|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| logoKM | logoKM | | POLITECHNIKA ŚLĄSKA  WYDZIAŁ ELEKTRYCZNY  KATEDRA MECHATRONIKI | logoKM |
| Instrukcja do ćwiczenia laboratoryjnego | | | | |
| Przedmiot: | | Przetwarzanie i Wizualizacja Danych Pomiarowych | |  |
| Symbol ćwiczenia: | | PiWDP15 | |  |
| Tytuł ćwiczenia: | | **Poprawa i kompilacja aplikacji** | | |

**SPIS TREŚCI**

|  |  |  |
| --- | --- | --- |
|  | **SPIS RYSUNKÓW** | 2 |
| 1. | **CELE ĆWICZENIA** | 3 |
| 2. | **WPROWADZENIE** | 3 |
| 2.1. | **Struktura VI-Server** | 3 |
| 2.2. | **Budowa hierarchiczna komponentów panelu** | 3 |
| 2.3. | **Programowa obsługa właściwości i metod obiektów** | 4 |
| 2.4. | **Programowa obsługa metod** | 6 |
| 3. | **LABORATORYJNE STANOWISKO BADAWCZE** | 6 |
| 3.1. | **Obiekt badany** | 6 |
| 3.2. | **Urządzenia dodatkowe** | 6 |
| 3.3. | **Oprogramowanie** | 6 |
| 4. | **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI** | 6 |
| 5. | **PRZYKŁAD REALIZACJI ZADANIA – właściwości *Value* oraz *Value (Signaling)*** | 8 |
| 6. | **RAPORT** | 8 |
| 7. | **PYTANIA** | 8 |

# SPIS RYSUNKÓW

|  |  |  |
| --- | --- | --- |
| 1. | Struktura właściwości kontrolki typu *Numeric* | 4 |
| 2. | Terminal kontrolki Numeric (a), bezpośredni dostęp do właściwości (b), pośredni dostęp do właściwości (c) | 5 |
| 3. | Kontrolka Numeric i obsługa jej przykładowych właściwości | 5 |
| 4. | Metody komponentu Waveform Graph | 6 |
| 5. |  |  |
| 6. |  |  |
| 7. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **CELE ĆWICZENIA**

Celem ćwiczenia jest poznanie się z:

* Podstawą pracy struktury VI-Server,
* Podstawą programowania obiektowego w LabVIEW,
* Bezpośrednimi i pośrednimi referencjami do obiektów – komponentów panelu,
* Programowym zarządzaniem właściwościami komponentów panelu,
* Programowym wywoływaniem metod komponentów panelu.

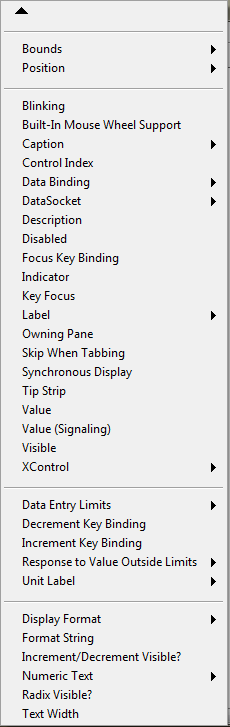
1. **WPROWADZENIE**
   1. **Struktura VI-Server**

Środowisko programistyczne LabVIEW bazuje na programowaniu obiektowym (PO). Wszystkie komponenty panelu czołowego są tworzone zgodnie z PO, posiadają swoje właściwości i metody, do których programista może odwoływać się w trakcie pisania programów. Strukturą zarządzającą pracą PO jest LabVIEW VI Server, umożliwia on w sposób programowy kontrolować obiekty panelu czołowego, inne aplikacje VI. Umożliwia on dynamiczną edycję oraz uruchamianie plików VI z innej aplikacji tworząc np. aplikacje bazujące na strukturze wtyczek, lub tworząc wielozadaniową aplikację z nadrzędnym plikiem VI wywołującym odpowiednie SubVI w zależności od zadania do realizacji. LabVIEW VI Server umożliwia także pracę zdalną przez sieć.

* 1. **Budowa hierarchiczna komponentów panelu**

Jedną z cech PO jest dziedziczenie polegające na tworzeniu hierarchii klas. Budowanie struktury klas polega na możliwości dziedziczenia pewnych właściwości z klas o mniejszym poziomie abstrakcji (rodzic) przez klasy o wyższym poziomie abstrakcji.

Przykładowa struktura właściwości została przedstawiona na Rys. 1. Kontrolka *Numeric* dziedziczy z klas application oraz component, posiada także specyficzne właściwości swojej klasy.



1. Struktura właściwości kontrolki typu *Numeric*
   1. **Programowa obsługa właściwości i metod obiektów**

Właściwości komponentów panelu dostępne są w węzłach **Property Node**, natomiast metody w węzłach **Invoke Node**. Dostęp do nich możliwy jest przez:

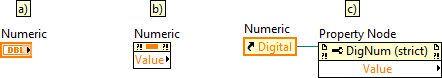
**PPM** 🡪 **Create** 🡪 **Property Node**/**Invoke Node**

oraz wybranie odpowiedniej właściwości/metody. Drugą metodą jest wybranie w oknie diagramu:

**PPM** 🡪 **Application** 🡪 **Property Node**/**Invoke Node**

W obydwu przypadkach zostanie utworzony węzeł właściwości/metody. Jednak w drugim przypadku nie będzie on domyślnie powiązany z żadnym obiektem, wymagał będzie podanie referencji do obiektu: **PPM** 🡪 **Create** 🡪 **Reference**.

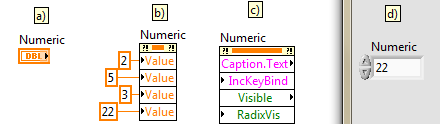
W pierwszym przypadku mamy bezpośredni a w drugim pośredni dostęp do właściwości obiektu. Na Rys. 2 przedstawiono kontrolkę typu Numeric (a) oraz metodę bezpośrednią (b) oraz pośrednią (c) dostępu do właściwości **Value** (wartość). Właściwość Value umożliwia w sposób programowy wprowadzanie wartości do kontrolki z poziomu aplikacji, cechą charakterystyczną programowej modyfikacji wartości jest fakt, że taka operacja nie jest wykrywana przez strukturę Event.



1. Terminal kontrolki Numeric (a), bezpośredni dostęp do właściwości (b), pośredni dostęp do właściwości (c)

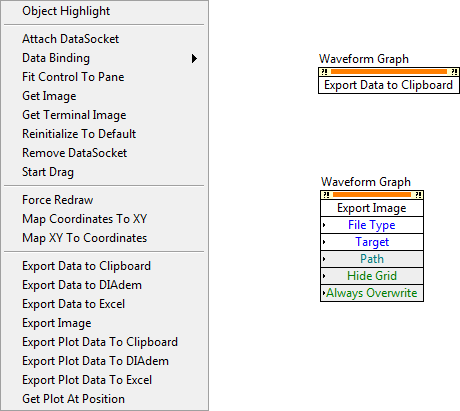
Zaletą bezpośredniego dostępu jest łatwość i czytelność kodu, jednak może być zastosowana TYLKO w tym samym pliku VI, w którym znajduje się komponent, do którego odwołuje się, zaletami dostępu pośredniego jest możliwość modyfikacji właściwości wielu komponentów (o ile mają wspólną właściwość) oraz dostęp do właściwości z poziomu SubVI.

Pojedynczy węzeł **Property Node** może obsługiwać wiele właściwości na raz, każda właściwość ma określony swój kierunek danych – wejście lub wyjście. Kierunek pojedynczej właściwości może być zmieniany **PPM** 🡪 **Change to Write/Read**, kierunek wszystkich właściwości może być ustawiony **PPM** 🡪 **Change All to Write/Read**. Kolejność właściwości jest wykonywana od góry do dołu, w przypadku, jeżeli w jednym Property Node kilkakrotnie modyfikujemy jedną właściwość ostatecznie będzie miała ona wartość najniżej wpisanej. Na Rys. 3 Przedstawiono kontrolkę Numeric (a), przypadek wpisania różnych wartości do jednej właściwości (b) oraz efekt uruchomienia (d) oraz możliwość obsługi różnych właściwości w różnych kierunkach (c).



1. Kontrolka Numeric i obsługa jej przykładowych właściwości
   1. **Programowa obsługa metod**

Metody dostępne są w węźle Invoke Node. W przeciwieństwie do właściwości, jeden węzeł Invoke obsługuje tylko jedną metodę. Jeżeli metoda wymaga podania dodatkowych parametrów lista tych parametrów wyświetlana jest w węźle. Na Rys. 4 przedstawiono listę metod komponentu **Waveform Graph** oraz przykładowe węzły Invoke Node z metodą nie wymagającą parametrów (Export Data to Clipboard) oraz wymagającą parametry (Export Image).



1. Metody komponentu Waveform Graph
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:

* Obsługa właściwości komponentów panelu użytkownika

- otworzyć projekt „*System pomiarowy DAQ*”,

- utworzyć nowy plik „*Property i event.vi*”,

- przepisać kod z przykładu (Rozdział 5),

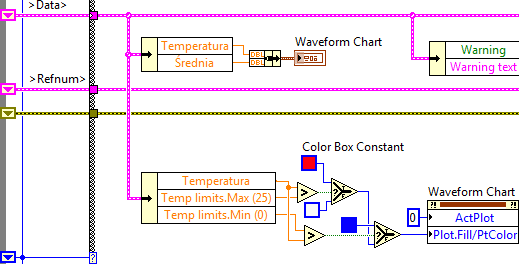
- sprawdzić działanie aplikacji,

- zamknąć aplikację,

- otworzyć plik „*Main\_maszyna.vi*”,

- przejść do stanu **Display**,

- dopisać fragment kodu przedstawiony na rysunku:



- zapisać fragment kodu zmieniający kolor wykresu w postaci SubVI,

- przejść do stanu **Save**,

- korzystając z właściwości *History* czyścić wskaźnik Waveform Chart (wpisać stałą zawierającą puste dane),

- przejść do stanu **Config**,

- korzystając z właściwości *History* czyścić wskaźnik Waveform Chart,

- obliczyć z jakim odstępem czasowym wykonywane są kolejne pomiary,

- dodać właściwość XScale.Multiplier (wpisać wartość z poprzedniego kroku lub, w przypadku nie zrealizowania, wpisać wartość 0,4),

- dodać właściwość XScale.Maximum i wpisać wartość 10,

- sprawdzić działanie aplikacji,

- zapisać aplikację,

- zamknąć aplikację i projekt.

1. **PRZYKŁAD REALIZACJI ZADANIA – właściwości *Value* oraz *Value (Signaling)***

Właściwości **Value** oraz **Value (Signaling)** wprowadzają (lub wyprowadzają) wartość z danej kontrolki, umożliwiają wprowadzenie wartości początkowej do kontrolki lub odczytanie jej wartości w innym miejscu programu. Obsługa tych właściwości jest analogiczna do działania **zmiennej lokalnej**. Istotną różnicą między działaniem **zmiennej lokalnej** oraz właściwości **Value** a działaniem właściwości **Value (Signaling)** jest fakt, że ostatnia właściwość, poza modyfikacją wartości kontrolki/wskaźnika aktywuje zdarzenie **Value change** wykrywane przez strukturę **Event**. Aplikacja została przedstawiona na Rys. 5 oraz Rys. 6, kod zawarty na Rys. 5 prezentuje wszystkie 5 zaprogramowanych zdarzeń struktury **Event** ([0] – [4]).

Aplikacja zawiera następujące komponenty:

- przycisk **STOP** typu *boolean*,

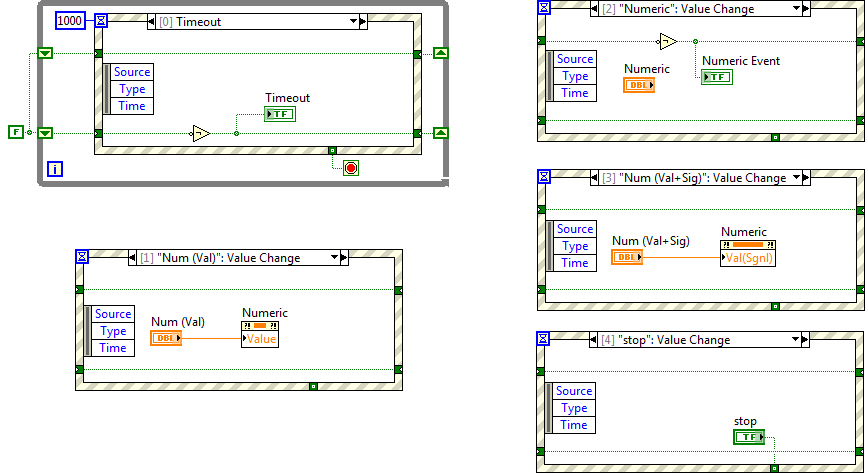
- kontrolkę **Numeric** typu *double*,

- kontrolkę **Num (Val)** typu *double*,

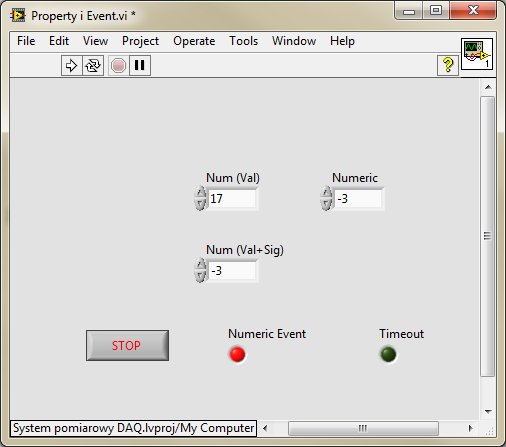
- kontrolkę **Num (Val+Sig)** typu *double*,

- wskaźnik **Numeric Event** typu *boolean*,

- wskaźnik **Timeout** typu *boolean*.



1. Aplikacja prezentująca działanie właściwości Value oraz Value (Signaling) – kod programu



1. Aplikacja prezentująca działanie właściwości Value oraz Value (Signaling) – panel użytkownika
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. Które z następujących elementów zaliczają się do klasy VI a które do klasy Control
3. Format and Precision,
4. Blinking,
5. Reinitialize to Default Value,
6. Show Tool Bar.
7. ee

# 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. https://www.youtube.com/watch?v=iNm0zWY7o8g&list=PLUnVykytJXxPxm5u0vRKpPRVVg2u\_WgFg&index=3
6. https://www.youtube.com/watch?v=RuIN31rSO2k

Opracowanie: Marek Kciuk

# ZADANIA DO REALIZACJI PRZED PRZYSTĄPIENIEM DO ĆWICZENIA

- ukończyć zadania z instrukcji nr 14.

# ZADANIA DODATKOWE PO WYKONANIU ĆWICZENIA

-

# ZAŁĄCZNIKI

## Opóźnienia i zarządzanie czasem działania aplikacji