĆWICZENIA

- poprawić estetycznie okno programu „*main.vi*”

- poprawić wskaźnik **Waveform Chart** aby prezentował dane w taki sam sposób, jak to zostało przedstawione na Rys. 10.

- poprawić aplikację tak, żeby przetwarzanie danych następowało tylko w przypadku prawidłowo wykonanej akwizycji. Należy zastosować klaster błędu (oraz szablon „*SubVI With Error Handling*”).

- napisać program obliczający iteracyjnie, z dokładnością do 1km, następujące zadanie z fizyki (proponowana prezentacja graficzna wyników):

Odległość między dwoma miastami wynosi s [km]. Z każdego z nich w tej samej chwili wyrusza pociąg w stronę drugiego miasta. Jakie drogi przebędą pociągi do chwili spotkania, jeśli szybkości wynoszą odpowiednio v1 = [km/h] oraz v2 = [km/h].

Kontrolki prezentujące dane wejściowe to:

s, v1, v2, wszystkie typu bouble,

- w podanym zakresie liczb (min do max) typu int wypisać te liczby jako string wielokrotności 3 zastąpić "fizz" wielokrotności 5 zastąpić "buzz" wielokrotności obydwu "fizzbuzz"

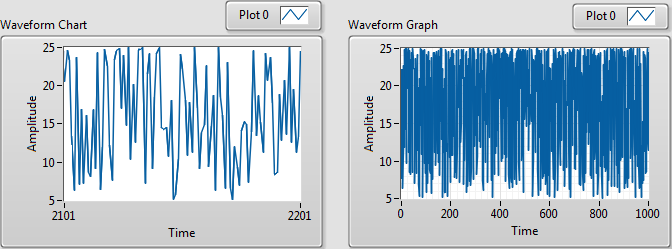
Kontrolki prezentujące dane wejściowe to:

Min, max typu I32

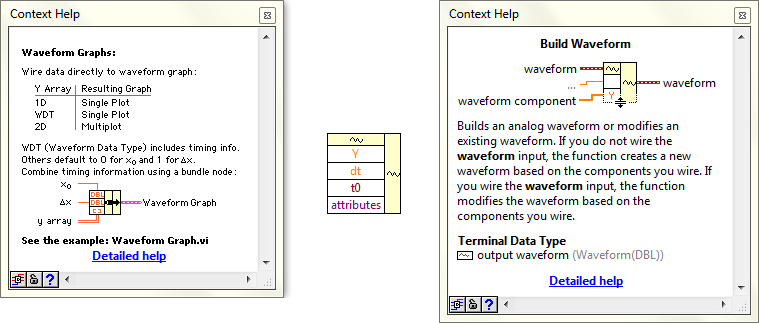
# ZAŁĄCZNIKI

## Waveform Graph vs Waveform Chart

W LabVIEW najczęściej wykorzystywanymi wskaźnikami prezentującymi wykresy są wskaźniki **Waveform Graph** i **Waveform Chart**. Widok obydwu wykresów (z palety **Silver**) został przedstawiony na rysunku.

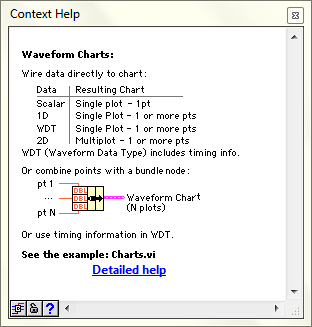


- **Waveform Graph** – prezentuje dane zebrane w postaci wektora (lub przebiegu **waveform**), w przypadku wektora dane wyskalowane są w postaci próbek, w przypadku przebiegu dane są skalowane za pomocą składowej T klastra waveform. **Graph** wyświetla dane całościowo i zazwyczaj po zakończeniu serii pomiarowej. Na rysunku zaprezentowano informację z pomocy kontekstowej oraz strukturę i opis klastra waveform.



- **Waveform Chart** – zbiera dane do bufora, umożliwia wyświetlanie danych „w czasie rzeczywistym”, każda kolejna próbka jest dodawana na końcu bufora. Domyślny rozmiar bufora wynosi 1024 próbki dla jednego kanału. Na wykresie przedstawiona jest określona liczba próbek, może być mniejsza lub równa rozmiarowi bufora, domyślnie na wykresie prezentowane jest ostatnie 100 próbek. **Podstawa osi X skalowana jest w próbkach!**

W celu prezentacji kilku kanałów na jednym wykresie próbki należy zgrupować w klastrach (patrz rysunek)



## Pętla For – dodatkowe informacje

1. Określenie liczby iteracji pętli **For** na podstawie długości wektora (obydwa kody są równoważne),
2. Generacja wektora dwuwymiarowego za pomocą pętli **For**, w LabVIEW można generować wektory o wielu wymiarach, w każdym wymiarze może znajdować się do 231-1 elementów, pierwsze trzy wymiary mają nazwy: długość, wysokość, głębokość (length, height, depth).

|  |  |  |
| --- | --- | --- |
| a) |  | b) |
|  |  |  |