|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| logoKM | logoKM | | POLITECHNIKA ŚLĄSKA  WYDZIAŁ ELEKTRYCZNY  KATEDRA MECHATRONIKI | logoKM |
| Instrukcja do ćwiczenia laboratoryjnego | | | | |
| Przedmiot: | | Przetwarzanie i Wizualizacja Danych Pomiarowych | |  |
| Symbol ćwiczenia: | | PiWDP5 | |  |
| Tytuł ćwiczenia: | | **Grupowanie danych – tablice i klastry** | | |

**SPIS TREŚCI**

|  |  |  |
| --- | --- | --- |
|  | **SPIS RYSUNKÓW** | 2 |
| 1. | **CELE ĆWICZENIA** | 3 |
| 2. | **WPROWADZENIE** | 3 |
| 2.1. | **Wektory** | 4 |
| 2.2. | **Klastry** | 4 |
| 2.3. | **Struktury sekwencyjnePętla While** | 5 |
| 2.4. | **Klaster błęduBłąd! Nie można odnaleźć źródła odwołania.** | 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 |
|  |  |  |

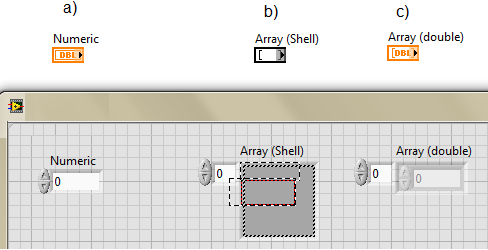
1. **CELE ĆWICZENIA**

Celem ćwiczenia jest zapoznanie się ze złożonymi strukturami danych:

* Wektory
* Klastry

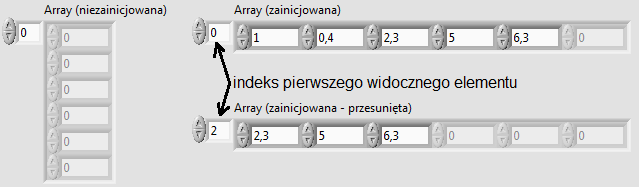
1. **WPROWADZENIE**
   1. **Wektory**

Wektory (**Array**) są złożonymi strukturami danych. Grupują dane jednego typu. Wektory składają się z wymiarów i elementów. **Elementy w wektorze są numerowane od 0!** Można tworzyć jedno i wielowymiarowe wektory, wektory różnych typów lub rozmiarów są niekompatybilne. Wektory tworzy się w powłokach (**Shell**) kolejne kroki tworzenia wektorów są zaprezentowane na Rys. 1.



1. Etapy tworzenia wektora.

Kontrolki, wskaźniki oraz stałe typu wektorowego tworzy się w ten sam sposób. Obiekt wybranego typu przeciąga się do powłoki. W tym etapie tworzy się niezainicjowany wektor wybranego typu. Na Rys. 1 zmienna typu **double** została przeciągnięta do powłoki (**Shell**) wektora (pusta powłoka jest reprezentowana kolorem czarnym). W wyniku tej operacji powstał jednowymiarowy wektor typu double. Wektor ten jest niezainicjowany tzn. nie zawiera żadnych elementów. Zainicjowanie elementów polega na wypełnieniu elementów. Na Rys. 2 przedstawiono widok wektora niezainicjowanego oraz zainicjowanego. W przypadku wektorów 1D elementy mogą być prezentowane w postaci wektora wierszowego lub kolumnowego. W przypadku wektorów 2D kolejność wymiarów jest określona.

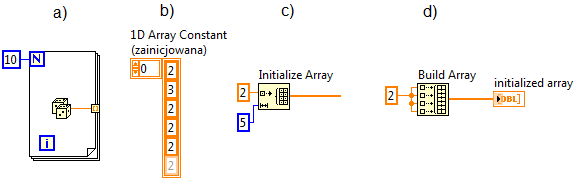


1. Widok niezainicjowanej i zainicjowanej kontrolki wektora 1D.

Pierwszym wymiarem wektora 2D jest wiersz, drugim kolumna.

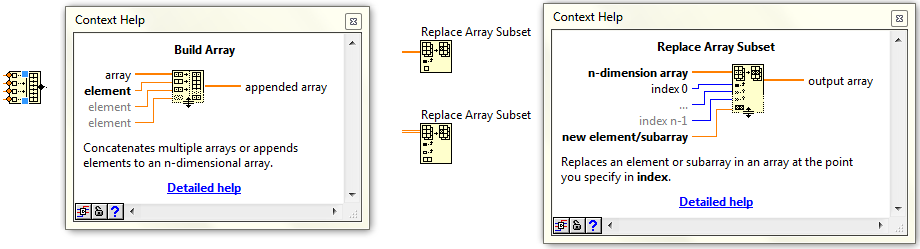
Metody tworzenia wektorów zostały przedstawione na Rys. 3:

1. Zastosowanie pętli for do tworzenia 10-cio elementowego wektora wypełnionego losowymi wartościami,
2. Stała wektor typu double,
3. Zastosowanie węzła Initialize Array do tworzenia wektora 5 elementowego wypełnionego wartościami 2,
4. Zastosowanie węzła Build Array do tworzenia czteroelementowego wektora wypełnionego wartościami 2.



1. Metody tworzenia wektorów.

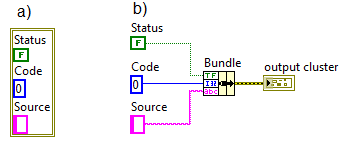
Węzły operujące na wektorach automatycznie dostosowują się do rozmiaru wektora. Mogą także być zmieniane ręcznie. Możliwość ta została przedstawiona na Rys. 4.



1. Widok węzłów oraz ich opis w pomocy kontekstowych
   1. **Klastry**

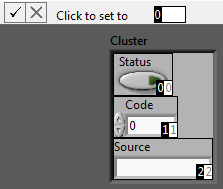
Klastry są złożonymi typami danych zawierającymi dane różnych typów. Są odpowiednikiem struktur i rekordów z tekstowych języków programowania. Wszystkie elementy klastra muszą jednak mieć ten sam „kierunek” (tzn. muszą być kontrolkami lub wskaźnikami). Podobnie jak wektory klastry tworzy się w powłokach. Elementy klastra są ponumerowane. Dwa klastry zawierające te same elementy, jednak ułożone w różnej kolejności są różnymi klastrami. Przykładem jest klaster błędu zawierający elementy: **Status** (**Boolean**), **Kod** (**I32**), **Źródło** (**String**).

Klastry można tworzyć jako kontrolki, wskaźniki i stałe, można także grupować dane za pomocą węzłów **Bundle** oraz **Bundle by Name**. Przykłady tworzenia klastrów zostały przedstawione na Rys. 5.



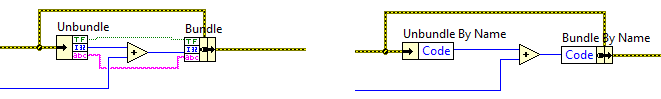
1. Metody tworzenia klastrów.

Widok klastra z zaznaczonymi numerami elementów został przedstawiony na Rys. 6. Organizację klastra można zmieniać: **PPM** 🡪 **Reorder Controls In Cluster**.



1. Klaster z zaznaczoną numeracją elementów

Węzły **Bundle** oraz **Unbundle** można stosować do wszystkich klastrów natomiast klastry **Bundle By Name** oraz **Unbundle By Name** mogą być zastosowane tylko do klastrów, których elementy posiadają nazwy. Węzły **Bundle** i **Unbundle** operują zawsze na wszystkich elementach z zachowaniem ich organizacji. Węzły **Budle By Name** oraz **Unbundle By Name** umożliwiają operowanie na wybranych elementach oraz w dowolnej kolejności. Rys. 7 przedstawia realizację zadania zmiany wartości wybranego elementu klastra za pomocą pary **Unbundle** i **Bundle** oraz **Unbundle By Name** i **Bundle By Name**.



1. Metody realizacji zmiany wartości wybranego elementu.
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**

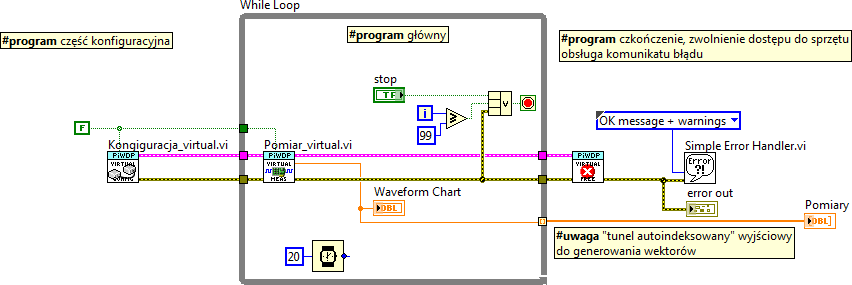
Kolejne kroki do wykonania podczas zajęć:

- 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.

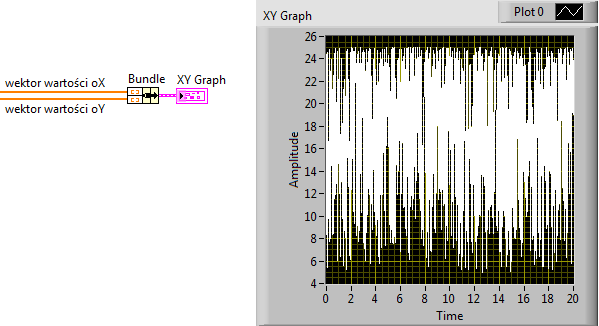
- uzupełnić program główny, aby liczył średnią biegnącą z ostatnich 5 pomiarów,

- zapisać plik,

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

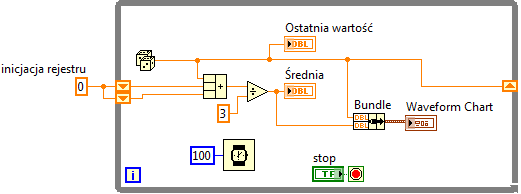
- przygotować wektor danych osi X na podstawie czasu opóźnienia w głównej pętli programu (skalowane w sekundach),

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

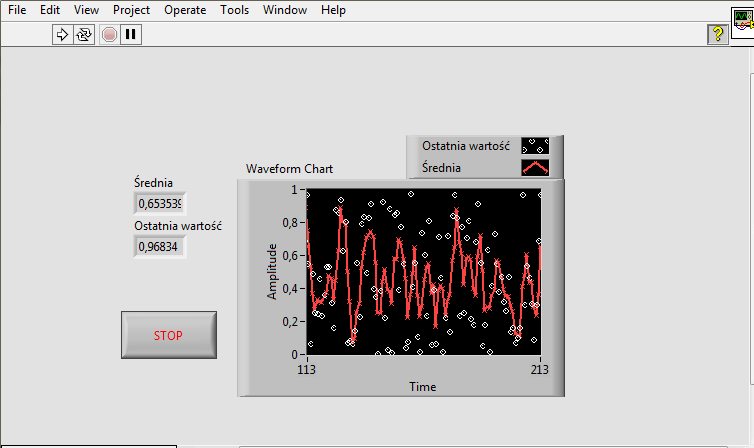


1. Zastosowanie i widok wskaźnika *XY Graph*
2. **PRZYKŁAD REALIZACJI ZADANIA – średnia biegnąca**

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**.

# 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

- brak

# ZADANIA DODATKOWE PO WYKONANIU ĆWICZENIA

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

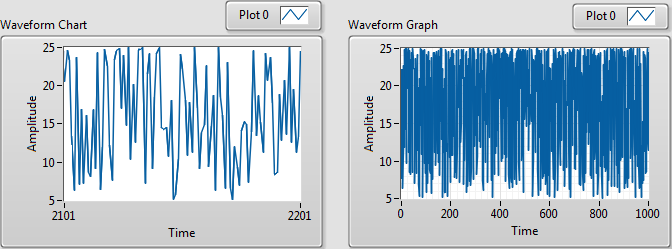
- poprawić wskaźnik **Waveform Chart** aby prezentował dane w taki sam sposób, jak to zostało przedstawione na Rys. 10.

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

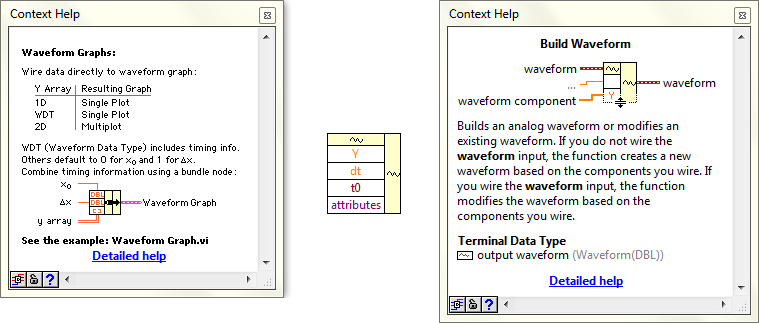
# ZAŁĄCZNIKI

## 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.

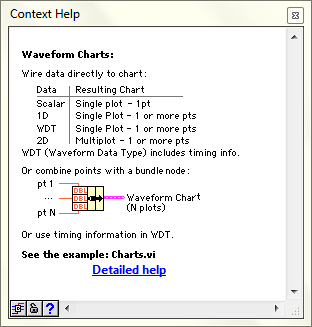


- **Waveform Graph** – prezentuje dane zebrane w postaci wektora (lub przebiegu **waveform**), w przypadku wektora dane wyskalowane są w postaci próbek, w przypadku przebiegu dane są skalowane za pomocą składowej T klastra waveform. **Graph** wyświetla dane całościowo i zazwyczaj po zakończeniu serii 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) |
|  |  |  |