|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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** | 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 – przygotowanie pliku vi do kompilacji** | 7 |
| 6. | **RAPORT** | 7 |
| 7. | **PYTANIA** | 8 |

# SPIS RYSUNKÓW

|  |  |  |
| --- | --- | --- |
| 1. | Przykłady modyfikacji prowadzenia przewodów (połączenie z tunelem oraz przewód przechodzący przez węzeł). | 4 |
| 2. | Niewłaściwie napisany „spaghetti code”. [5] | 4 |
| 3. | Widok okna projektu z Application Builder. | 5 |
| 4. | Zakończenie aplikacji z zamknięciem środowiska LabVIEW w przypadku działania pliku skompilowanego. | 7 |
| 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.

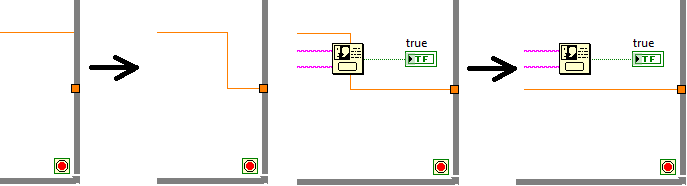
Kod wymaga poprawy jeżeli:

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

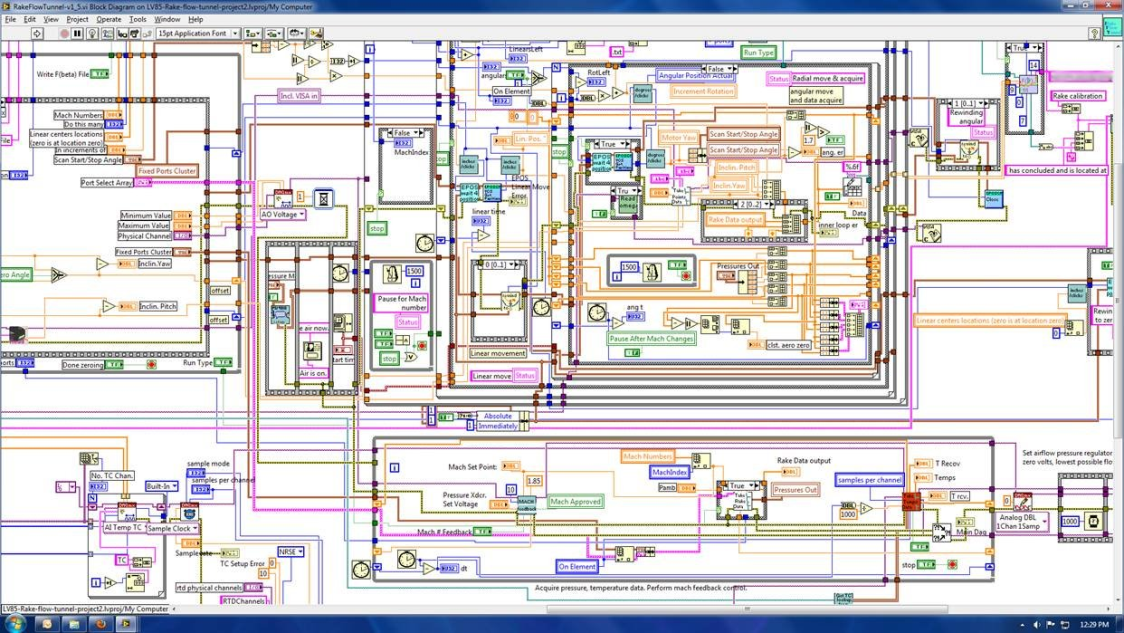
W trakcie poprawy struktury programu należy zwrócić uwagę na następujące cechy pisania kodu w środowisku LabVIEW:

* 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]

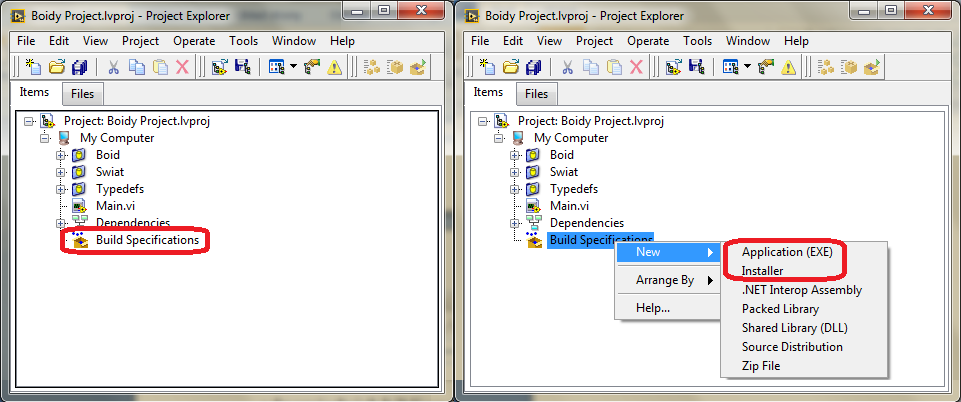
Realizacja zadania poprawy struktury aplikacji polega na cyklicznym wprowadzaniu zmian oraz sprawdzaniu jego funkcjonalności. Jeżeli modyfikacja nie wpływa negatywnie na działanie aplikacji to może być zatwierdzona i można analizować następny fragment kodu.

* 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 np. z obsługą urządzeń peryferyjnych.

Kompilowanie aplikacji jest realizowane z poziomu projektu. Widok okna projektu z zaznaczonym toolboxem oraz jego opcjami został przedstawiony na Rys. 3.



1. Widok okna projektu z Application Builder.

Przygotowanie pliku do kompilacji wymaga wprowadzenia kilku istotnych funkcji. Pamiętać należy, że wszystkie odniesienia do plików (np. pliki konfiguracyjne, pliki danych) mogą być podane w sposób bezwzględny (pełna ścieżka dostępu do pliku) lub względny (ścieżka skrócona, określona względem głównego folderu aplikacji). Zaleca się przygotować dostęp do plików w sposób względny oraz przed uruchomieniem skopiowanie wymaganej struktury katalogów do docelowego katalogu kompilacji.

Program uruchomiony z pliku skompilowanego po wyłączeniu wyłącza się pozostawiając uruchomiony Runtime. W celu prawidłowego zamknięcia aplikacji należy zakończyć także działanie środowisko Runtime. Przykład realizacji zadania został przedstawiony w rozdziale 5.

Przygotowanie instalatora jest również realizowane za pomocą kreatora i musi być zrealizowane po przygotowaniu i sprawdzeniu działania pliku wykonywalnego. Kreator umożliwia także przygotowanie całej wymaganej struktury katalogów i plików.

1. **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:

* Poprawa kodu programu

- otworzyć projekt „*System pomiarowy DAQ*”,

- otworzyć plik „*Main\_maszyna.vi*”,

- sprawdzić plik pod kątem błędów edycyjnych (czas realizacji 40min.),

Zamknąć projekt „System pomiarowy DAQ”.

* Przygotowanie do kompilacji programu

- otworzyć projekt „*System pomiarowy DAQ*”,

- Kliknąć *Build Specifications* **PPM** 🡪 **New...** 🡪 **Application (exe)**,

- postępować zgodnie z wymogami kreatora,

- w zakładce „Source Files” przenieść do okna „Startup VI” plik „Main\_maszyna.vi”,

- zapoznać się z pozostałymi opcjami kreatora i wybrać interesujące,

- skompilować program przyciskiem „Build”,

- po pozytywnie zakończonej kompilacji odnaleźć plik na dysku i uruchomić go,

- sprawdzić działanie aplikacji,

- zmodyfikować ustawienia pliku VI w opcjach w celu ukrycia niepotrzebnych pasków narzędzi oraz rozmiaru okna, opcje te znajdują się w następujących zakładkach:

Window Apperance (wraz z Customize...), Window Size, Window Run-time Position, Execution,

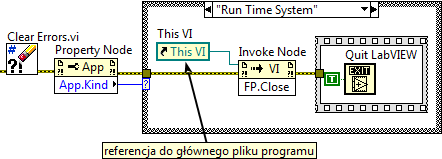
- zmodyfikować główny plik programu w celu prawidłowego zamykania się aplikacji (patrz rozdział 5),

- ponownie skompilować program (nie trzeba ponownie uruchamiać kreatora),

- sprawdzić działanie programu.

1. **PRZYKŁAD REALIZACJI ZADANIA – przygotowanie pliku vi do kompilacji**

Prawidłowo zamykana aplikacja powinna kończąc swoje działanie zamknąć okno programu. W celu realizacji tego zadania w pliku skompilowanym w środowisku LabVIEW należy użyć węzła „Quit LabVIEW”. W celu uzależnienia działania (praca z plikiem VI w środowisku LabVIEW / praca z plikiem skompilowanym) należy uzależnić działanie tego węzła od typu uruchamianej aplikacji. Przykład realizacji tego zadania został przedstawiony na Rys. 4. Przedstawiony na rysunku fragment kodu należy umieścić na końcu aplikacji po sprawdzeniu, zraportowaniu i wyczyszczeniu wszystkich potencjalnych błędów powstałych w trakcie kończenia pracy pętli programu.



1. Zakończenie aplikacji z zamknięciem środowiska LabVIEW w przypadku działania pliku skompilowanego.
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.
4. Potrzebujesz LabVIEW Build Specifications żeby utworzyć następujące rzeczy:
5. Instalator,
6. Plik spakowany ZIP,
7. Pliki VI,
8. Plik wykonywalny.

# 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