Itasdi

Co-funded by the
Erasmus+ Programme
of the European Union

ITASDI PROJECT

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Erasmus+ KA2 2018-1-RS01-KA203-000432

Modifcation of course „Principles of the Virtual Instruments Design“

Instrukcja do laboratorium 8:
Property Nodes, Zmienne współdzielone i funkcjonalne

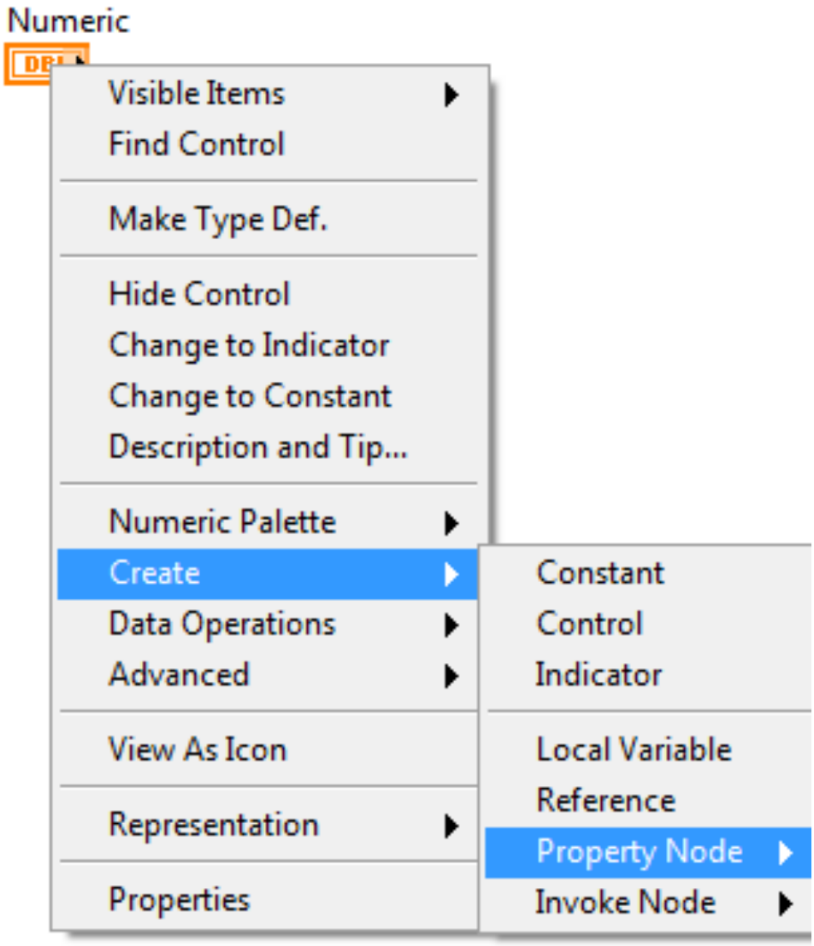
Leader Partner: [1]Warsaw University of Technology
Contribution: [2]University of Novi Sad, [3]Ss. Cyril and Methodius University in Skopje
Authors: Dariusz Tefelski [1], Angelika Tefelska [1], Boris Jakovljevic[2],Dimitar Taskovski[3]

Circulation: Public
Version: 01
Stage: Final
Date: 22.02.2019

Funding Disclaimer
This project has been funded with support from the European Commission. This communication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

1 Property Nodes

Property nodes umożliwiają dostęp do własności obiektów z poziomu *block diagram*, jest to narzędzie szczególnie przydatne w sytuacjach gdy chcemy zmienić własność obiektu w zależności od danych wejściowych. Przykładowo mierzymy temperaturę otoczenia za pomocą czujnika temperatury oraz wyświetlamy ją na wykresie. Chcielibyśmy aby kolor linii na wykresie zmienił się z zielonego na czerwony gdy przekroczymy maksymalną dozwoloną temperaturę. W takiej sytuacji właśnie możemy skorzystać z *property nodes*. Inny przykład to zabezpieczamy pewne dane hasłem, jeśli użytkownik wpisze poprawne hasło to odkryjemy ukryte pola z danymi.

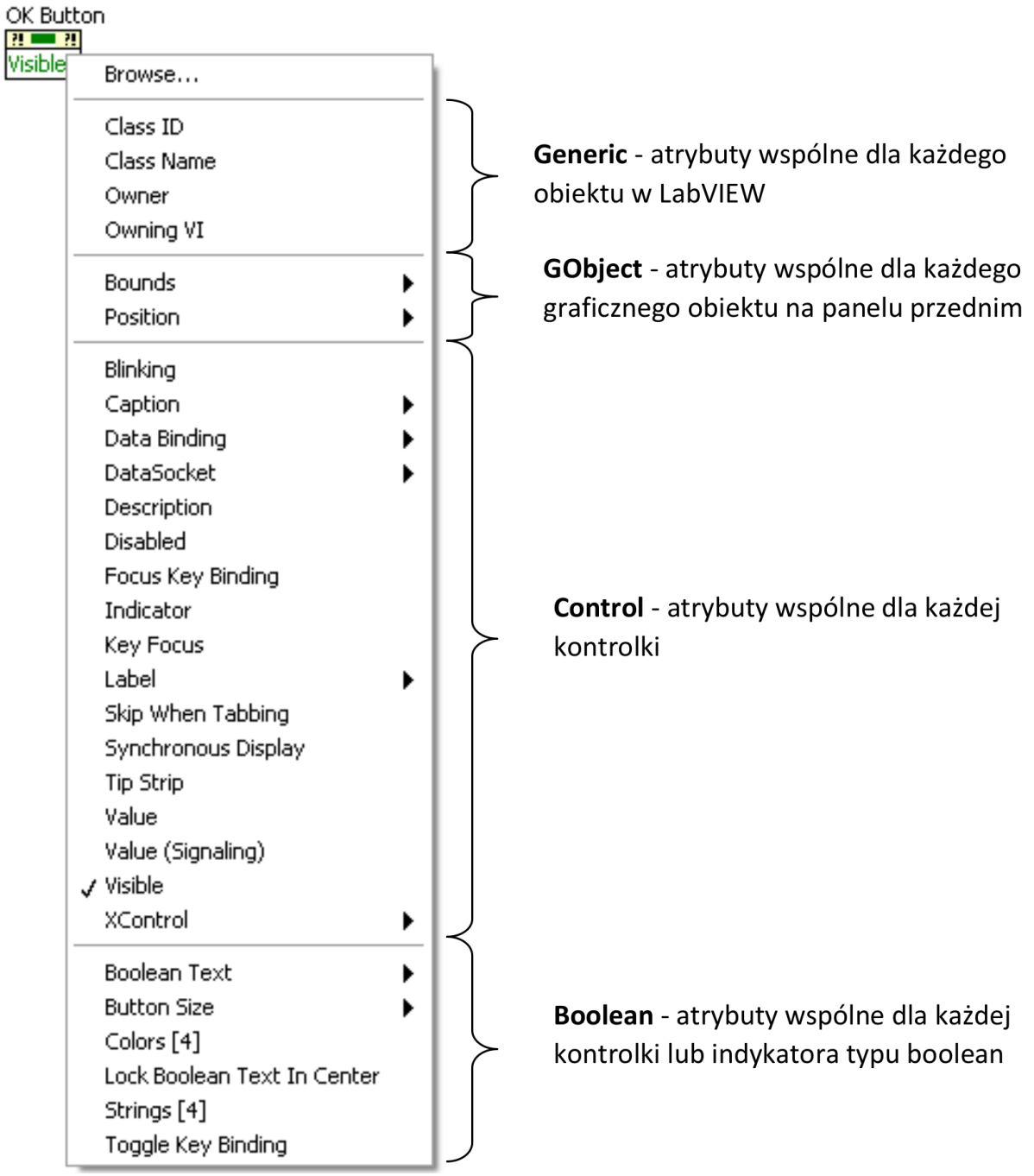


Rysunek 1: Tworzenie *property node*.

Wszystkie parametry kontrolek oraz indykatorów, które mogą być zmieniane za pomocą okienka właściwości bądź menu kontekstowego mogą być również zmienione programowo za pomocą *property nodes*. Aby stworzyć *property node* należy kliknąć prawym przyciskiem myszy na terminal kontrolki bądź indykatora i z menu wybrać opcję **Create** → **Property Node**, jak na rysunku 1. Następnie z listy należy wybrać interesującą nas własność.

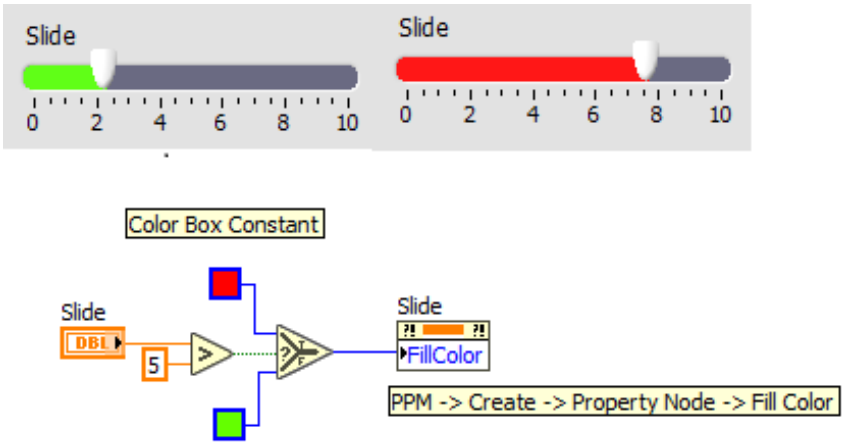
Każda kontrolka ma przyporządkowane właściwości w sposób hierarchiczny - od właściwości ogólnych, wspólnych dla wielu różnych elementów do właściwości szczegółowych. Przykładową listę *property nodes* dla kontrolki typu *boolean* przedstawiono na rysunku 2.

Po wybraniu interesującej własności utworzy się terminal *property node*, który domyślnie będzie do odczytu. Aby zapisać interesującą nas wartość należy kliknąć prawym przyciskiem myszy na terminal *property node* a następnie wybrać **Change to Write**.



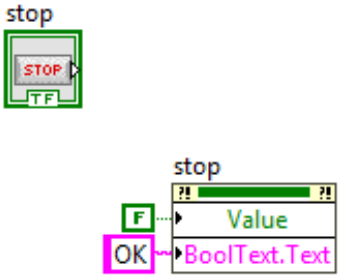
Rysunek 2: Podział hierarchiczny *property nodes*.

Przykładowy program z wykorzystaniem *property node* przedstawiono na rysunku 3. W zależności od wyboru wartości na suwaku *Slide* wypełnienie suwaka zmieni kolor na jeden z dwóch możliwych: zielony lub czerwony. Jeśli wartość będzie większa niż 5 to kolor czerwony zostanie przekazany do *property node* zwanego *Fill Color* a tym samym wypełnienie suwaka zmieni swój kolor na czerwony. W przeciwnym wypadku będzie miał kolor zielony.



Rysunek 3: Przykładowy program z użyciem *property node*.

W niektórych aplikacjach jest wymagana zmiana więcej niż jednej właściwości danego obiektu. W takim przypadku nie trzeba tworzyć dwóch oddzielnych terminali *property node* lecz można obecny terminal rozszerzyć na więcej pozycji. Domyślnie pojawi się kolejny z listy *property node*. Aby wybrać interesującą nas własność, należy na niepożądany *property node* najechać łapką i po kliknięciu prawym przyciskiem myszy wybrać właściwą własność. Przykład pokazano na rysunku 4.



Rysunek 4: Przykład użycia terminala *property node* z dwoma własnościami.

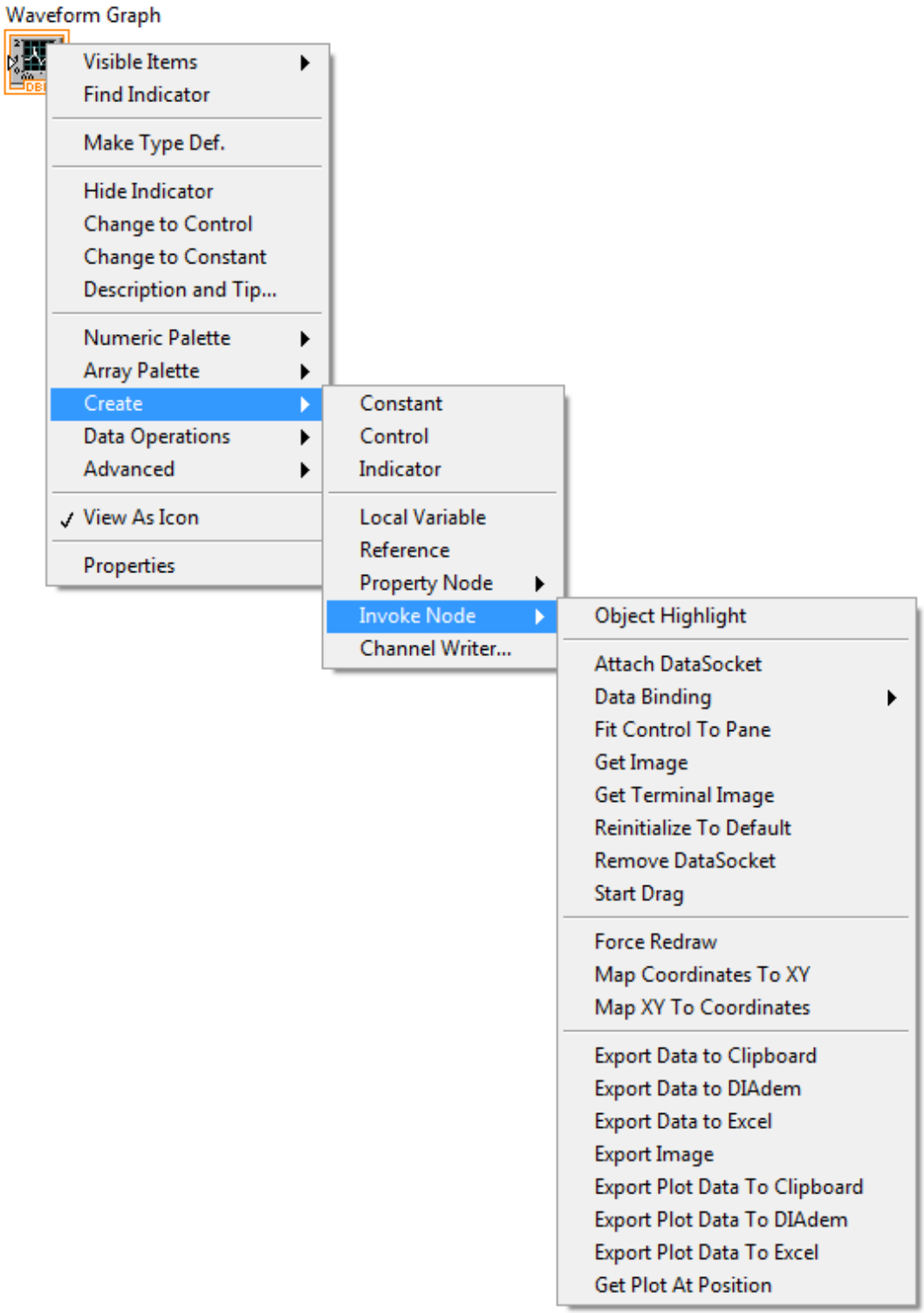
Uwaga 1: Nie wszystkie *property nodes* umożliwiają zapis i odczyt danej wartości. Niektóre własności są tylko do odczytu np. własności *Label* a niektóre tylko do zapisu np. *value signaling property*.

Uwaga 2: Niektóre własności obiektów są przechowywane w klastrze np. pozycja kontrolki. Aby odczytać położenie kontrolki należy użyć funkcji *unbundle*. Natomiast aby zapisać dane do kontrolki trzeba najpierw stworzyć z nich klaster a potem przekazać do *property node*.

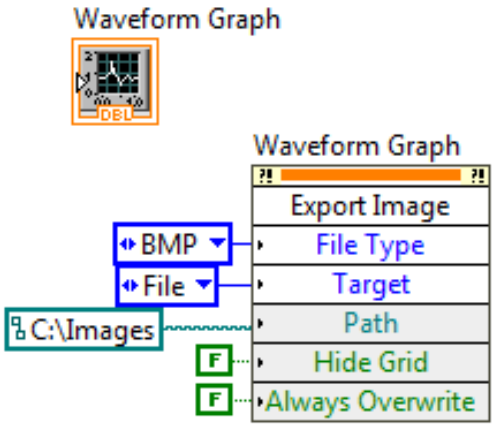
2 Invoke Nodes

Invoke nodes są używane do wywołania metody na obiekcie. W przeciwieństwie do *property nodes*, jeden terminal *invoke node* może realizować tylko jedną metodę. Pierwszy element w terminalu *invoke node* to nazwa metody, następnie wyświetlone są parametry metody: wymagane na białym tle oraz opcjonalne na szarym tle. Jeśli dana metoda zwraca wartość to również terminal *invoke node* będzie zawierać wyjście ze zwracaną wartością. Przykładowy terminal *invoke node* został przedstawiony na rysunku 6.

Aby stworzyć *invoke nodes* należy na danym obiekcie kliknąć prawym przyciskiem myszy a następnie wybrać **Create** → **Invoke Node**. Z wyświetlonej listy należy wybrać interesującą metodę. Sposób tworzenia *invoke node* został przedstawiony na rysunku 5.



Rysunek 5: Tworzenie *invoke node*.

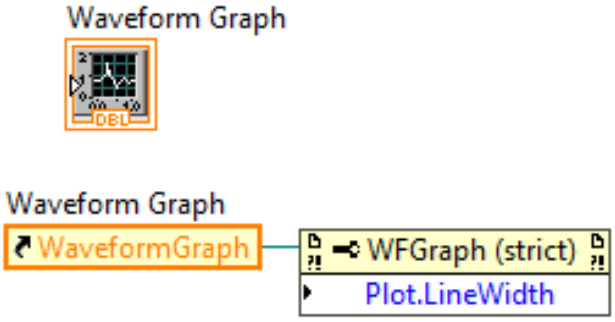


Rysunek 6: Przykładowy terminal *invoke node*.

3 Control References

Opisane wyżej metody tworzenia *property nodes* oraz *invoke nodes* powiązują dany obiekt umiejscowiony na *front panel-u* z konkretnym terminalem *property/invoke node*. W takim przypadku mówimy o niejawnym powiązaniu obiektu z *property/invoke node* (implicitly linked property/invoke nodes). W przypadku stworzenia subVI tracimy powiązanie *property/invoke node* z danym obiektem. W takim wypadku należy podać referencje do obiektu.

Jeśli subVI jest tworzony poprzez zaznaczenie danego fragmentu diagramu a następnie wybranie **Edit** → **Create SubVI** to referencje do obiektów zostaną stworzone automatycznie. Natomiast jeśli subVI tworzony jest poprzez zapis danego programu to należy stworzyć *property node* z palety funkcji. W tym celu wybierz: **Functions palette** → **Programming** → **Application Control** → **Property Node**. Następnie kliknij prawym przyciskiem myszy na obiekt, który chcesz powiązać z *property node* i wybierz **Create** → **Reference**. Powstałą referencje połącz z górnym wejściem terminala *property node* a następnie wybierz interesującą własność do odczytu/zapisu. Przykład przedstawiono na rysunku 7. Tak stworzony *property node* nazywa się jawnie powiązanym *property node* (explicitly linked Property Node). Opisana metoda może również być użyta w przypadku *invoke node*.

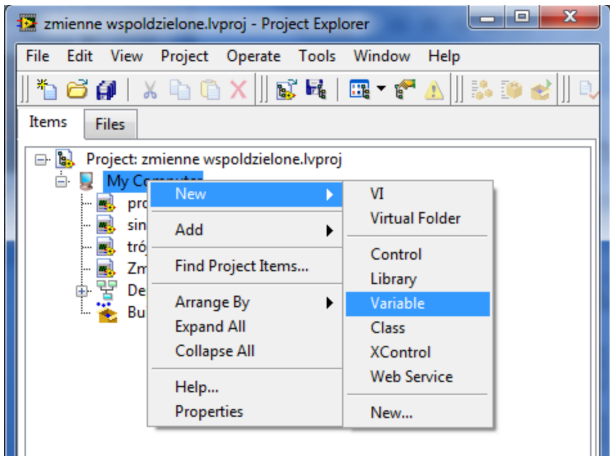


Rysunek 7: Przykład tworzenia *explicitly linked Property Node*.

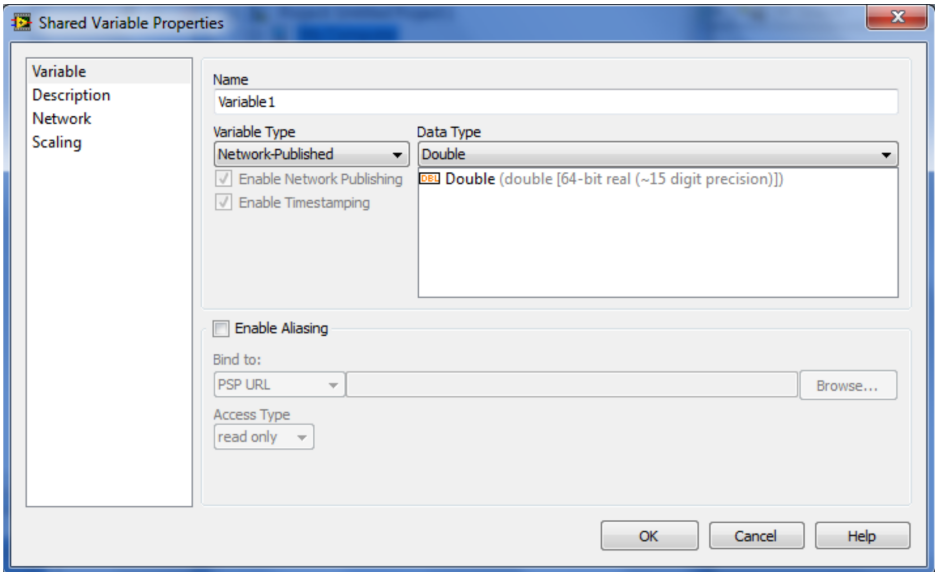
4 Zmienne współdzielone

Zmienne współdzielone służą do przekazywania danych pomiędzy różnymi VI w ramach jednego projektu lub do przekazywania danych pomiędzy różnymi urządzeniami podłączonymi do sieci.

Przy tworzeniu takiej zmiennej należy zdefiniować jej nazwę oraz wybrać typ. Zapis bądź odczyt ze zmiennej możliwy jest poprzez przeciągnięcie ikonki zmiennej z okna projektu na diagram blokowy VI. Domyślnie zmienna współdzielona jest do odczytu, aby zapisać do niej dane kliknij na nią prawym przyciskiem myszy a następnie wybierz **Access Mode** → **Write**.



Rysunek 8: Tworzenie zmiennej współdzielonej.



Rysunek 9: Okno wyboru właściwości zmiennej.

Zmienne współdzielone powinno się stosować z umiarem, ponieważ ich wykorzystanie nie narzuca kolejności wykonywania programu. Z tego powodu podczas stosowania zmiennych mogą wystąpić niepożądane zachowania programu np:

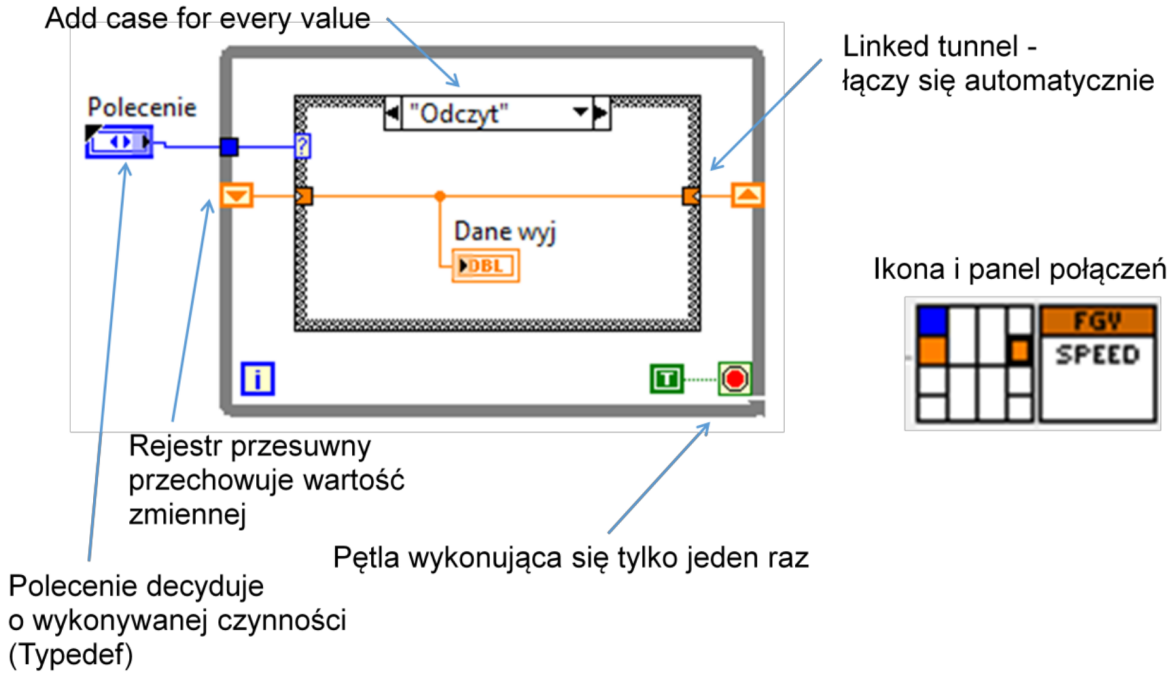
- odczytanie dwa razy tej samej wartości
- można przegapić zmianę wartości zmiennej

5 Zmienne funkcjonalne

Zmienna Funkcjonalna (Functional Global Variable (FGV)) jest to zmienna stworzona w oparciu o subVI z rejestrem przesuwnym. Wykorzystuje ona fakt, że rejestr przechowuje wartości z ostatniego uruchomienia programu. Zmienna funkcjonalna może oprócz przechowywania wartości wykonywać operacje na danych, albo zmieniać ich postać (np. zapisujemy do zmiennej wartość skalarną, odczytujemy natomiast tablicę z danymi). Nazwa zmiennej jest interpretowana jako:

- Functional - zmienna może zostać użyta do wykonania obliczeń na danych lub zmiany ich postaci,
- Global - jest dostępna globalnie tzn. może być wykorzystana w różnych subVI,
- Variable - można wykonywać zapis i odczyt danych analogicznie jak ze zwykłej zmiennej.

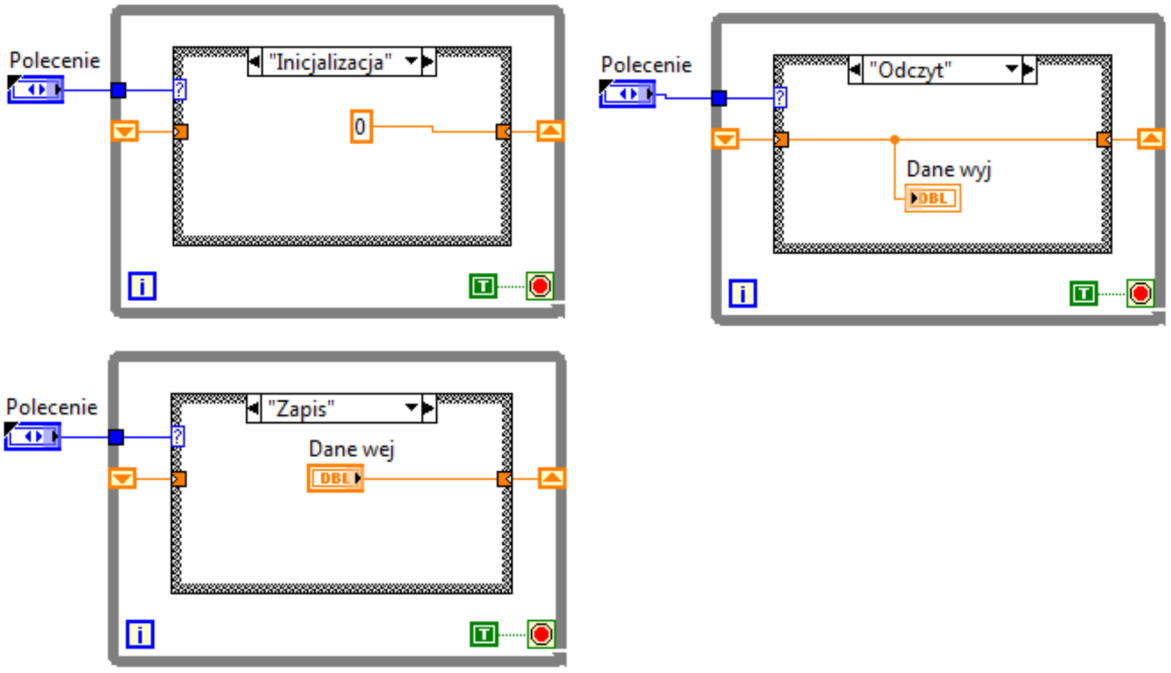
Zmienna Funkcjonalna składa się z pętli while z rejestrem przesuwnym oraz strukturą case. Pętla while ma na stałe podłączony warunek zatrzymania - w ten sposób kod w pętli uruchamia się tylko raz przy każdym uruchomieniu subVI ze zmienną. Pętla while jest użyta w celu stworzenia rejestru przesuwnego, w którym przechowywane są dane. Zmienna FGV oprócz wartości danych przyjmuje również polecenie realizowane jako definicja typu (strict type def.) dla typu wyliczeniowego (enum). Podstawowe trzy polecenia realizowane przez zmienną funkcjonalną to inicjalizacja, zapis, odczyt. W praktyce często zmienne funkcjonalne posiadają inne funkcje, takie jak analiza danych, zapis do pliku, zmiana formatu danych itp.



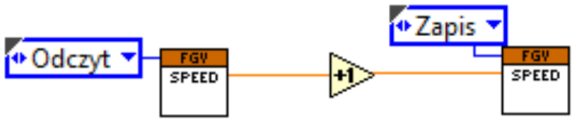
Rysunek 10: Opis zmiennej funkcjonalnej.

Zmienna FGV jest jednym z dwóch najpopularniejszych sposobów chronienia krytycznego fragmentu diagramu przed zjawiskiem *race conditions*, gdyż zmienna FGV to *non-reentrant VI*. W praktyce oznacza to, że jest tylko jedno miejsce w pamięci gdzie przechowywane są dane z subVI. Taki subVI nie może być wywoływany równolegle. Inne wątki muszą w takim przypadku "poczekać".

Przy tworzeniu zmiennej funkcjonalnej warto pamiętać o pewnych ułatwieniach w LabVIEW. Po kliknięciu na strukturę Case możliwe jest wybranie opcji "Add case for every value" co umożliwia automatyczne stworzenie wszystkich przypadków w strukturze.



Rysunek 11: Realizacja prostej zmiennej funkcjonalnej.



Rysunek 12: Użycie FGV w diagramie programu.

6 Zadania do wykonania

1. (2,5p.) Stwórz program umożliwiający zmianę wartości wykresu typu XY Graph podczas działania programu. W tym celu:
   a) Stwórz nowy subVI.
   b) Wygeneruj testowe dane dwóch przebiegów (jeden okres funkcji sinus i cosinus). Wskazówka: skorzystaj z subVI stworzonego na poprzednich laboratoriach.
   c) Stwórz kontrolki umożliwiające poprzez Property Nodes zmianę następujących właściwości wykresu (każdy pkt za 0.5 pkt.):
      – pokaż/ukryj wykres,
      – pokaz/ukryj legendę,
      – zmień nazwę wykresu 1 i 2,
      – zmień kolor wykresu 1 i 2,
      – ustaw styl linii (prosta, linia przerywana itd) wykresu 1 i 2. Pamiętaj aby zabezpieczyć program przed ewentualnie nieprawidłowo wpisaną wartością.

2. (2,0p.) Otwórz projekt "Zmienne wspoldzielone.lvproj". W projekcie znajdź i otwórz plik "Zmienne wspoldzielone 0.vi". Dodaj do projektu odpowiedni typ danych oraz zmienne współdzielone, które zapewnią przesyłanie sygnałów testowych między programami symulującymi nadajniki, a programem symulującym odbiornik. Należy zapewnić również zatrzymywanie wszystkich programów jednym wyłącznikiem. **Punktacja:** za każdy poprawnie przesłany i odebrany sygnał - 0,5p., za zatrzymanie wszystkich programów jednym przyciskiem - 0,5p.

   WSKAZÓWKA: Do przekazywania przebiegów użyj zmiennej typu waveform.

3. (3,5p.) Stwórz projekt "Zmienne funkcjonalne.lvproj". Dodaj w nim SubVi "Pomiary.vi", który będzie realizował zmienną funkcjonalną. Zmienna ta będzie przechowywała symulowane dane pomiarowe. Stwórz następujące metody/stany zmiennej funkcjonalnej:

   a) "add" - która będzie dodawała tablicę 2D nowych punktów pomiarowych. Każdy punkt ma składać się ze współrzędnej czasowej oraz wylosowanej wartości za pomocą funkcji "Random numbers" (stwórz odpowiedni typ danych do zrealizowania tego zadania), (0,5p.)

   b) "read" - która będzie zwracała tablicę 2D 5-ciu ostatnich pomiarów, (0,5p.)

   c) "readAll" - która będzie zwracała tablicę 2D wszystkich pomiarów dodanych do tej pory, (0,5p.)

   d) "reset" - która będzie kasowała wszystkie dane ze zmiennej, (0,5p.)

   e) "zapis do pliku" - która będzie zapisywała wszystkie dane do pliku, (0,5p.)

   f) "odczyt z pliku" - która będzie wczytywała dane z pliku do zmiennej FGV. (0,5p.)

   W głównym programie należy umieścić pętlę while oraz strukturę zdarzeniową umożliwiającą przetestowania zmiennej funkcjonalnej. (0,5p.)

4. (2p.) Stwórz program w którym za pomocą funkcji ⟨Simulate Sig⟩ będzie generowana funkcja sinus o amplitudzie równej 10, z częstotliwością 10 Hz. Do sygnału należy dodać szum o maksymalnej amplitudzie równej 1. W programie należy dodać kontrolkę umożliwiającą ustawienie poziomu krytycznego. Aplikacja powinna zmieniać kolor wykresu dla wartości sygnału przekraczających poziom krytyczny. **Punktacja:** za wygenerowanie i wyświetlenie funkcji sinus - 1p., za zmianę fragmentu wykresu przekraczającego poziom krytyczny - 1p.

Rysunki o numerach: 2, 8, 9, 10, 11, 12 oraz fragmenty opisu zmiennej funkcjonalnej pochodzą z pierwotnej wersji instrukcji autorstwa dr inż. Marcina Biedy.