LabView – Podstawy programowania

wg

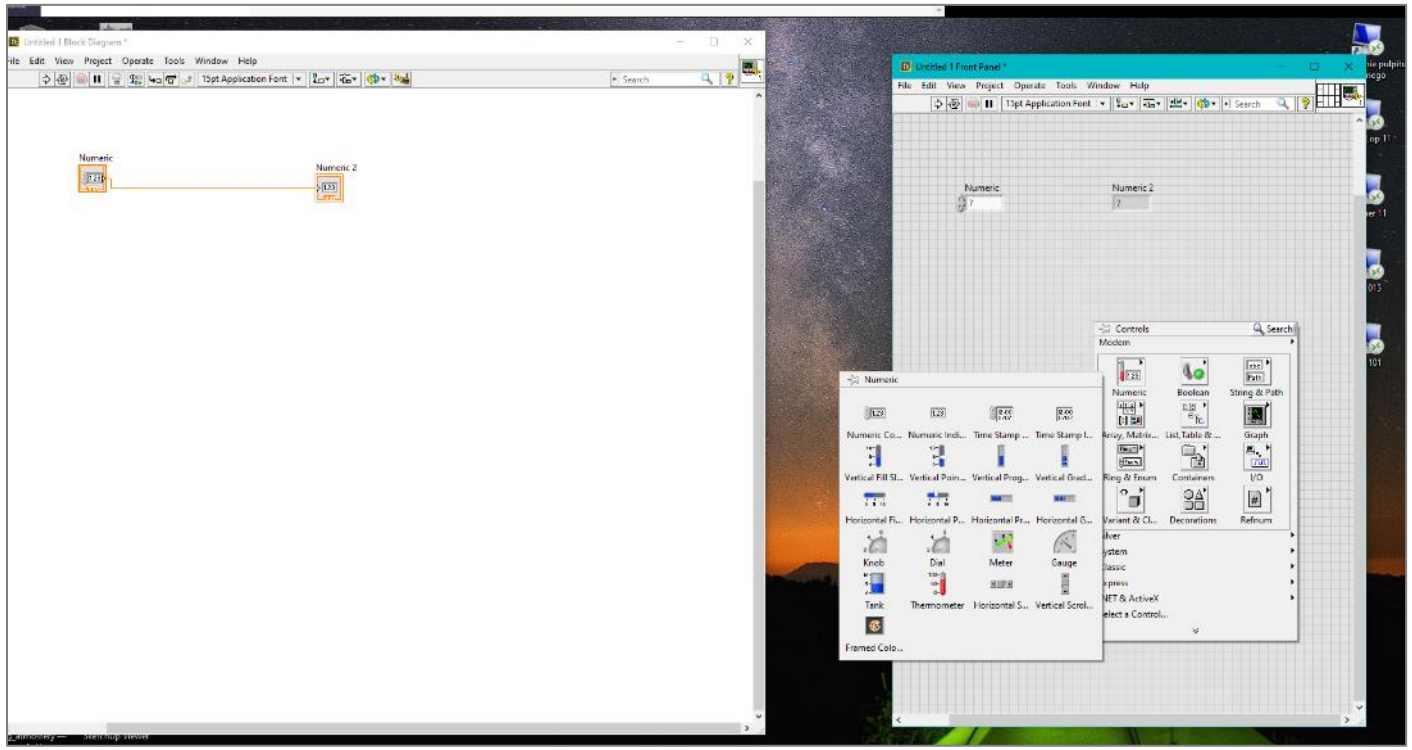
mgr inż. Konrad Gumowski

LabView (czasem nazywany językiem G) jest językiem programowania (Visual Programming Language VPL) szczególnie ułatwiającym obsługę urządzeń, zbieranie danych (akwizycję) z urządzeń zewnętrznych i prezentację wyników w formie graficznej. Początki – 1983, od 2006 forma dojrzała. Pozwala na dołączanie bibliotek (podobnie jak w Matlab). Praca polega na tworzeniu Virtualnych Instrumentów (pliki mają rozszerzenie .vi).

Plan na najbliższe zajęcia – elementy języka

Nr	Zadanie	
1	Tworzenie kontrolek wejścia / wyjścia	Praca ze typami: int oraz double
2	Tworzenie relacji między terminalami	Dodanie typu bool
3	Obsługa projektu i błędów, porządkowanie, kompilacja kodu	
4	Wprowadzenie struktury warunkowej IF	Case strukture
5	Wprowadzenie typu char	Obsługa nazw i tektu
6	Wprowadzenie pętli while	Pętla i shift register
7	Wprowadzenie tablic	
8	Obsługa czasu i generatora liczb losowych	Wait until next ms
9	Wprowadzenie pętli for	Statystyka
10	Własne funkcje	Sub vi
11	Reprezentacja graficzna elementów tablic	Graph, XY Graph
12	Operacje matematyczne	Fitting
13	Obsługa zdarzeń	Event structure
14	Zmienne oraz wyciąganie wartości kontrolek	Property Node
15	Obsługa plików	

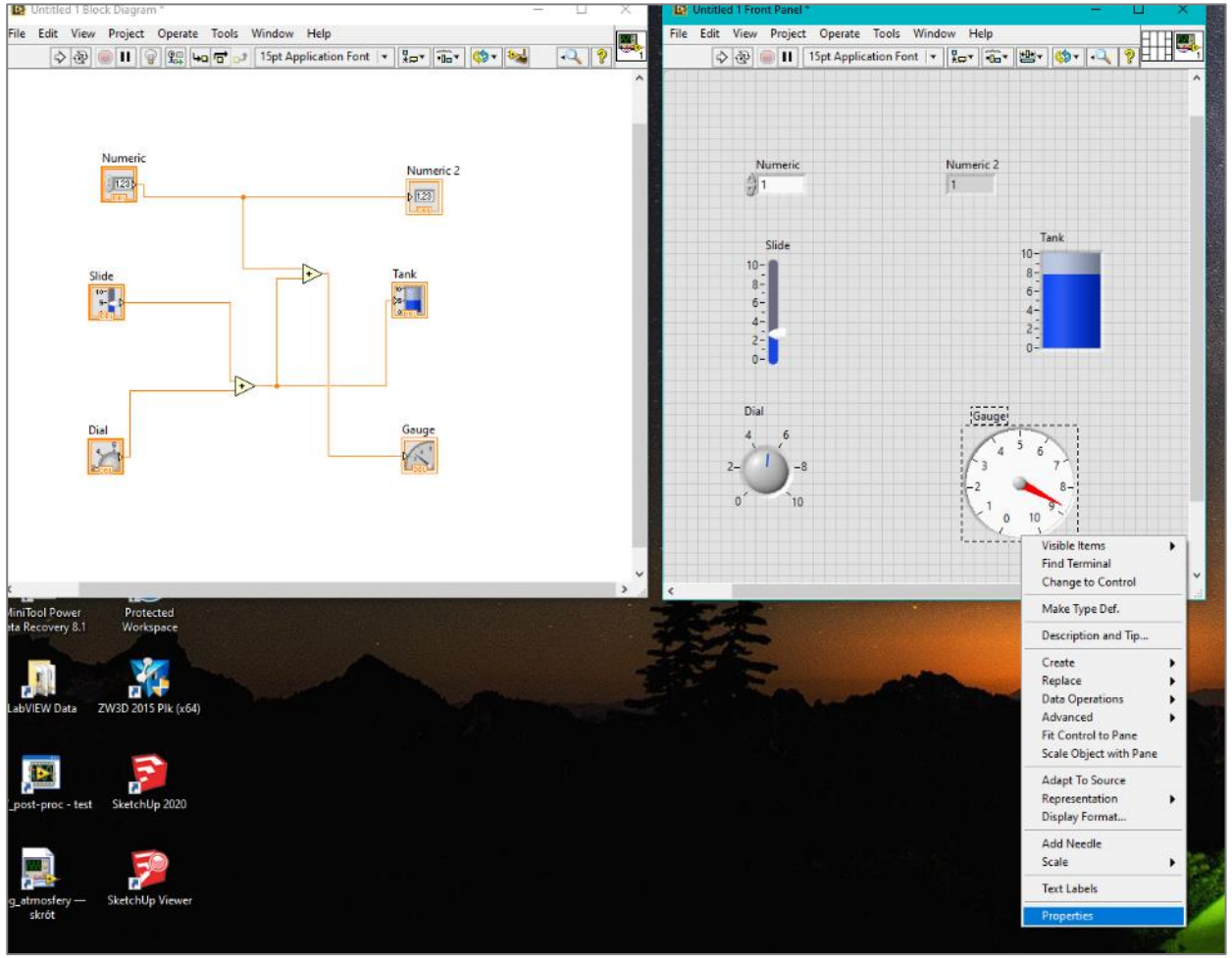
1 Tworzenie kontrolek We / Wy



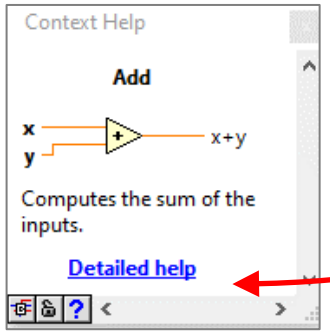
Na Front Panelu klikając prawym przyciskiem myszy przywołujesz rozwijaną listę menu. Z grupy Numeric wybierz i wstaw (na front panelu) elementy Numeric Control i Numeric Indicator, zauważ że każdy z tych elementów ma swój terminal na Block Diagramie.

Podstawowa zasada każdego bloczka w Blok Diagramie: przy jego prawej krawędzi są wyjścia zaś przy lewej krawędzi są jego wejścia. Klikając lewym przyciskiem myszy na wyjściu i dalej na wejściu utworzysz połączenie między nimi.

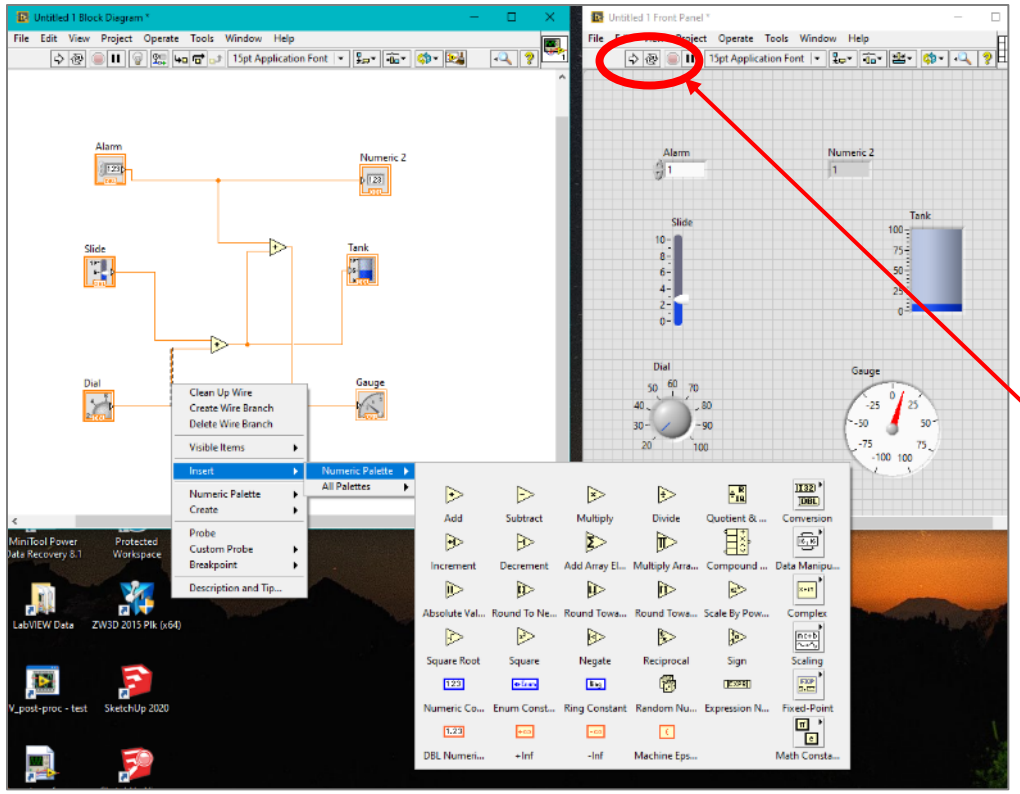
1 Tworzenie kontrolek We / Wy



W Block Diagramie z grupy Neumeric wybierz podstawowe operacje. Połącz niteczki między terminalami przez operacje do indykatorów. Możesz je zaznaczać i usuwać przyciskiem delete na kalwiarurze.
Na każdym elemencie tj. terminalu i kontrolce (indikatorze) klikając prawym przyciskiem możesz przywołać jego właściwości. Zmień skale zegarowi od -100 do 100.
Przydatny jest Context Help uruchomisz go skrótem klawiszowym Ctrl+H. Po najechaniu na bloczek zawartość okna pomocy się zmienia prezentując krótki opis danego bloczku, rozszerzony opis dostępny jest po wciśnięciu.



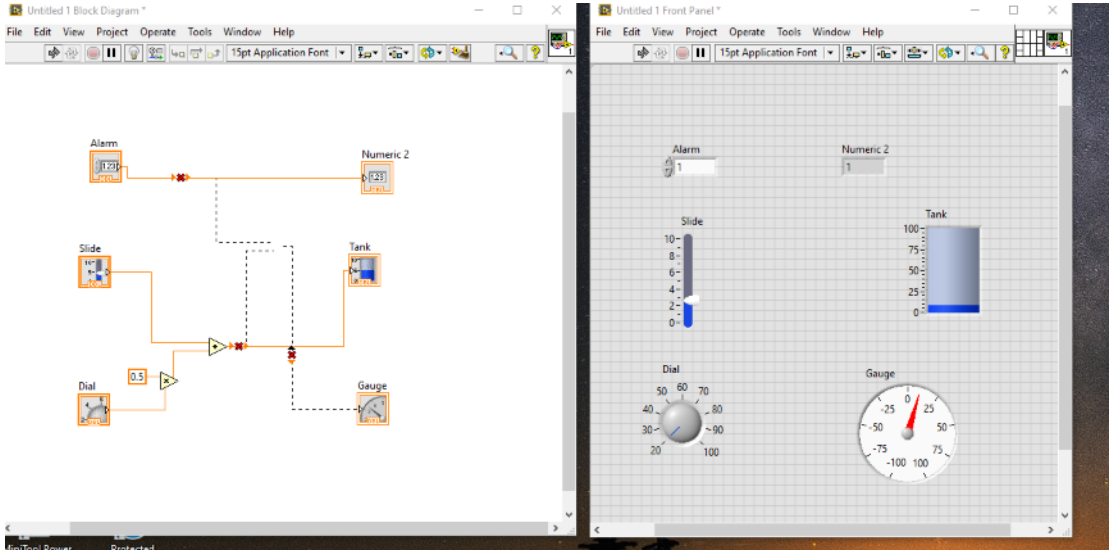
2 Tworzenie relacji między terminalami



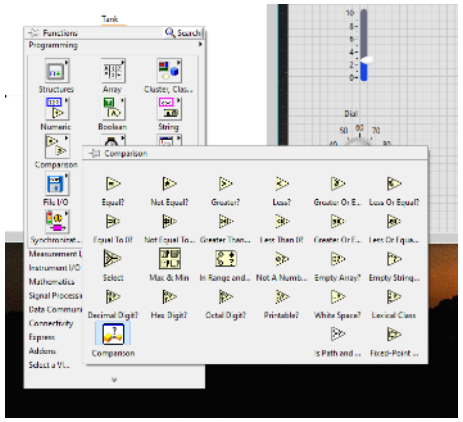
W schemacie na Block Diagramie możesz dodawać operacje przez menu przywoływane prawym przyciskiem myszy na białym tle. Możesz także kliknąć prawym przyciskiem myszy na nitce i wstawić w niej dowolną operacje.

Wykonaj kod wg. schematu i przetestuj
uruchom wciskając Run Continuously,
aby zatrzymać wciśnij Abort Execution.
Przyciski te są zdwojone i równoważne
w górnym menu front panelu jak i block diagramu.

2 Tworzenie relacji między terminalami

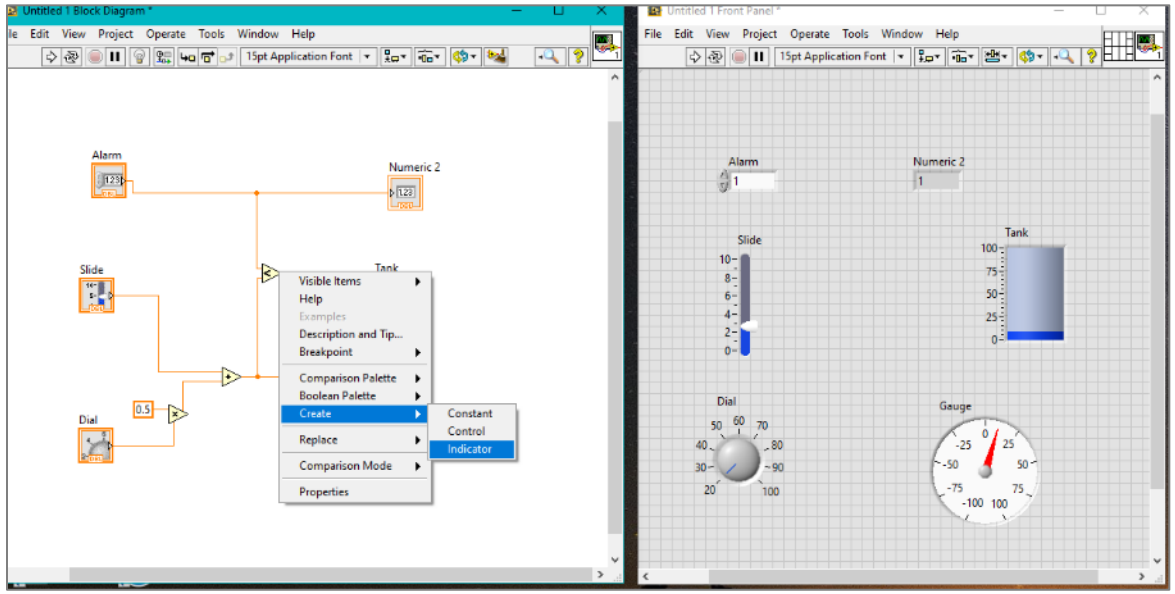


Gdy zaznaczysz dowolny element możesz go usunąć klikając Delete na klawiaturze.
Przydatnym skrótem klawiszowym będzie Ctrl+B

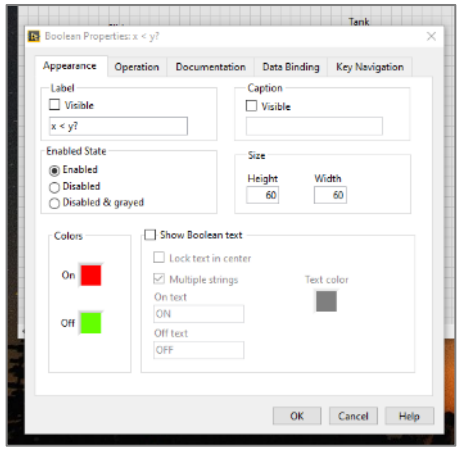


Sprawdź grupę Comparision wybierz „Less?”

2 Tworzenie relacji między kontrolkami

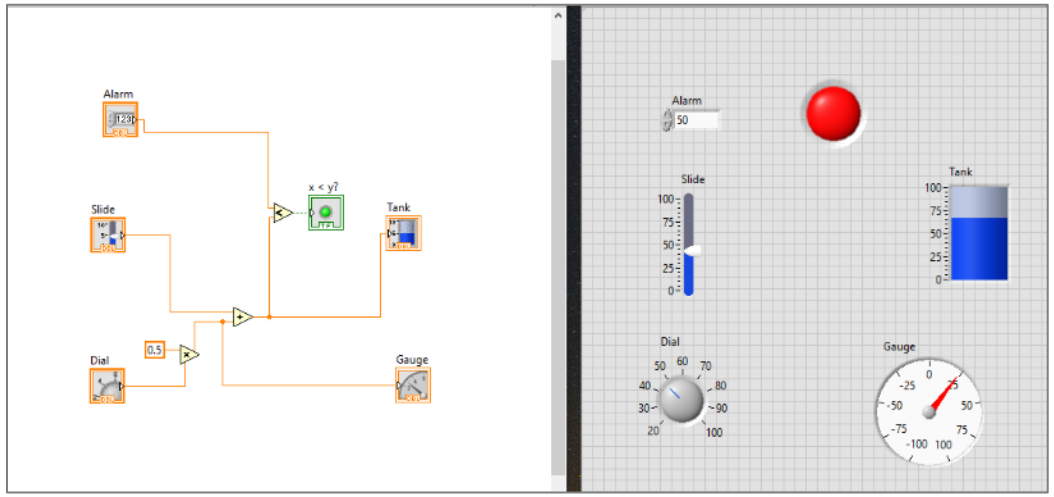


Umieść go w schemacie i precyzyjnie najedz myszką na jego wyjście(ten ogonek po jego prawej) prawym przyciskiem myszy przywołaj menu i utwórz indicator.
Pojawi się standardowy indicator pamiętaj że możesz go podedytować.



Wykonaj kod wg. schematu i przetestuj uruchom Run Continuously, aby zatrzymać wciśnij Abort Execution,

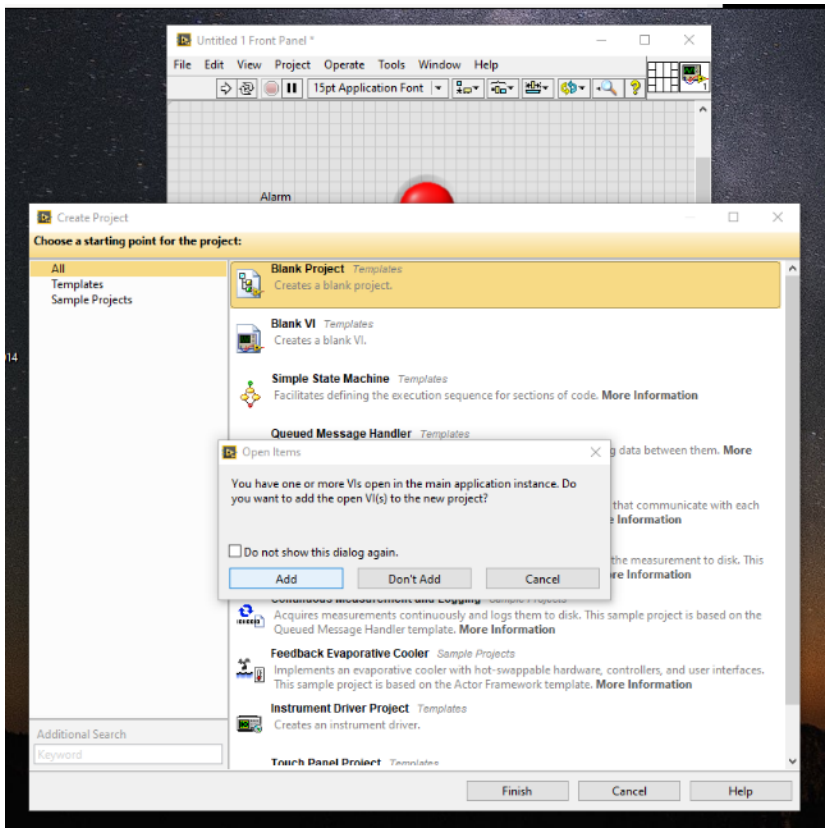
3 Obsługa projektu – kompilacja programu



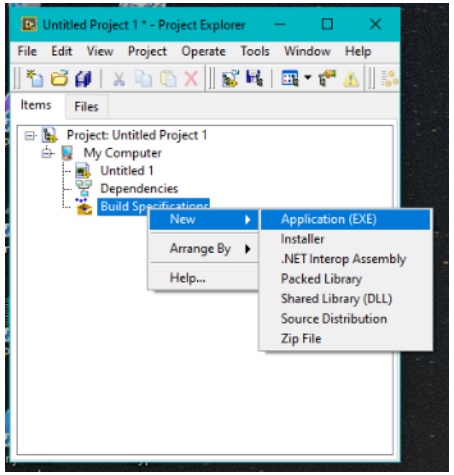
Ustaw odpowiednie zakresy i przetestuj.
Blok Diagram możesz bezkarnie zamknąć lub go przywołać skrótem Ctrl+E.

Dobrą praktyką jest nadanie wartości początkowej co jest odpowiednikiem inicjalizacji. Tutaj nadajemy ją w sposób niejawny przez wpisanie wartości np. w kontrolce Alarm wartości 50 (na Front Panelu) a następnie klikając na niej prawym klawiszem myszy i w grupie Data Operations -> Make Current Valule Default.

3 Obsługa projektu – kompilacja programu

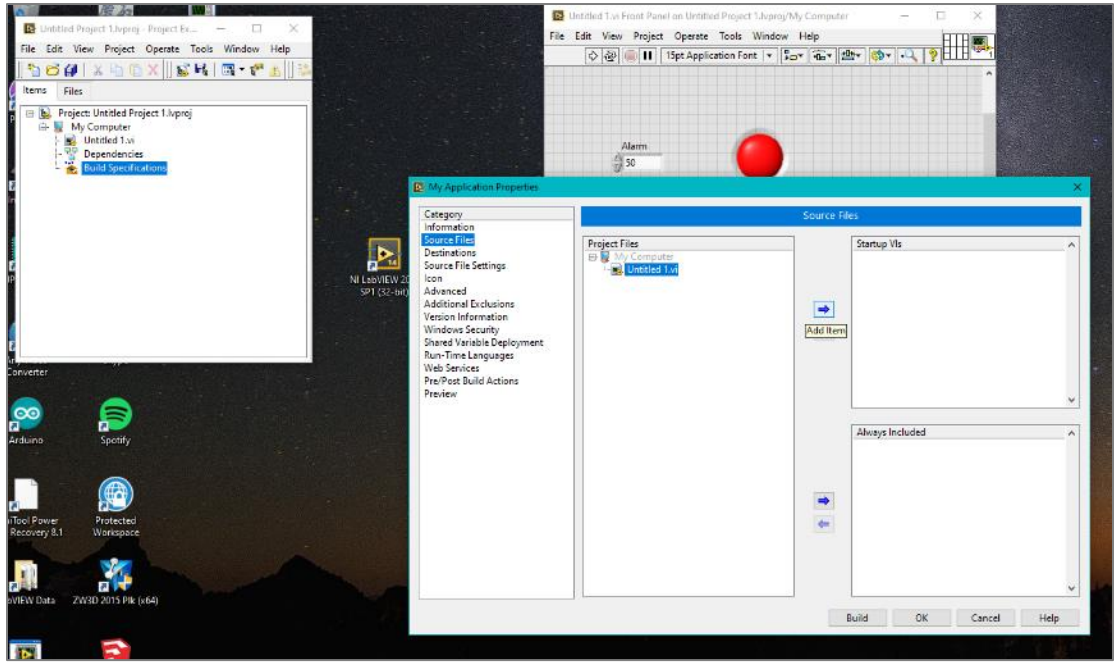


W menu Front Panelu wybierz Create Project i wybierz pierwszy z góry Blank Project gdzie dodaj utworzony program do projektu.
W grupie Build Specification prawym przyciskiem myszy wybierz New -> Aplication (EXE).



Zostaniesz poproszony o zapisanie na dysku pliku projektu i pliku z kodem.

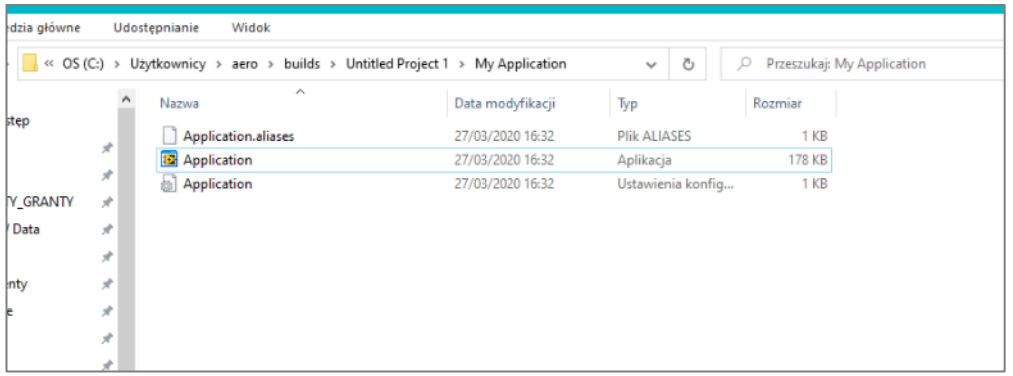
3 Obsługa projektu – kompilacja programu



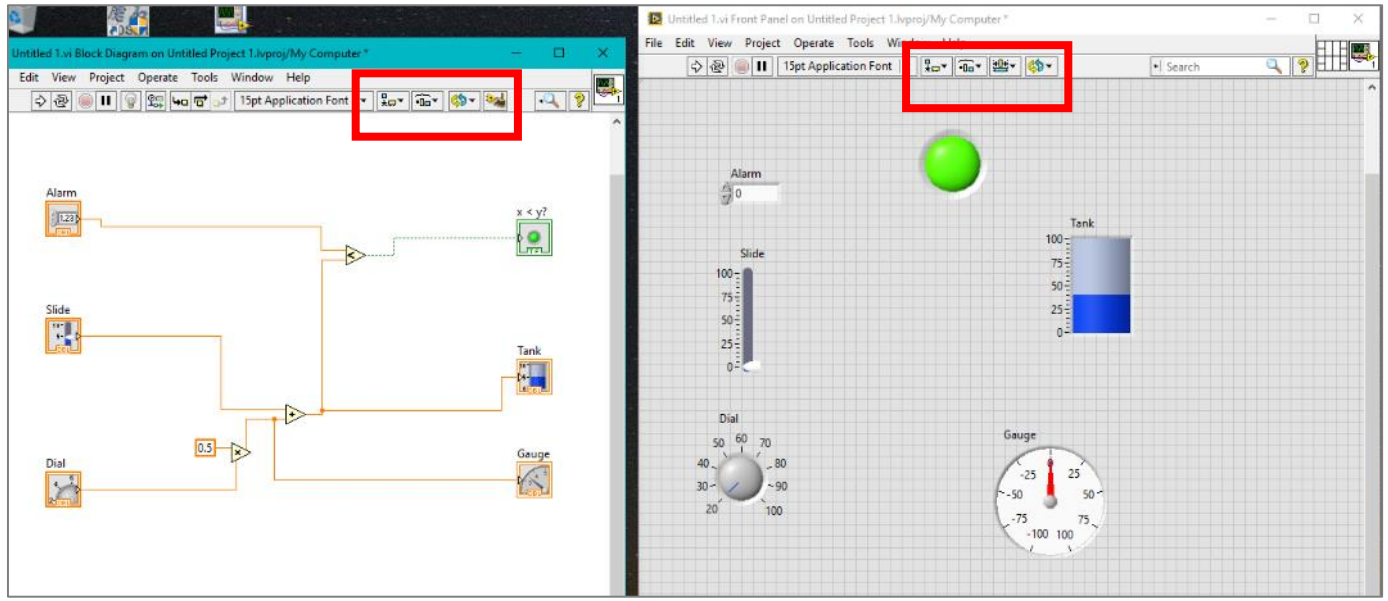
Przy konfiguracji aplikacji musisz jedynie w grupie Source Files dodać swój (jak dotąd jedyny) plik do grupy Startup Vis i OK.

Już tylko klikając prawym przyciskiem myszy na Build Specification wybierz Build All i rozpocznie się proces kompilacji.

Na dysku powstanie katalog ze skompilowanym programem.



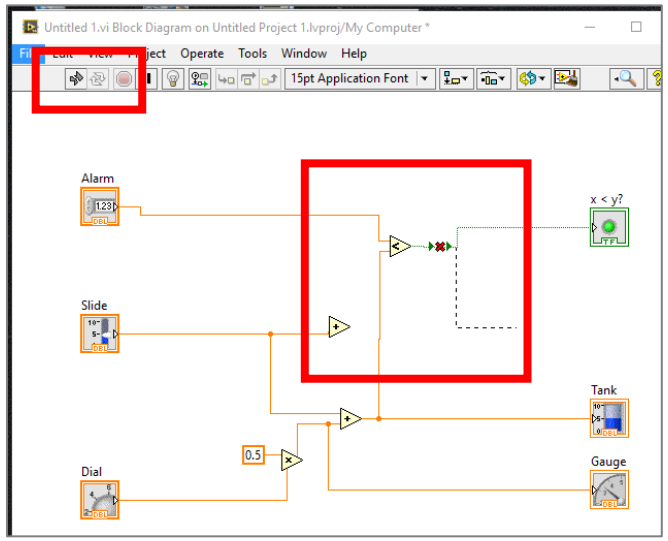
3 Porządkowanie kodu



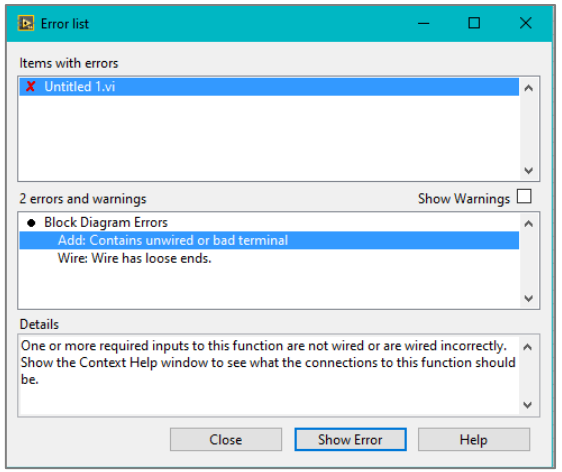
Elementy kodu możesz łatwo przesuwać co poprawia przejrzystość programu. Zapoznaj się z ikonami zaznaczonymi w czerwonych ramkach ułatwią one zarządzanie pozycją ikonek szeregując, wyrównując pozycje, grupując i blokując pozycję.
W przypadku nakładania się na siebie elementów (głównie na Front Panelu) pozwalają na zarządzanie.

Pamiętaj że nitki możesz zaznaczać, kasować Delete, rysować nowe,
a nieciągłe nitki możesz usunąć Ctrl+B

3 Obsługa błędów



Podobnie jak w innych językach programowania błędny kod nie zostanie uruchomiony. O błędach jeszcze przed kompilacją informuje szara i złamana ikona Run. Mimo to wciśnij ją, pokarze się okno listy błędów.

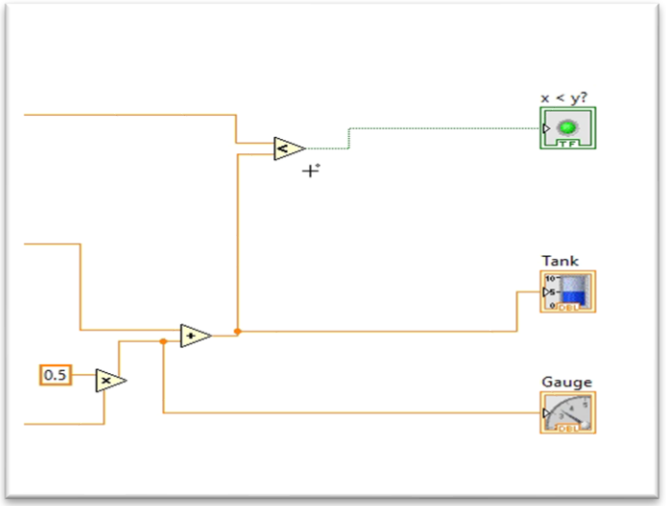


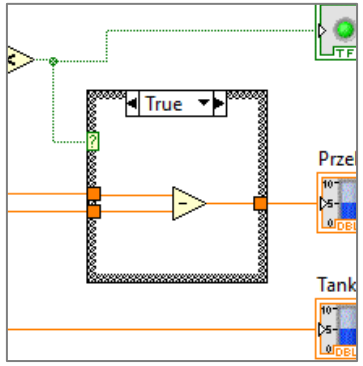
Pewne błędy są do przewidzenia np. niepodpięta nitka (której pozbędziesz się Ctrl+B).

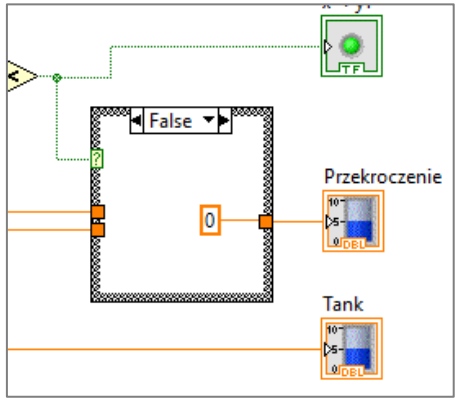
Pierwszy z błędów dotyczy niepodpiętego wejścia do bloczka i w tym przypadku to wejście jest wymagane. W tym kodzie suma objęta w czerwonej ramce jest niepotrzebna i należy ją usunąć.

Podobnie było by gdyby został jakiś bloczek np. sumy gdzieś z boku niepodłączony do niczego ale wstawiony i zapomniany.

4 Wprowadzenie struktury warunkowej IF







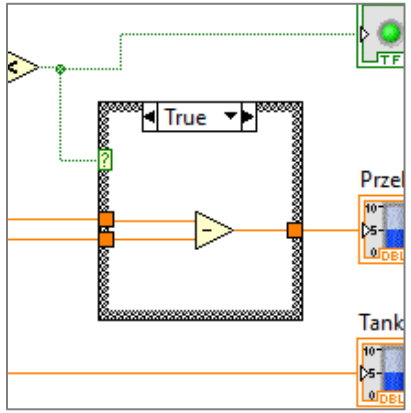
Umieść w Block Diagramie strukturę typu Case Structure. Możesz utworzyć dwa warianty kodu na kartach odpowiadających True oraz False. Przydatną opcją jest kopiowanie elementu przez jego przeciągnięcie (lewym klawiszem myszy) z wciśniętym Ctrl.

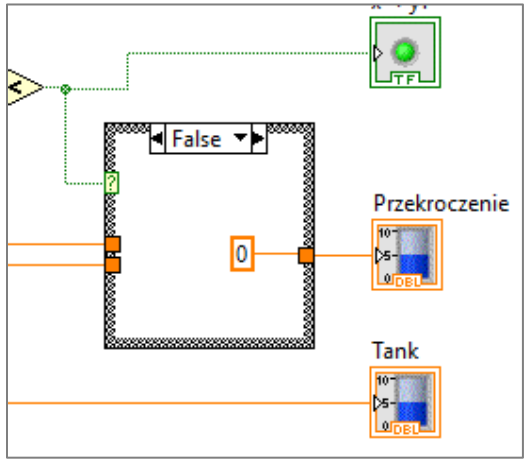
Taki kod w języku C zapisalibyśmy:

```
if(Alarm <Tank)
        Przekroczenie=0;
else
        Przekroczenie=Tank-Alarm;
```

Wykonaj kod wg. schematu i przetestuj uruchom Run Continuously, aby zatrzymać wciśnij Abort Execution.

4 Wprowadzenie struktury warunkowej IF



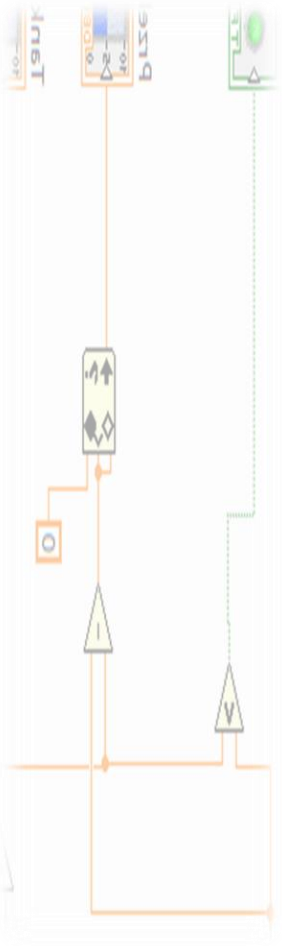


W tym przypadku zawsze jest „nierówność ostra” bo nasza struktura Case jest sterowana „nitką” typu *bool* i są tylko dwa możliwe stany.

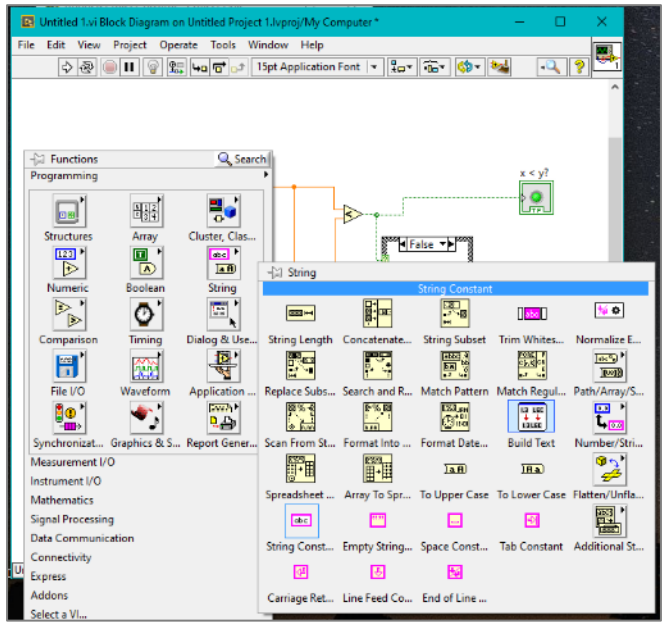
Powstały dwa wejścia do struktury Case i jedno wyjście. Wyjście w każdej sytuacji musi być obsłużone, ale wejścia mogą być niewykorzystane.

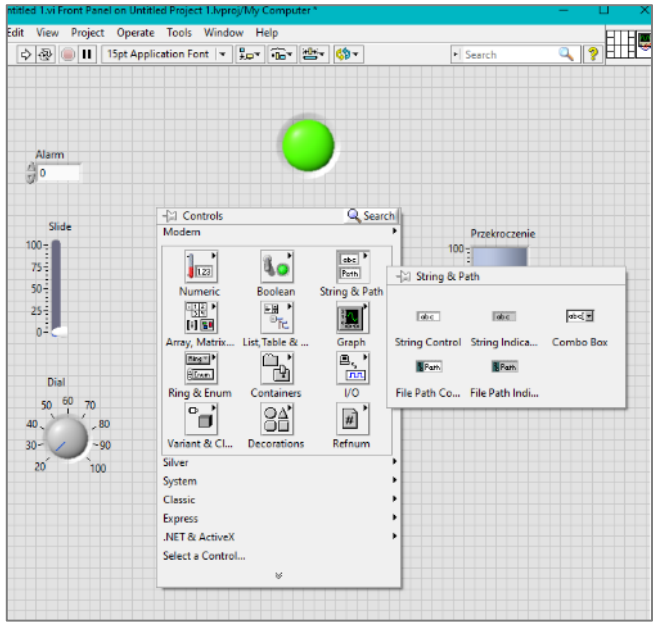
Tak proste zadanie można byłoby zrealizować jednym bloczkiem np. In Range and Coerce z grupy Comparision.

Przetestuj wspomniany bloczek  .

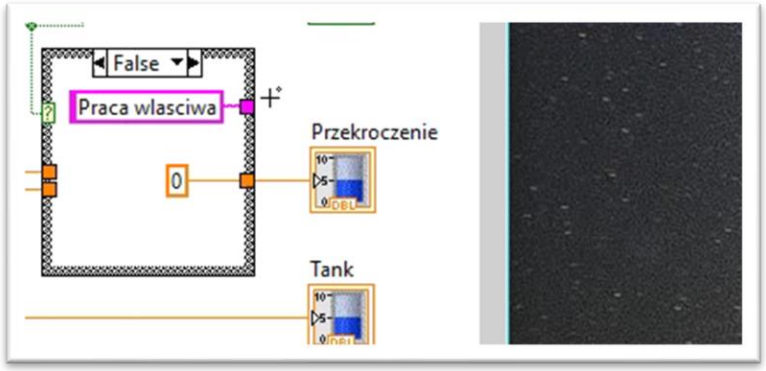


5 Wprowadzenie typu char



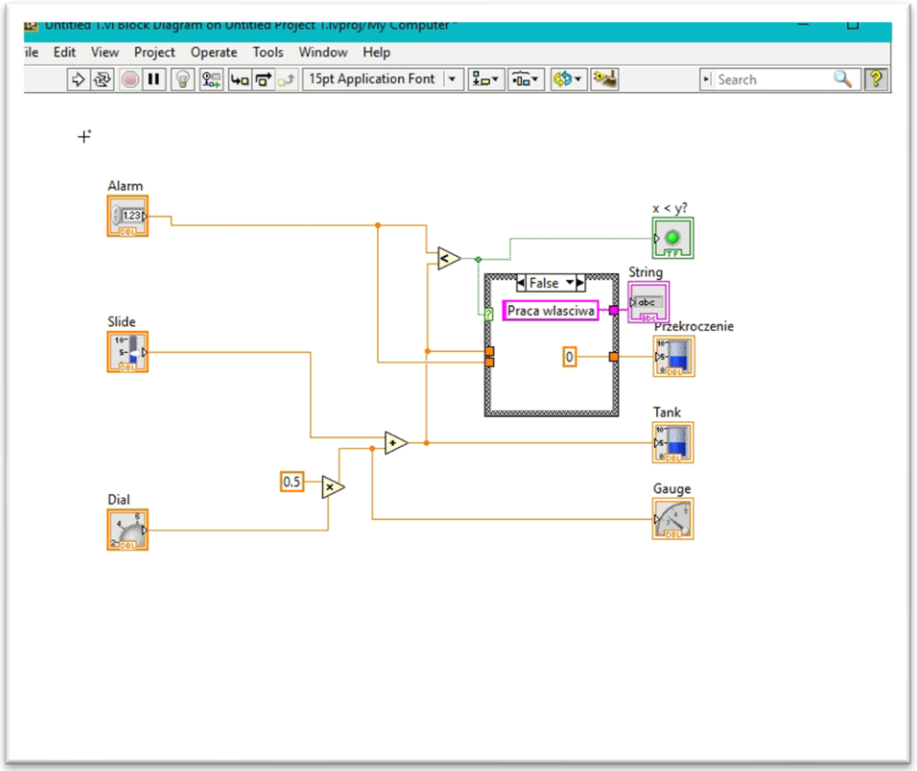


Zmienne tekstowe dostępne są jako tzw. *String* i z poziomu Front panelu mamy możliwość bezpośredniego wprowadzania i wyświetlania zaś z poziomu Block Diagramu mamy w grupie String kilka funkcji. Ich nazwy sugerują znaczenie, ale pomocne są podpowiedzi w oknie Contekst Help pojawiające się po najechaniu na konkretny bloczek.

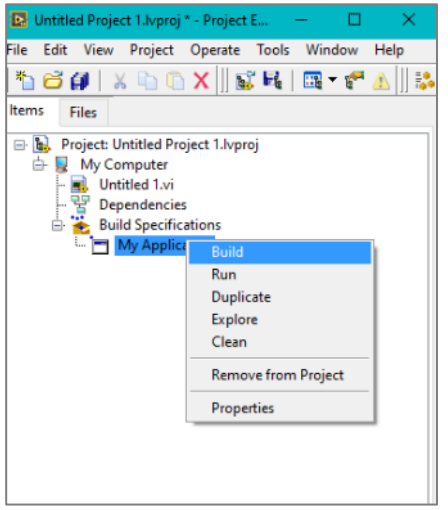


Wstaw w strukturze Case elementy String Constant zawierające „Praca wlasciwa” i „Przekroczenie”.
A następnie klikając prawym przyciskiem myszki na zewnętrzny „ogonek” stwórz Create Indicator.
Wykonaj kod wg. schematu i przetestuj uruchom Run Continuously, aby zatrzymać wciśnij Abort Execution.

6 Wprowadzenie pętli While Loop

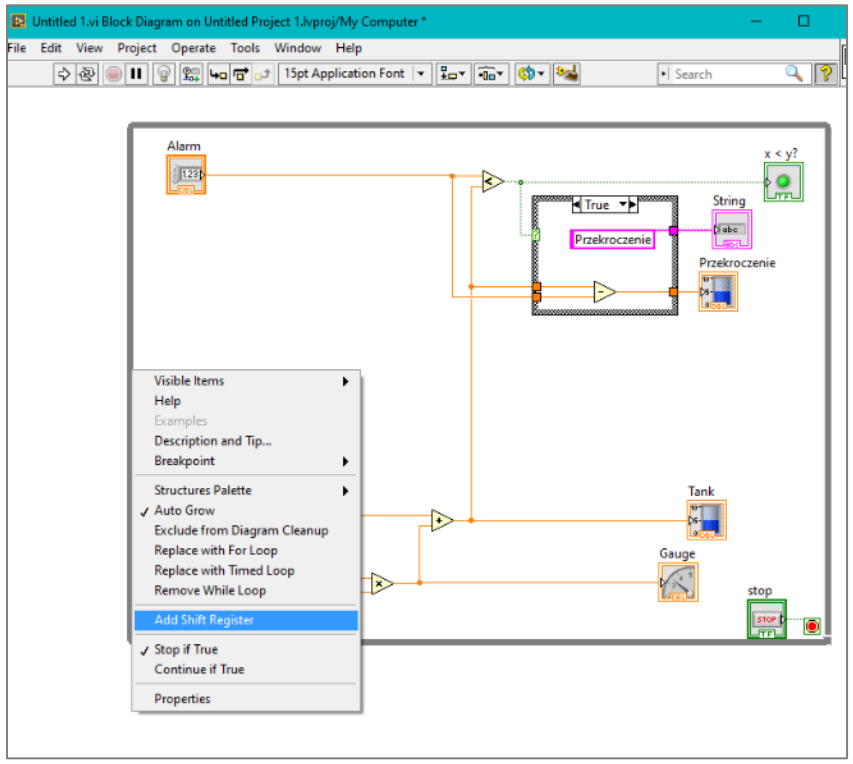


Każda szanująca się aplikacja posiada pętlę główną taką która zapewnia ciągłą pracę. Dotychczas ciągłą pracę zapewniał tryb uruchamiania Run Continously. Gdy nasz kod będzie w stanie pracować ciągle tj. będzie się wykonywał w pętli będziemy go uruchamiali przez Run a po skompilowaniu będzie pracował natychmiast po odpaleniu programu.
Pętlą główną otocz cały dotychczasowy kod i zrób jej „wyłącznik” (Create Control). Po pojawieniu się terminala kliknij na nim dwukrotnie lewym klawiszem myszy to Front Panel podpowie gdzie się znajduje jego odpowiednik
czyli kontrolka.



W oknie Project Explorera:
- skompiluj tj. Build,
- uruchom program tj. Run.

6 Wprowadzenie pętli - Shift Register

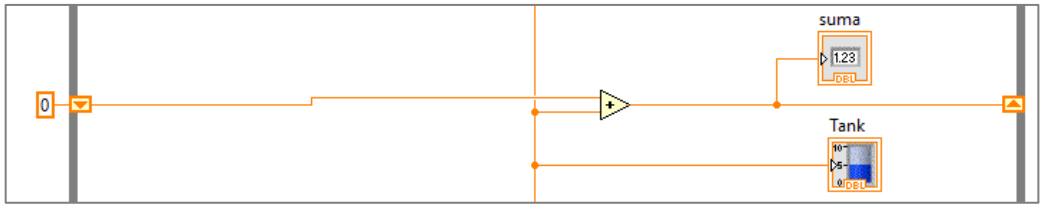


Shift register jest często wykorzystywany w obsłudze pętli.
Pozwala przekazać z wyjścia pętli do wejścia przy kolejnej iteracji pętli.
Przykładem użycia może być sumowanie:

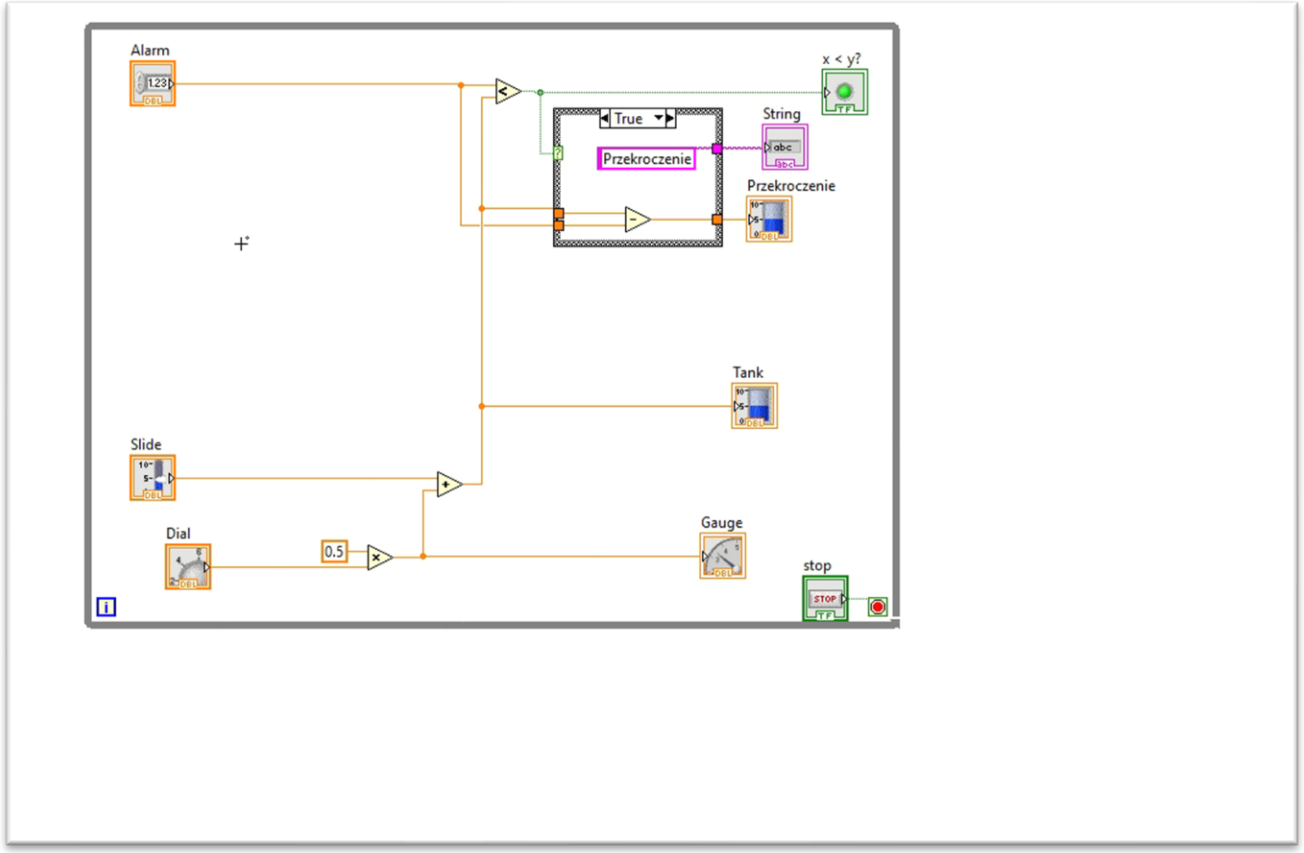
```
suma = 0;
{
        Tank = .....
        suma = suma + Tank;

}while(stop != 0)
```

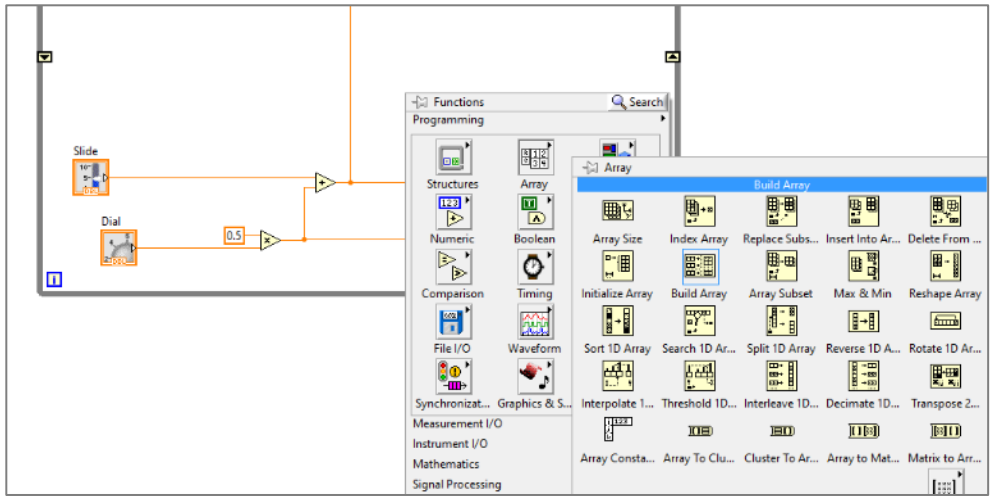
Odpowiednikiem w LabView z wykorzystaniem Shift Registrem:

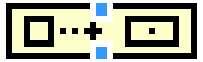


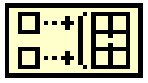
6 Wprowadzenie pętli - Shift Register

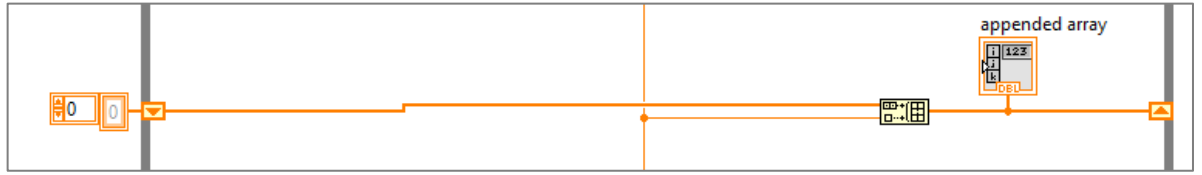


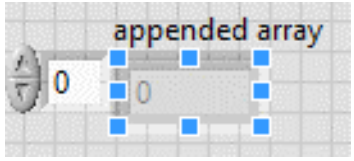
7 Wprowadzenie tablic



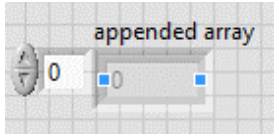
Poprzedni przykład użycia Shift register'a zastąpimy budowanie tablicy. Bloczki do tablic znajdziesz w grupie Array, wybierz Build Array. Wstaw w dowolnym miejscu bloczek. 

Najedz precyzyjnie myszką i złap lewym guzkiem myszy za niebieski punkcik i rozciągnij w dół o jeden. W ten sposób mamy dwa wejścia i jedno wyjście z tego bloczka. 
Podłącz podobnie jak przy sumowaniu.





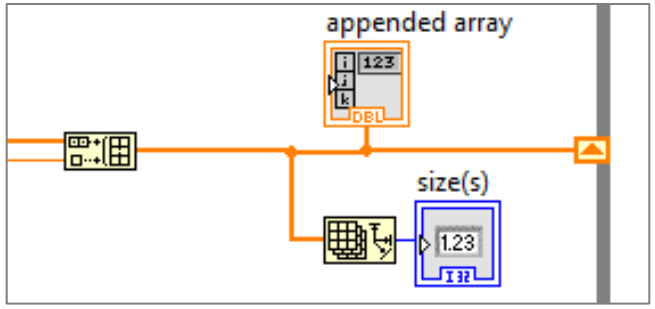
Na Front panelu tablicę możesz „rozciągnąć" w dół łapiąc za dowolny środkowy punkt zewnętrznej ramki. Jeśli klikniesz na niej prawym przyciskiem przywołasz menu gdzie z grupy Visible Items możesz włączyć wyświetlanie między innymi Vertical Scrollbar.



Indicator pojedynczego elementu możesz rozciągnąć łapiąc za punkt wewnętrznej ramki.

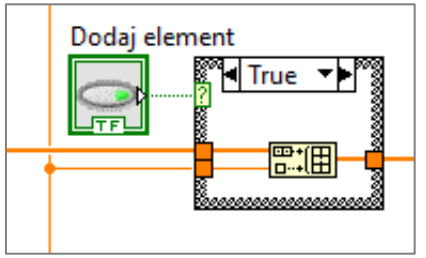
7 Wprowadzenie tablic

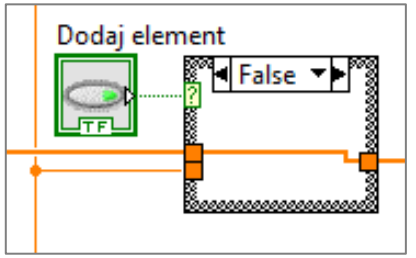
Tablica budowana w taki sposób szybko się zapełnia to dlatego że pętla działa „bardzo szybko” i w każdej iteracji pętli tablica jest zwiększana o kolejny element. Rozmiar tablicy możesz sprawdzać wykorzystując bloczek



Array -> Array Size a na jego prawym ogonku Create Indicator. Uruchom ten kod i sprawdź jak szybko rośnie rozmiar tablicy. Przydatnym skrótem klawiszowym jest Ctrl+R czyli Run.

Otocz bloczek Build Array Case Stucture i dodaj element sterujący (Create Control na zielonym ogonku z lewej).

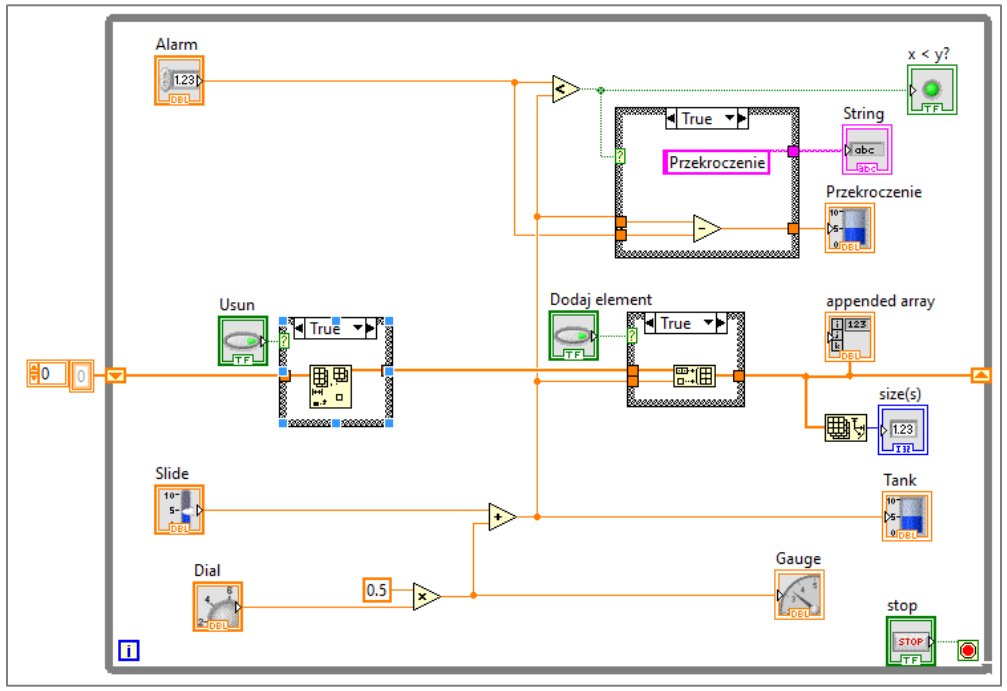




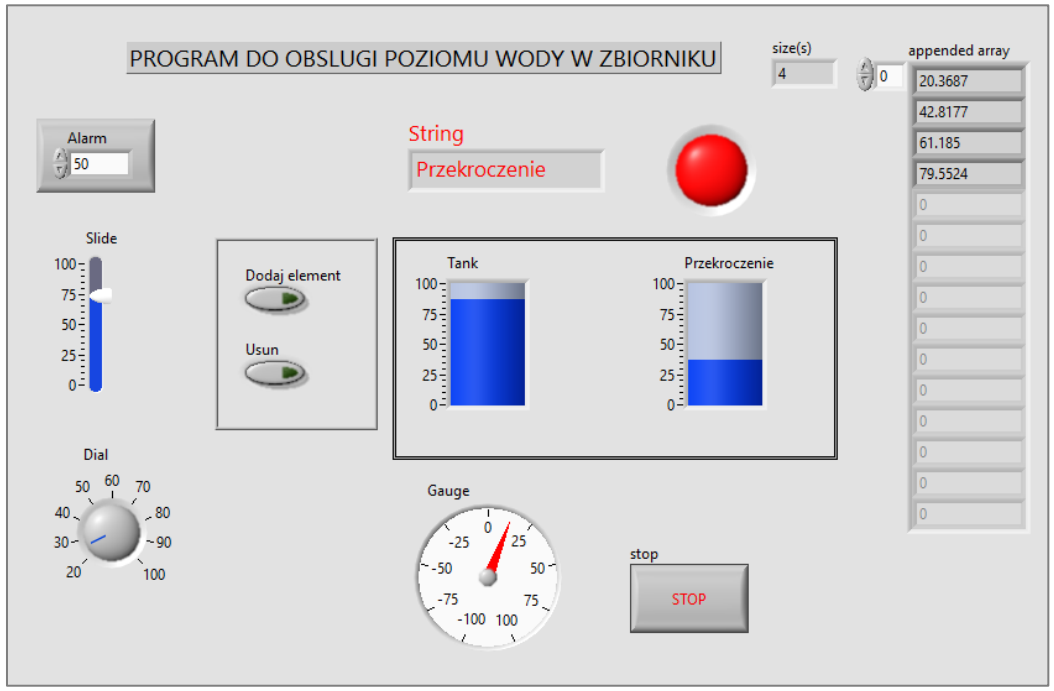
Na Front Panelu powstanie kontrolka, musisz zmienić jej sposób działania. Kliknij na niej prawym i wybierz Mechanical Action -> Latch When Releaset (środkowa na dole), piktogramy (ikonki) obrazują sposób działania. W ten sposób tablica jest zwiększana jedynie po wciśnięciu przycisku. A jego stan zmienia się na wysoki jedynie w jednej iteracji wracając. Trzymanie wciśniętego przycisku nie wpływa na działanie tego kodu.
Przetestuj działanie kodu.

7 Wprowadzenie tablic

W podobny sposób dodaj jeszcze usuwanie elementu z tablicy Array - > Delete From Array i umieść go w schemacie jak poniżej.

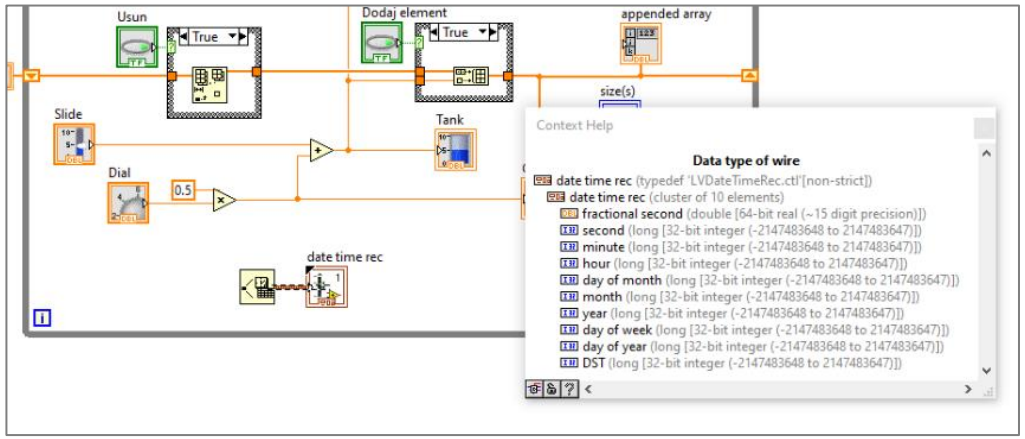


Na front panelu zrób porządki, poustawiaj po swojemu. Możesz dodać kilka elementów z grupy Decorations. Jak zaznaczysz jakiś element lub napis możesz zmienić mu czcionkę z menu powyżej (przydatny skrót to Ctrl++ na zwiększanie czcionki).



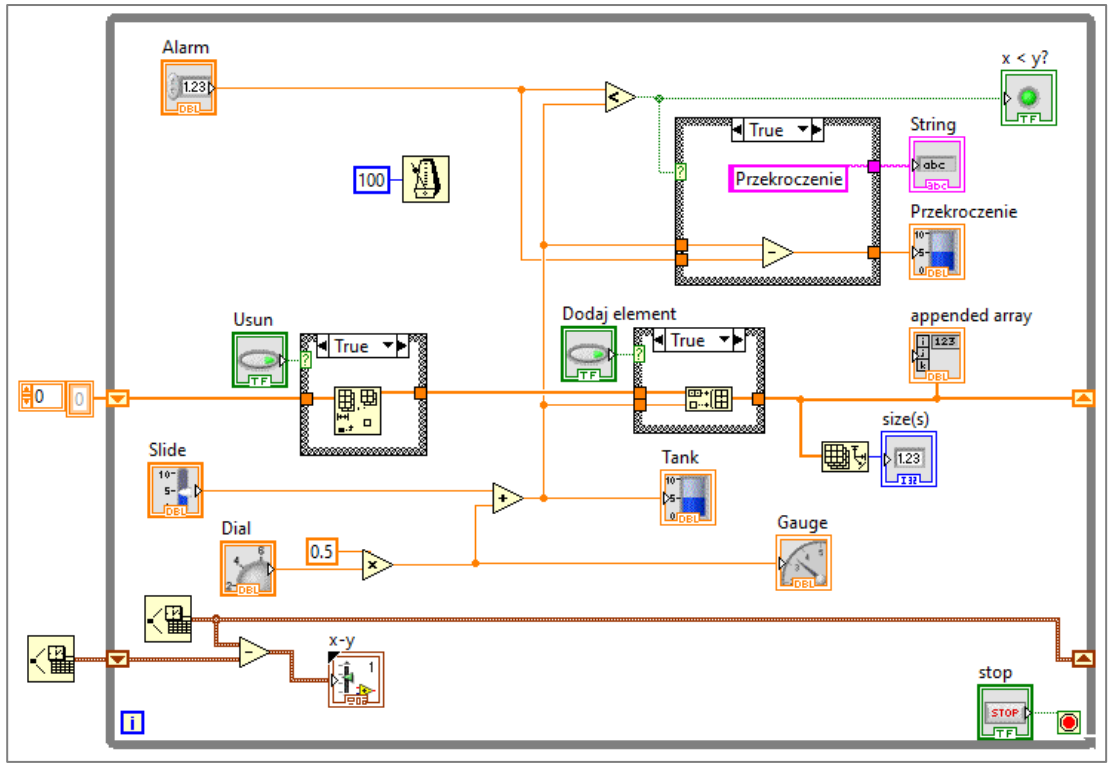
8 Obsługa czasu i generatora liczb losowych

Najczęstsze przypadki kiedy korzystamy z funkcji obsługujących czas to gdy potrzebujemy informację o czasie i aktualnej dacie lub gdy musimy opóźnić działanie programu. Wstaw w dowolnym miejscu schematu wewnątrz pętli bloczek z grupy Time o nazwie Date/Time to Seconds i po jego prawej Create Indicator. Przetestuj działanie i zapoznaj się z prezentowanymi informacjami o czasie. Są one zgrupowane w strukturze (klastrze) danych. Najedz myszką na nitkę i zwróć uwagę że w oknie Contects Help (przywoływanego przez Ctrl+h) masz informacje o elementach tego klastra (zawartości nitki).

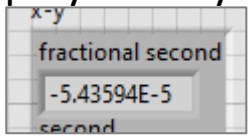


Zwróć uwagę na typy zmiennych gdzie podobnie jak w innych językach do zmiennych numerycznych są typy: double oraz integer (integer podwójnej precyzji to long). Podobne oznaczenia występują także przy dolnej krawędzi każdego terminalka   . Możemy w nie ingerować klikając prawym przyciskiem myszy na terminalu i wybierając Representation. Na razie nie będziemy ingerować. Przetestuj działanie programu.

8 Obsługa czasu i generatora liczb losowych

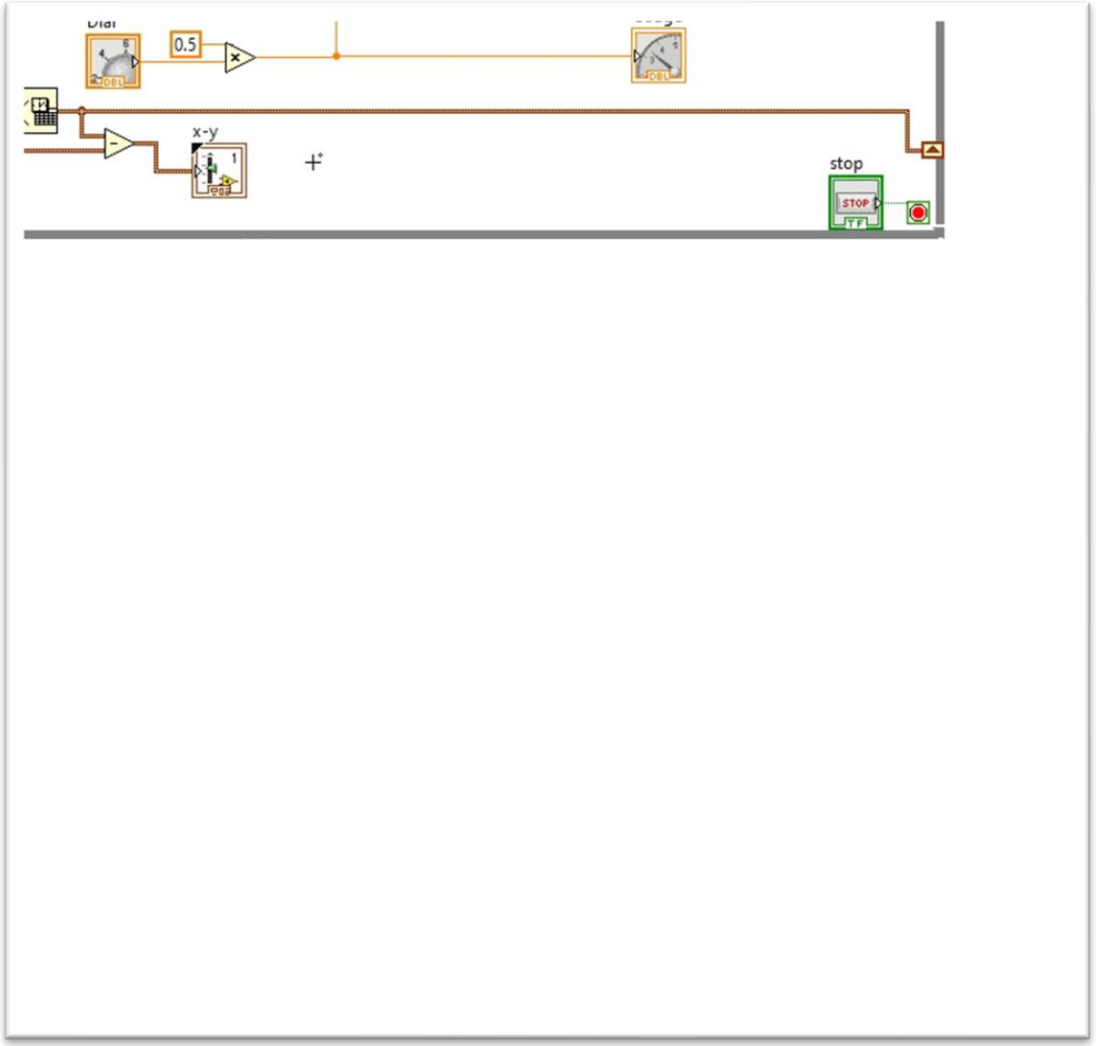


Ta druga sytuacja występuje gdy musimy odczekać żeby użytkownik nadążył, albo żeby komputer się nie przemęczał kiedy i tak nie robi nic specjalnie ważnego (niema sensu odświeżać wykresu 1000000 razy na sekundę gdy wystarczy np. 100 razy.
Wstaw w dowolnym miejscu schematu wewnątrz pętli bloczek z grupy Time o nazwie Wait Until Next ms Multiple i podłącz do jego wejścia (po lewej) stałą. Możesz sprawdzić użycie procesora swojego komputera gdy podana wartość będzie wynosiła 0 i gdy ustawisz na np. 10. Powinieneś zauważyć różnice.

Często warto rozciągnąć „ramkę wewnętrzną” indykatora podobnie jak to było przy tablicy żeby ustrzec się przed sytuacją gdy napis nie mieści się w okienku a program chce prezentować np. 

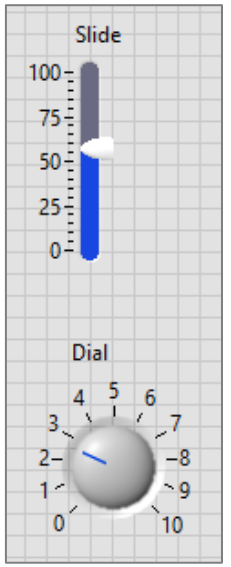
.
Możesz nad tym zapanować klikając nań prawym i w Propertysach w zakładce Display Format zmieniając z Automatic Formating na Floating point.

8 Obsługa czasu i generatora liczb losowych



Generator liczb losowych jest jak zawsze generatorem liczb pseudolosowych i daje nam wynik w postaci liczby rzeczywistej z zakresu od 0 lecz mniej niż 1. Możesz wprowadzić go do kodu w Blok Diagramie jako bloczek z grupy Numeric -> Rundom Number. W naszym przykładzie możemy zastąpić stałą 0.5 przez liczbę losową.

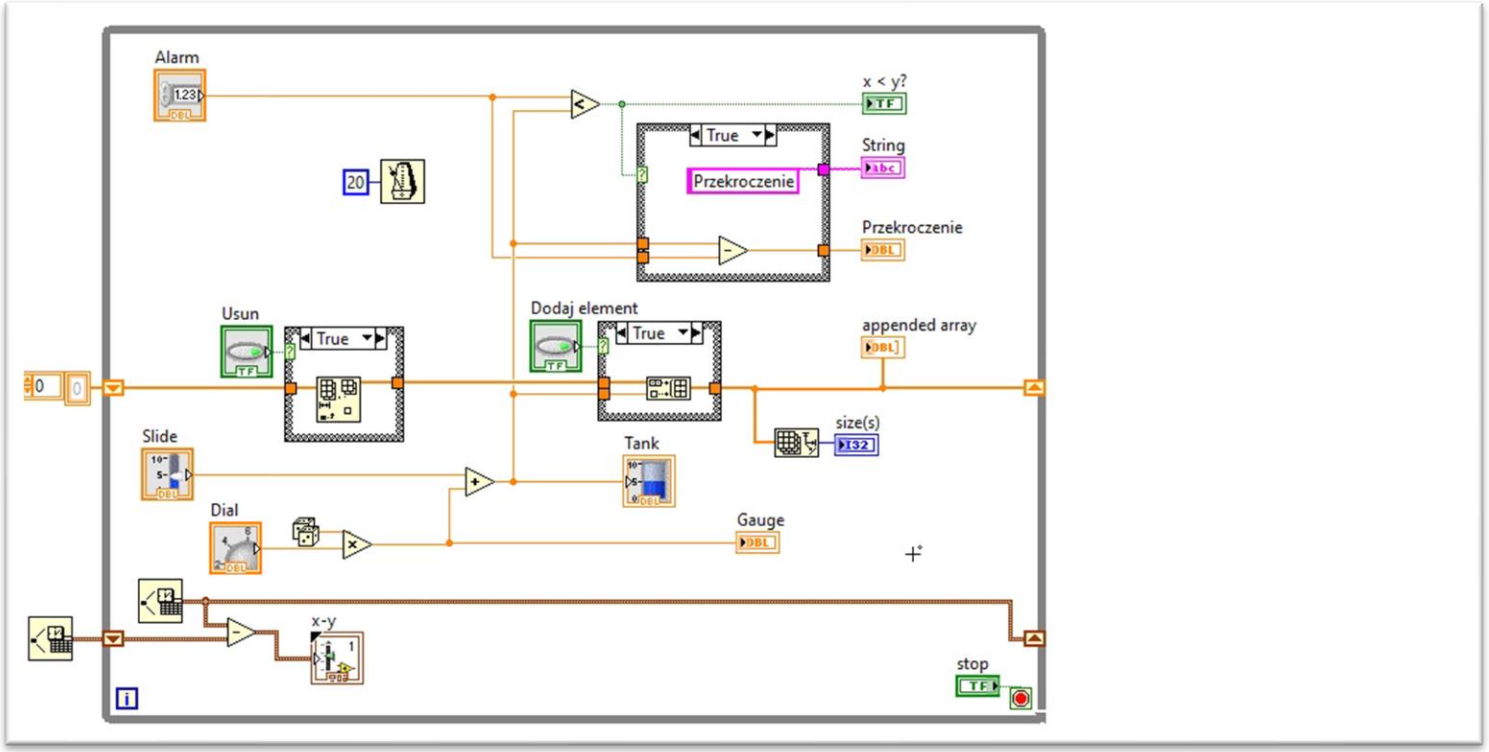
Na Front Panelu zmień zakres kontrolek Slide i Digital klikając dwukrotnie lewym klawiszem myszy na skraje wartości i ustaw jak w przykładzie. Możesz to zrobić także przez Properties -> Scale ustawiając Scale Range.
Generator liczb losowych da tu efekt oscylacji poziomu cieczy w zbiorniku.



Przetestuj kod.

Przydatna porada

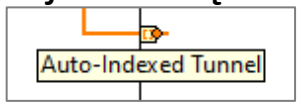
Kod już się zagęścił i brakuje miejsca ? Oczywiście możemy coś poprzesuwać i zamienić wygląd terminali na bardziej kompaktowy przez kliknięcie na nim prawym i odznaczenie View As Icon. Można też złapać za prawą krawędź ramki pętli While i ją rozciągnąć w prawo, jest na to też inny sposób. W LabView niema możliwości powiększania „zoom'owania" za to można sobie rozciągnąć przestrzeń jak ciasto przy rozwałkowywaniu.

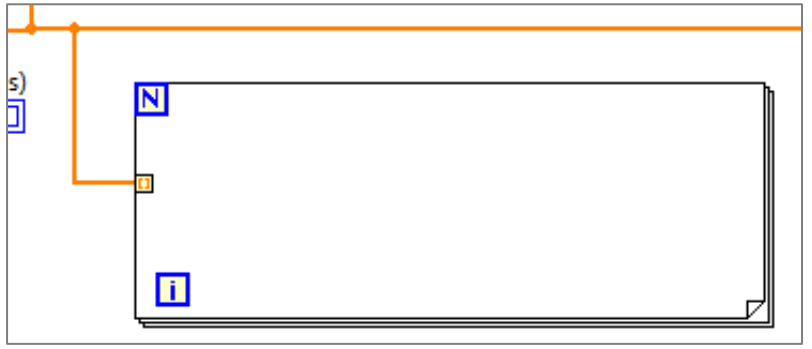


Trzymając Ctrl przeciągnijcie (lewym przyciskiem myszy) w miejscu w którym chcielibyście rozciągnąć.

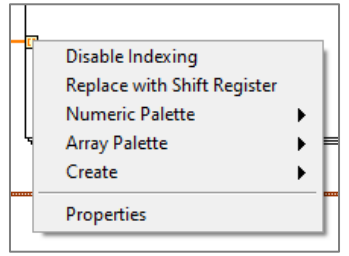
9 Wprowadzenie pętli for

Zrobiło się trochę wolnego miejsca zbuduj tam pętlę for z grupy Structures -> For Loop i do jej lewej krawędzi przyłącz nitkę z tablicą. Powstanie wejście do pętli w trybie auto indeksowania tj. wewnątrz pętli mamy dostęp do pojedynczego elementu tablicy.

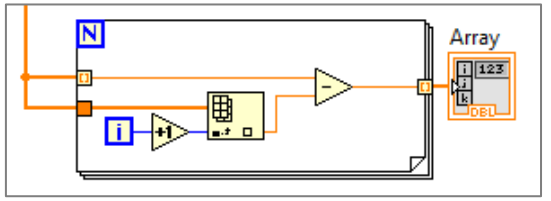




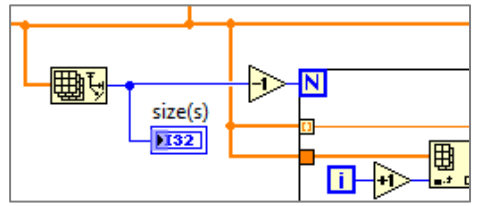
Pętla wykona się tyle razy ile jest elementów tablicy mimo niepodłączonego wejścia **N**. Wewnątrz pętli mamy do dyspozycji także **i** czyli numer iteracji (wykonania się) pętli. W pewnych okolicznościach będziemy potrzebowali dostępu do całej tablicy wewnątrz pętli i możesz wyłączyć indeksowanie klikając prawym przyciskiem myszy na „kwadraciku" i przełączając odpowiednią opcję.



Przygotuj kod wyznaczający przyrosty y(i) - y(i+1) czyli wahania poziomu wody w zbiorniku. Użyj bloczka Array -> Index Array. (+1) czyli Increment znajdziesz w grupie Numeric. Przetestuj kod.



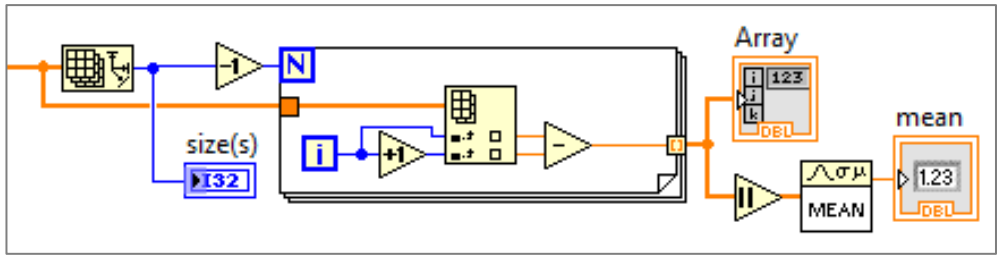
Zauważysz że ostatni element nowej tablicy Array jest liczony błędnie, przyrostów powinno być o jeden mniej niż poziomów dlatego ograniczyć ilość wykonań pętli. Zauważ że bloczek Index Array możesz rozciągnąć w dół i wyciągnąć dwa elementy tablicy.



9 Wprowadzenie pętli for

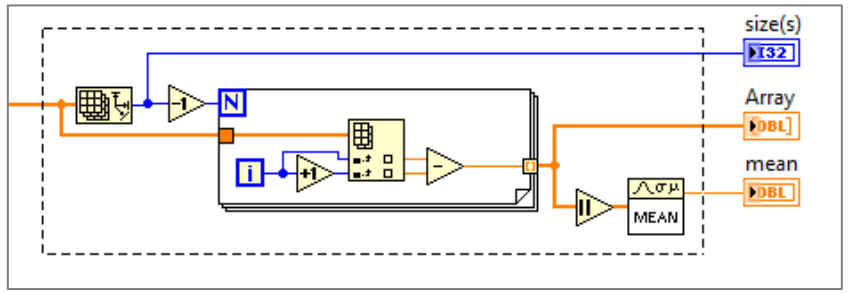
W Block Diagramie dotychczas korzystaliśmy z elementów z makro grupy Programming która jest domyślnie rozwinięta gdyż jest wykorzystywana jej elementy są wykorzystywane najczęściej. W innej grupie Mathematics znajdziecie przydatne funkcje jak Prob&Stat -> Mean która liczy wartość średnią z tablicy. Są tam też inne przydatne funkcje które warto sprawdzić. Ustawcie okno Contekst Help w takim miejscu żeby było widoczne podczas wyboru funkcji a będziecie widzieli podpowiedzi i funkcje bloczków.

Wyznacz wartość średnią modułu oscylacji poziomu cieczy w naszym zbiorniku.



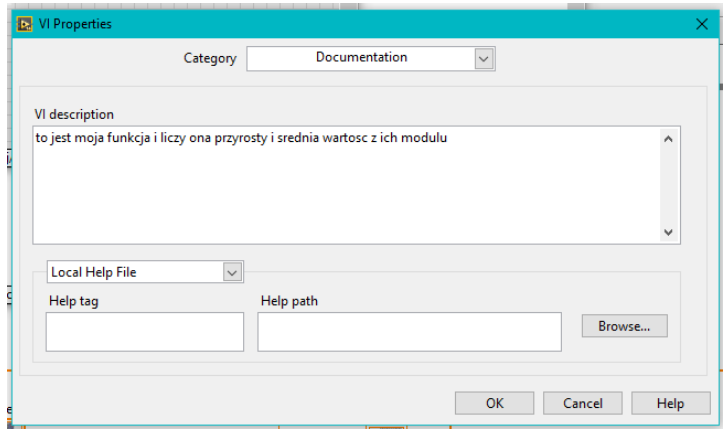
10 Funkcje

Ten sam kod możemy „ładnie poustawiać” i zamknąć kilka operacji w jedną nową własną funkcję która będzie miała jedno wejście i trzy wyjścia.



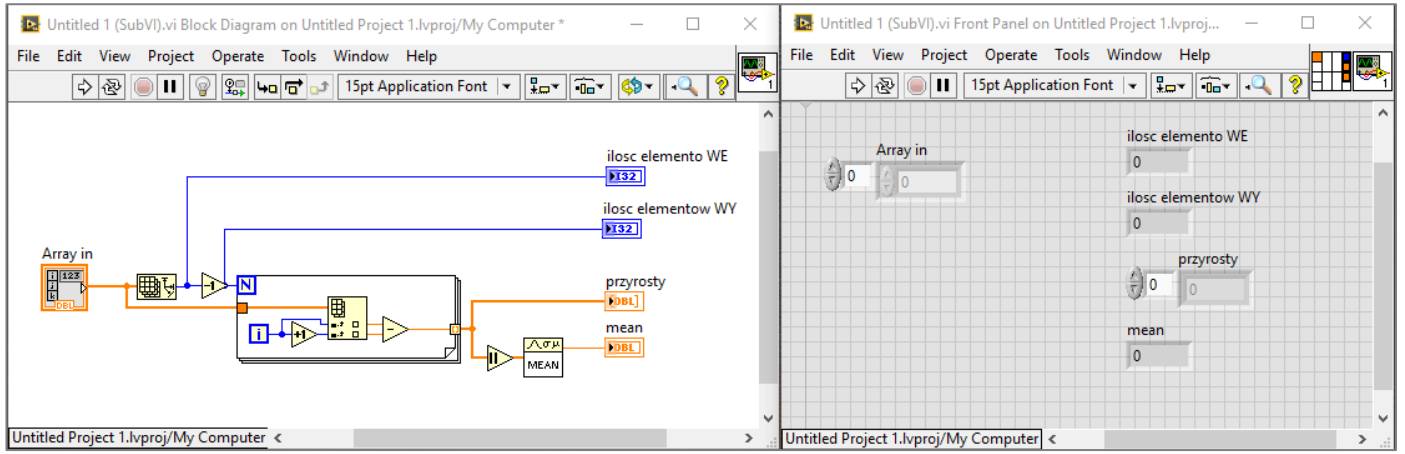
Trzymając lewy przycisk myszy zaznacz część kodu a następnie z menu Edit wybierz Create SubVi. Powstanie nowy bloczek w Block Diagramie, ale też w projekcie.
Sprawdź w Project Explorerze powstał (SubVI).

W Block Diagramie kliknij na niego dwukrotnie otworzy się nowe okno będą tam kontrolki (wejściowe) i indicatory (wyjścia). W oknie Context Help jest też opis. Ten opis możemy edytować oczywiście zmieniając nazwy kontrolek i indykatorów żeby ich nazwy coś oznaczały. Dodatkowy opis możemy ustawić wywołując z menu File -> Vi Properties naszego SubVi i w kategorii Documentarions w oknie Vi description możesz umieścić opis który będzie wyświetlany w Context Help.

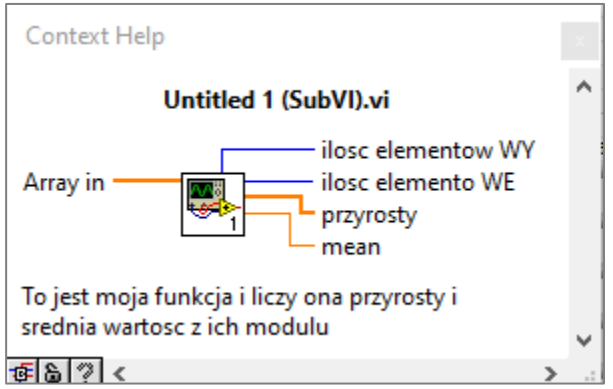


10 Funkcje

Dodaj indicator który będzie pokazywał ile elementów zawiera tablica wyjściowa, jak na schemacie.
Zwróć uwagę na ikonę w prawym górnym rogu okna Front Panel.

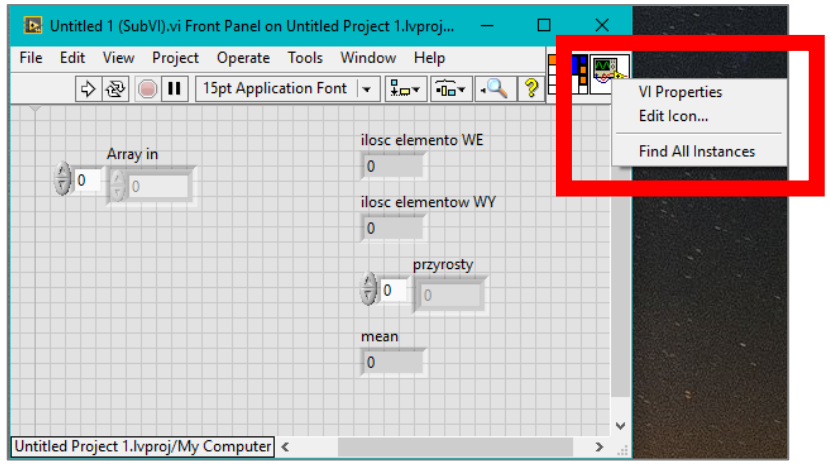


Pola podświetlone oznaczają utworzone podłączenia kontrolek indukatorów. Najeżdżając myszką na wolne pole kursor zmieni wygląd na „szpulkę” jak wybierzesz któreś pole (klikając lewym przyciskiem myszy) możesz połączyć je z nowo utworzonym indicatorem (klikając lewym przyciskiem myszy). Sprawdź jakie są dostępne opcje jeśli klikniesz na tym obszarze prawym przyciskiem myszy.

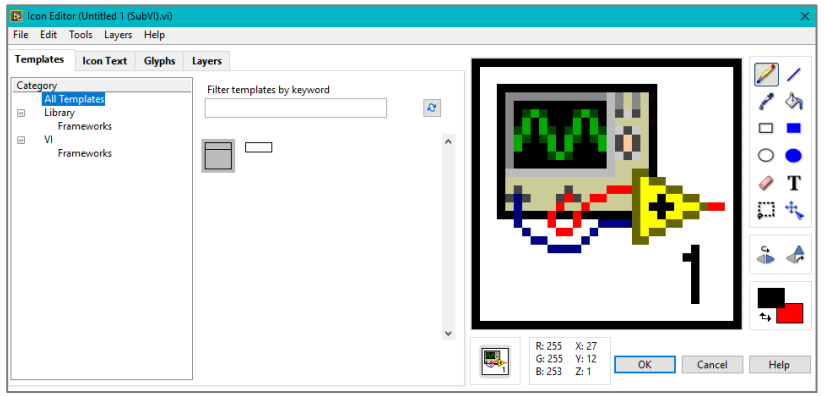


Zauważ że zwyczajono wejścia są podłączane od lewej a wyjścia po prawej, ale możesz użyć dowolnego pola (terminala). W oknie Context Help wejścia są uszeregowane po lewej, wyjścia po prawej.

10 Funkcje – edycja ikon



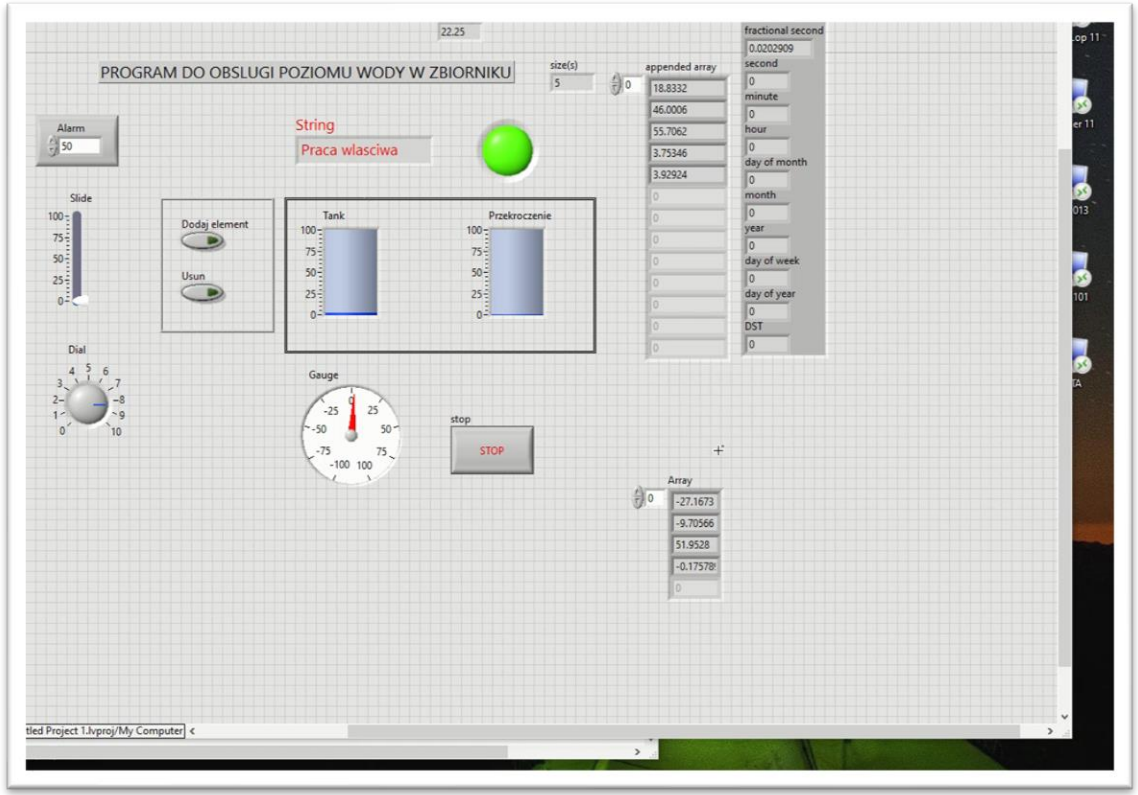
Twoja funkcja znajduje się już w kodzie i działa świetnie, możesz ją powielać (wstawiać w różnych miejscach). Jej wygląd niewiele mówi możesz to zmienić. Klikając prawym przyciskiem myszy na ikonie możesz przywołać edytor ikon.



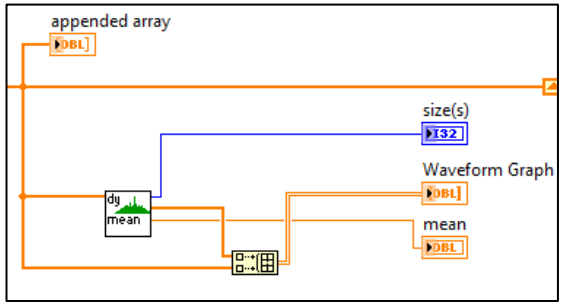
W oczy rzuca się prawa część gdzie jest podgląd ikony i kilka narzędzi znajomych z Paint'a. Spróbuj utworzyć swoją ikonę.

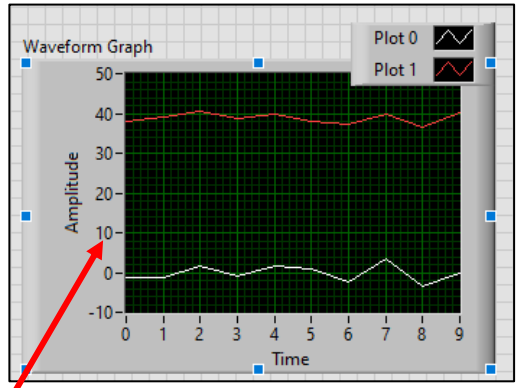
Jeśli nie masz duszy artysty to zwróć uwagę na zakładki w lewej części Edytora ikon gdzie Templates są puste, ale masz jeszcze zakładkę od tekstu, przykładowe grafiki (przeciągnij i upuść), i to co jest najlepsze czyli warstwy gdyż tekstem i przeciąganymi grafikami możesz zarządzać, tymi rysowanymi samodzielnie już nie.
Stwórz własną ikonę.

11 Reprezentacja graficzna elementów tablic



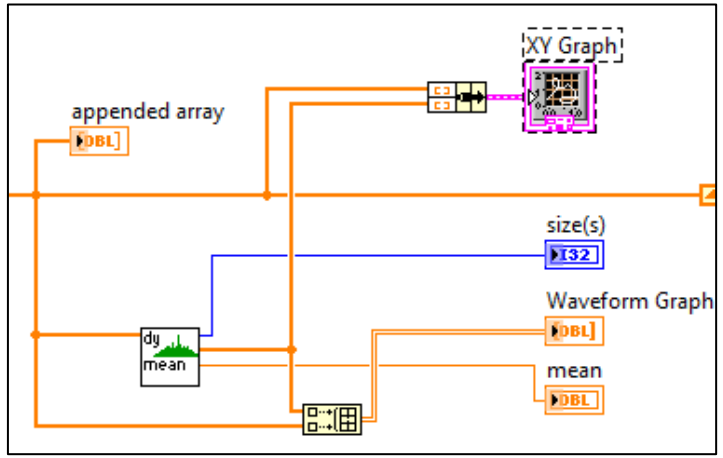
Reprezentacja graficzna tablic w rozumieniu wykresu. Wykresy zawarte są w grupie Graph i podstawowym elementem jest Waveform Graph. Możesz wstawić go i dzielnie podłączać, ale teraz zamień indicator tablicy z tekstowego na graficzny. Jest to podstawowa forma gdzie na osi poziomej domyślnie jest zapisany Time i jest to nr. elementu w tablicy. Możemy dołożyć drugą serię.



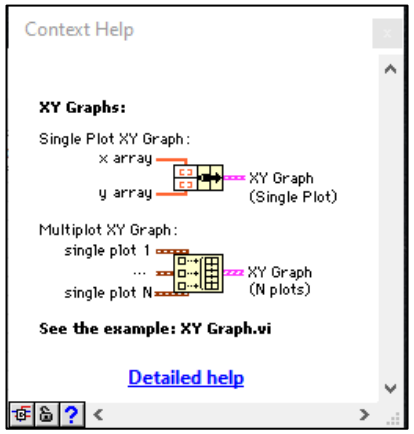


Koniecznie zapoznaj się z właściwościami, zmień opisy osi i nazwy serii, kolory kratek w tle. Aby dołożyć drugą oś pionową należy kliknąć prawym przyciskiem myszy na osi i Duplicate Scale, możesz także przerzucić ją na drugą stronę wykresu Swap Sides. Klikając prawym zwróć uwagę na Auto Scale który czasem jest niepotrzebny.

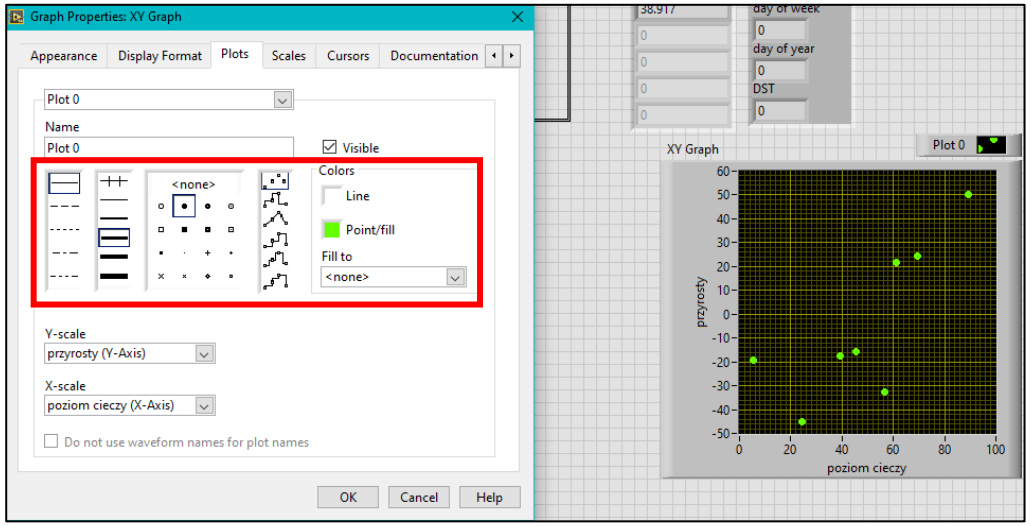
11 Reprezentacja graficzna elementów tablic



Gdy zależy nam na umieszczeniu na osi rzędnych jakiejś wielkości to musimy posłużyć się wykresem XY. Do takiego wykresu podpinamy strukturę składającą się z dwóch tablic reprezentujących (x i y) punktu na wykresie. Chodzi o punkty z jednej serii gdyż jeśli chcemy kilka serii na jednym wykresie XY to podepniemy tablicę struktur. Sprawdź opis w oknie Contekst Help.

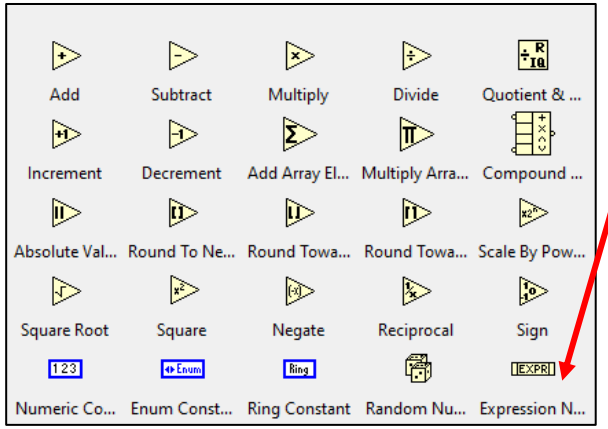


Zwróć uwagę na możliwości reprezentacji serii przez wybór stylu linii jej grubości oraz stylu punktu i prowadzenia linii między punktami jak również koloru linii i punktów.



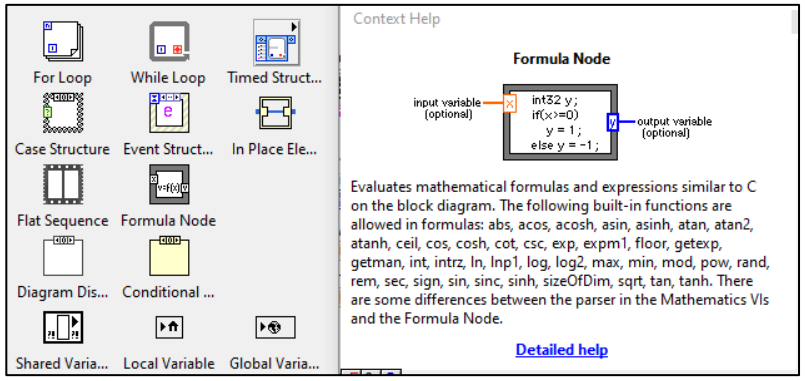
12 Operacje matematyczne

Operacje matematyczne możemy wykonywać na trzy sposoby: podstawowe bloczki operatorów i operatory własne, napisanie kodu wewnątrz Formula Node, oraz wykorzystanie funkcji z odpowiedniej grupy operacji matematycznych.



Bardzo użyteczny jest Expresion Node gdzie możesz napisać funkcje jednego parametru tj. jedno wejście jedno wyjście.

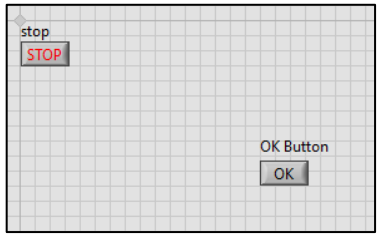
Formula Node pozwala na pisanie kodu podobnego do języka C w jego podstawowej funkcjonalności. Dostępne są pewne funkcje matematyczne. Można tworzyć kilka wejść nazywając je np. x, a, b, oraz wyjścia gdzie zmienne wyjściowe oraz pomocnicze muszą zostać zadeklarowane (jak w C). Występuje także odpowiednik Matlabowy gdzie możliwe jest importowanie skryptu.



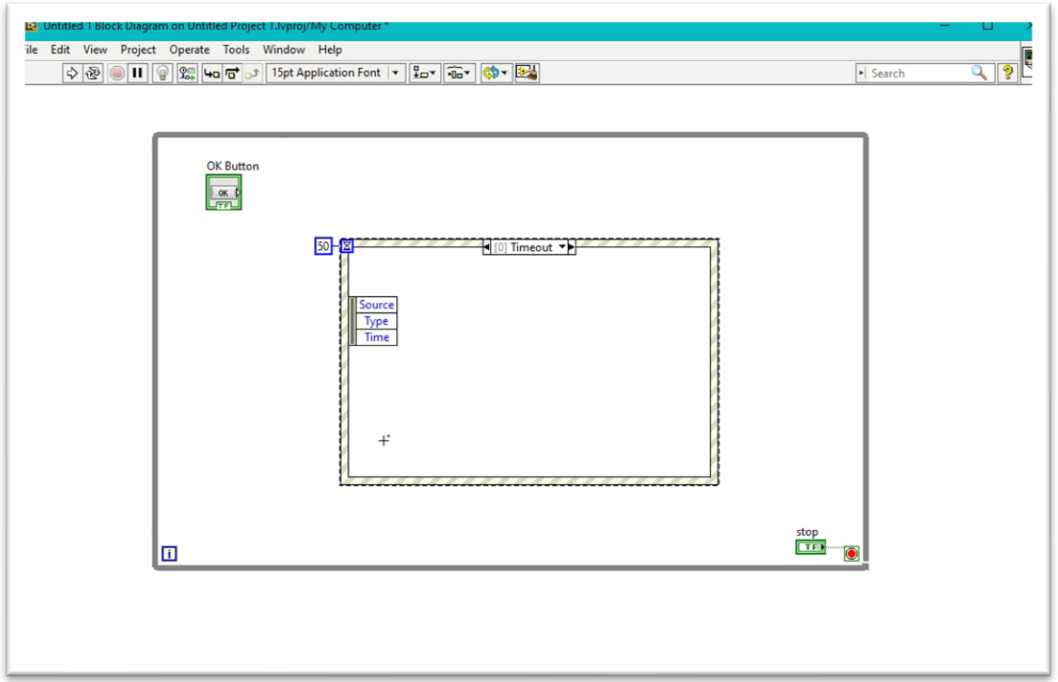
Grupa operacji matematycznych zawiera specjalizowane funkcje których wykorzystanie wymaga zapoznanie się z ich opisem. Należy liczyć się z tym że mogą zużywać znaczne zasoby sprzętowe i spowolnić działanie programu.

13 Obsługa zdarzeń

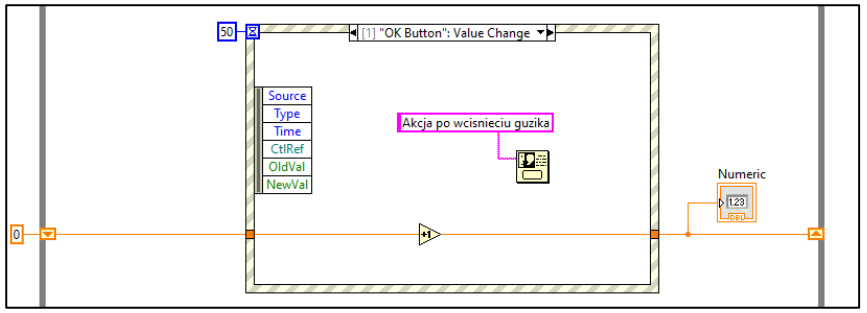
Proponuję w nowym pliku (Ctrl+N) w pętli while założyć strukturę Event Structure która będzie czekała na wystąpienie tzw. zdarzenia w programie i wykona odpowiednie operacje a jeśli nie wystąpi zdarzenie to wykona to co jest w Timeout.



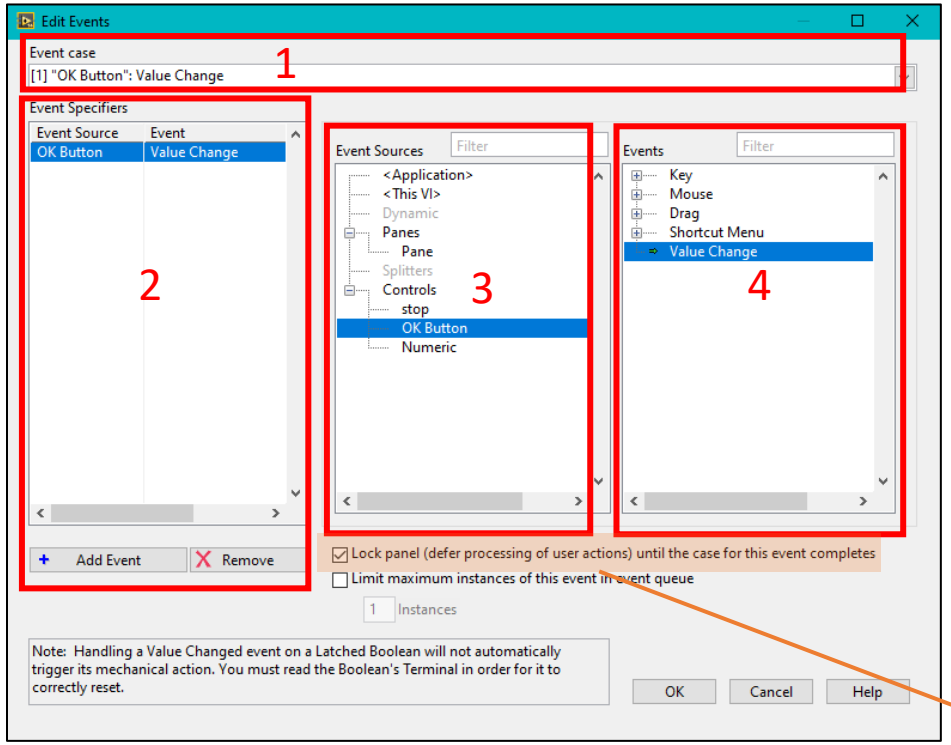
Wstawmy na Front Panelu pojedynczy guzik OK Button i stwórzmy dla niego zdarzenie wywoływane zmianą jego stanu.



W Block Diagramie z grupy Dialog & User Interface wybierz One Btn Dialog i umieść go wewnątrz nowego zdarzenia oraz przeprowadź nitkę jak na schemacie. W zdarzeniu Timeout połącz nitkę tak by przechodziła bez zmian. Po wciśnięciu przez użytkownika guzika OK będzie on informowany oraz będzie zwiększany licznik. Przetestuj program.



13 Obsługa zdarzeń – konstruowanie handlera



Metoda obsługi zdarzenia składa się z dwóch etapów gdzie pierwszy to przygotowanie konfiguracji co (jaka akcja użytkownika) uruchamia zdefiniowane zdarzenie. Często spotykany jest termin handler czyli zbiór parametrów których spełnienie powoduje uruchomienie jakichś funkcji warunkowych.

W oknie edycji zdarzeń wyróżniamy 4 pola:

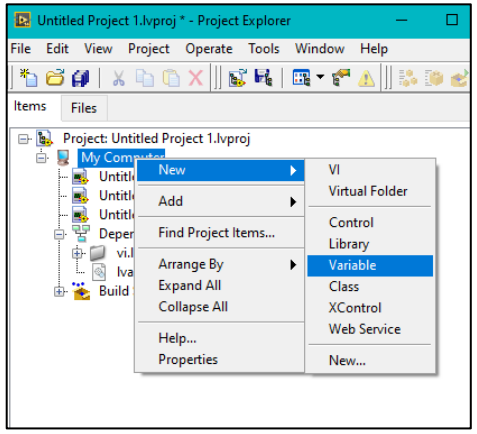
1. lista funkcji warunkowych (casów) – wybierz które edytujesz,
2. zdefiniowane zdarzenia wywołujące daną funkcję warunkową,
3. element front panelu który ma wywołać zdarzenie,
4. rodzaj akcji podjętej przez użytkownika która wywoła zdarzenie.

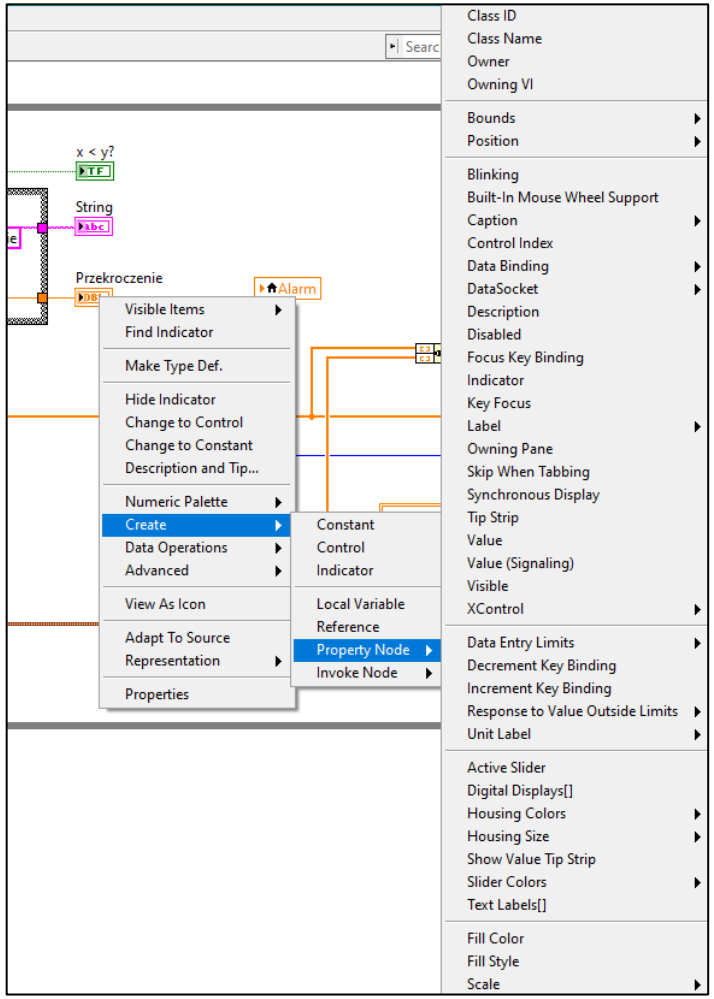
Uwaga: Standardowo blokowany jest Front panel co może być niepożądane.

ps. Ten cały Event Structure to taki jakby case structure który się sam wykona jak użytkownik np. wciśnie jakiś guzik, ale możesz ustawić żeby wystarczyło najechanie na coś albo kliknięcie prawym guzikiem gdzieś na wolnym polu.

14 Zmienne oraz wyciąganie wartości kontrolek

Daje się programować bez użycia zmiennych ciągając te nitki po całym ekranie, ale można sobie ułatwić wykorzystując zmienne oraz własności. Zmienne oczywiście są globalne oraz lokalne. Zmienne lokalne mogą reprezentować kontrolki i indicatory przez wartości zaś zmienne globalne (współdzielone) to nowy twór dodawany w oknie projektu i tak dodaną zmienną możemy przeciągnąć na Block Diagram w dowolnym miejscu projektu.

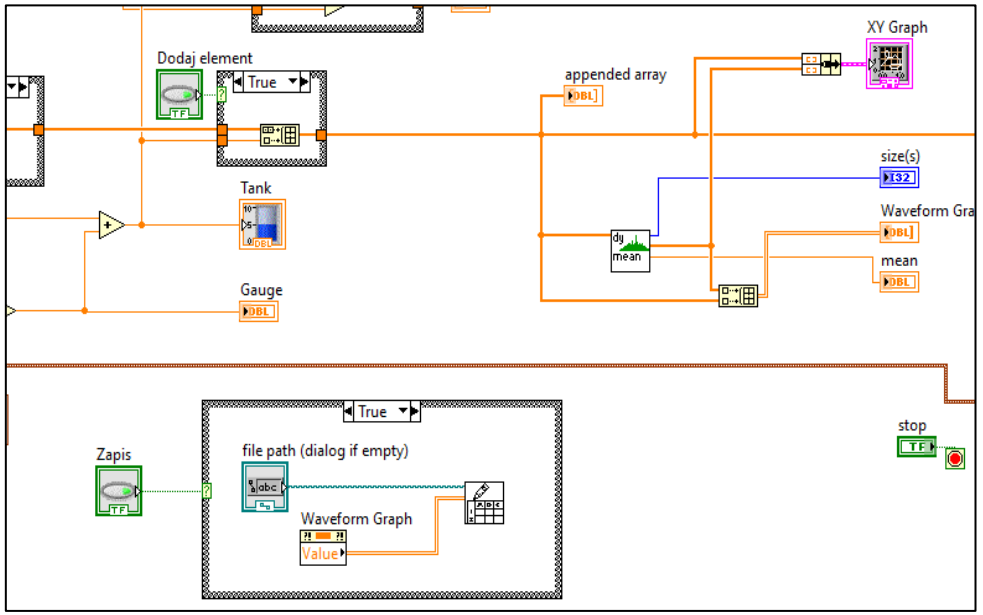


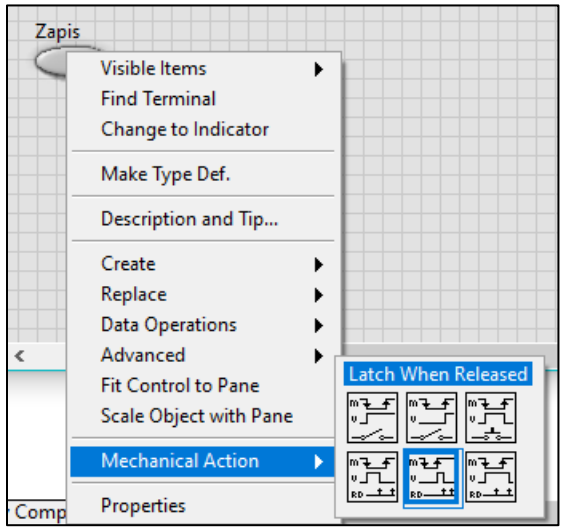


Każda kontrolka oraz indicator ma cechy i do tych cech mamy dostęp przez Property Node. Kliknij prawym przyciskiem myszy na kontrolce na Block Diagramie i sprawdź ile ciekawych cech możesz zmieniać w trakcie działania programu. Możesz np. zmieniać kolor punktów na wykresie albo przesuwać guzik uciekając przed myszką użytkownika. Jedną z użyteczniejszych cech jest wartość (value) i możesz korzystać z niej jak ze zmiennej. Zauważ że są dwa typy gdzie zwykłe value nie wywołuje Eventu zaś Value (Signaling) może zostać użyte do wywołania eventu.

15 Obsługa plików

Zaczynamy w Block Diagramie w grupie File I/O pierwsze to Wrire/Read Spreadshet i są to bardzo użyteczne operacje służące do zapisu i odczytu danych w formie tabelarycznej. Zauważ że na wejściu/wyjściu dane są w postaci wektora lub tablicy. Jest tam też kilka innych wejść z którymi należy się zapoznać. W późniejszym etapie przydają się także elementy z podgrupy Advanced File Functions.
Wprowadź w programie zapis wyników do pliku.





Zwróć uwagę na pracę przycisku aktywującego zapis do pliku. Powinien pozwolić na zapis raz po wciśnięciu guzika i guzik powinien sam odskoczyć tak aby operacja zapisu wykonała się tylko raz (tryb Latch). Jeśli tryb pracy guzika byłby Switch to przy wciśnięciu guzika na ON zwracana wartość była by True i program zapisywałby wielokrotnie te same dane aż użytkownik ponownie zmieniłby stan na OFF.
Zwróć na to uwagę gdyż nie daje się budować eventów na guzikach (Latch) a jedynie na przełącznikach (Switch).