|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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. | **Poprawa struktury aplikacji** | 3 |
| 2.2. | **Kompilacja aplikacji do programu wykonywalnego** | 4 |
| 2.3. |  |  |
| 2.4. |  |  |
| 3. | **LABORATORYJNE STANOWISKO BADAWCZE** | 5 |
| 3.1. | **Obiekt badany** | 5 |
| 3.2. | **Urządzenia dodatkowe** | 5 |
| 3.3. | **Oprogramowanie** | 5 |
| 4. | **PROGRAM ĆWICZENIA – WYKAZ ZADAŃ DO REALIZACJI** | 5 |
| 5. | **PRZYKŁAD REALIZACJI ZADANIA – właściwości *Value* oraz *Value (Signaling)*** | 6 |
| 6. | **RAPORT** | 8 |
| 7. | **PYTANIA** | 8 |

# SPIS RYSUNKÓW

|  |  |  |
| --- | --- | --- |
| 1. | Przykłady modyfikacji prowadzenia przewodów (połączenie z tunelem oraz przewód przechodzący przez węzeł). | 4 |
| 2. |  |  |
| 3. |  |  |
| 4. |  |  |
| 5. |  |  |
| 6. |  |  |
| 7. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **CELE ĆWICZENIA**

Celem ćwiczenia jest poznanie się z:

* Metodami poprawy aplikacji,
* Zastosowaniem narzędzia Application Builder do kompilowania programu,
* Zastosowaniem narzędzia Application Builder do tworzenia wersji instalacyjnej programu.

1. **WPROWADZENIE**
   1. **Poprawa struktury aplikacji**

W trakcie pisania aplikacji dochodzi do sytuacji, kiedy kod staje się mało czytelny. Należy wtedy zmodyfikować kod, aby poprawić czytelność, zarządzalność i stabilność. Pojęcie związane z poprawą kodu z angielskiego nazywa się **Refactoring** i oznacza proces przeprojektowania oprogramowania w celu uczynienia go bardziej czytelnym zarządzanym w celu ułatwienia wprowadzania modyfikacji.

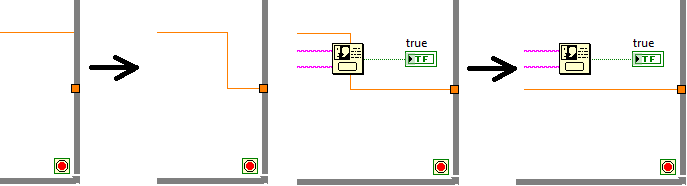
Kod wymaga poprawy jeżeli:

* nie działa w sposób właściwy,
* realizuje tylko część wymagań,
* jest nieczytelny dla innych programistów lub ciężko wrócić do niego po dłuższej przerwie.

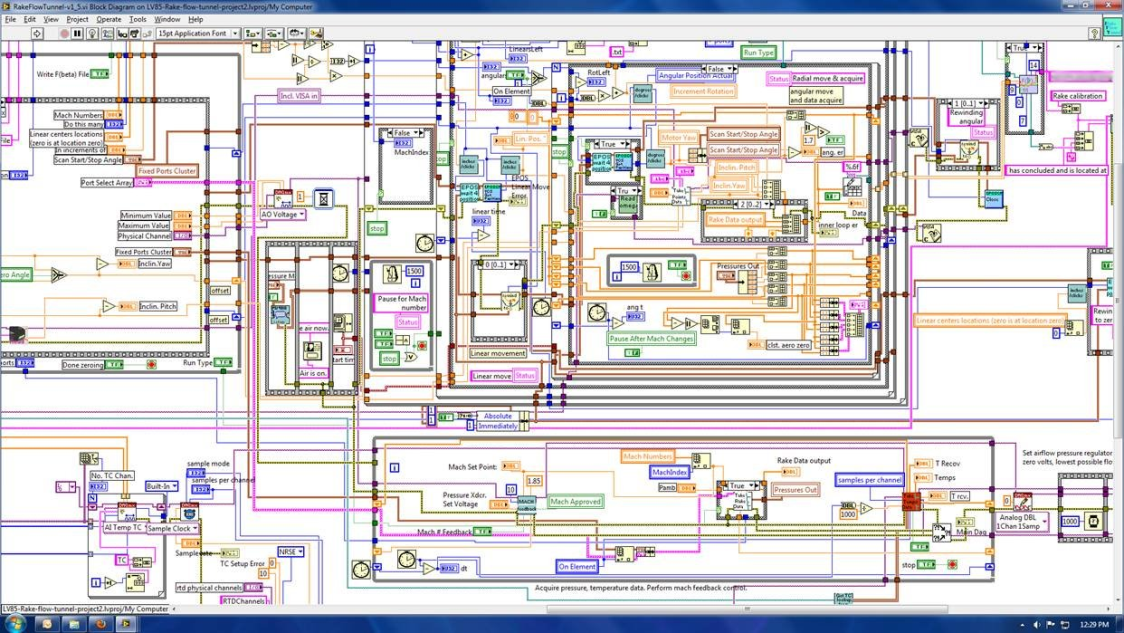
W trakcie poprawy struktury programu należy zwrócić uwagę na następujące cechy pisania kodu w środowisku LabVIEW:

* niepotrzebne załamania przewodów,
* przewody dochodzące do boku struktury, ale nie wchodzą do tunelu,
* przewody zakryte są przez węzły programu,
* przygotowanie zrozumiałych ikon subVI prezentujących ukryty kod,
* przerwanie struktury przepływu danych dataflow,
* zamknięcie powtarzających się fragmentów kodu w postaci plików subVI (modułowość),
* optymalizacja kodu z zastosowaniem wbudowanych funkcji LabVIEW,
* modyfikacja struktur danych i zapisanie ich w postaci definicji typu,
* redukcja zagnieżdżenia struktur *Case*,
* modyfikacja struktur programistycznych w celu uzyskania rozwiązań skalowalnych,
* zmiana struktury aplikacji z bazującej na poolingu na bazującą na zdarzeniach.

Kod niewłaściwie napisany, w którym nie wiadomo o co chodzi – niezdatny do jakiejkolwiek analizy nosi potoczną nazwę „spaghetti code” – Rys. 2.



1. Przykłady modyfikacji prowadzenia przewodów (połączenie z tunelem oraz przewód przechodzący przez węzeł).



1. Niewłaściwie napisany „spaghetti code”. [5]

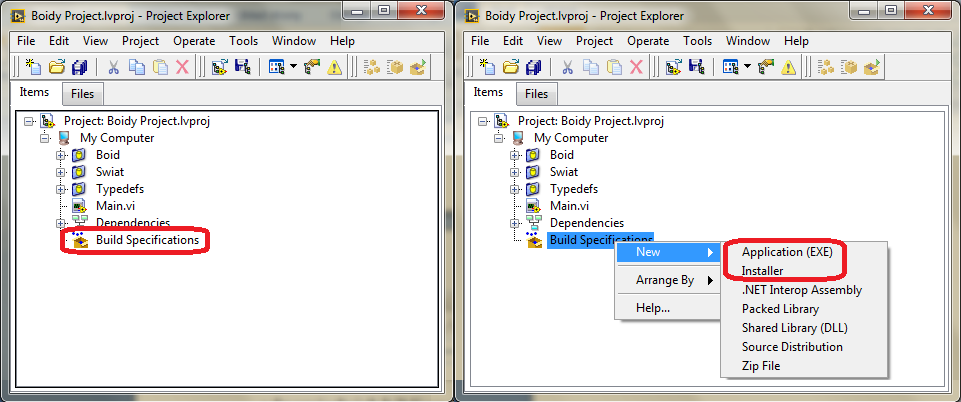
Realizacja zadania poprawy struktury aplikacji polega na cyklicznym wprowadzaniu zmian oraz sprawdzaniu jego funkcjonalności. Jeżeli modyfikacja nie wpływa negatywnie na działanie aplikacji to może być zatwierdzona i można analizować następny fragment kodu.

* 1. **Kompilacja aplikacji do programu wykonywalnego**

Gotowa aplikacja jest dostarczana do klienta w postaci skompilowanego pliku wykonywalnego (w systemie Windows jest to plik z rozszerzeniem *exe*). Środowisko LabVIEW umożliwia kompilację programu do pliku wykonywalnego jeżeli ma zainstalowany toolkit **LabVIEW Application Builder** oraz odpowiednią licencję LabVIEW. Uruchomienie skompilowanej aplikacji wymaga zainstalowania bibliotek **LabVIEW Runtime Engine** na docelowym komputerze.

Toolkit LabVIEW Application Builder umożliwia także tworzenie wersji instalacyjnych oprogramowania. W trakcie tworzenia instalacji można uwzględnić dodanie odpowiednich bibliotek Runtime oraz dodatkowych bibliotek związanych z obsługą urządzeń peryferyjnych.

Kompilowanie aplikacji jest realizowane z poziomu projektu. Widok okna projektu z zaznaczonym toolboxem oraz jego opcjami został przedstawiony na



1. Widok okna projektu z Application Builder.
2. **LABORATORYJNE STANOWISKO BADAWCZE**
   1. **Obiekt badany**

- Środowisko programistyczne LabVIEW,

* 1. **Urządzenia dodatkowe**

- brak,

* 1. **Oprogramowanie**
* LabVIEW 2014 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*”,

- otworzyć plik „*Main\_maszyna.vi*”,

- sprawdzić plik pod kątem

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. 4 oraz Rys. 5, kod zawarty na Rys. 4 prezentuje wszystkie 5 zaprogramowanych zdarzeń struktury **Event** ([0] – [4]).

Aplikacja zawiera następujące komponenty:

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

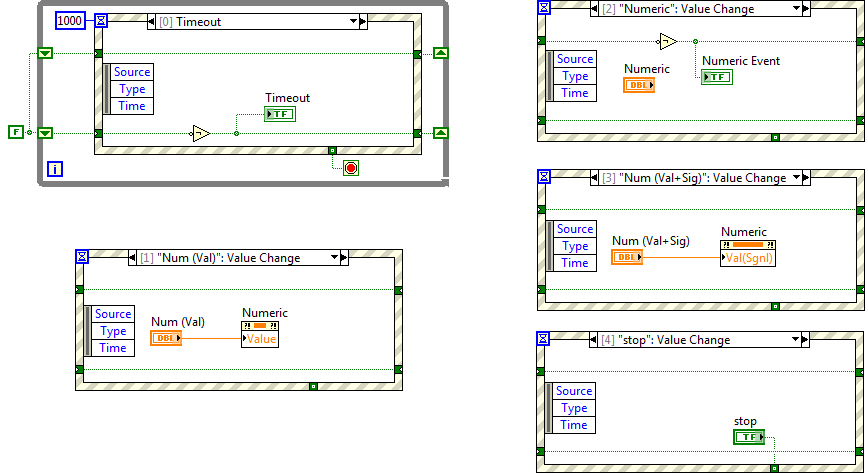
- kontrolkę **Numeric** typu *double*,

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

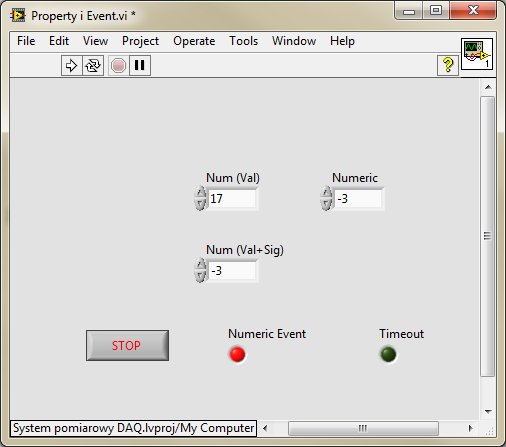
- kontrolkę **Num (Val+Sig)** typu *double*,

- 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. Wymienić cechy dobrze napisanego kodu.
3. Co to jest pooling.

# 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. http://www.paleotechnologist.net/wp-content/uploads/2011/09/LabVIEW\_spaghetti\_code.png
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

## Biblioteka OpenG