Podstawy programowania w LabView (1)

Środowisko programistyczne i pierwszy program

Celem tego kursu jest przedstawienie podstaw nowoczesnego graficznego języka programowania zastosowanego w LabView. Środowisko to staje się coraz bardziej popularne, ponieważ pozwala szybko i łatwo przygotować aplikację pomiarową. Udostępnia bogaty zestaw bibliotek do komunikacji przez niemal wszystkie interfejsy. Do tej pory najczęściej stosowane w laboratoriach naukowych i studenckich uczelni technicznych, obecnie chętnie stosowane w wielu firmach. Kurs jest skierowany do osób, które po raz pierwszy spotykają się z tym środowiskiem. Prezentowane zagadnienia będą zilustrowane przykładami.

LabView jest zintegrowanym środowiskiem programistycznym przeznaczonym głównie do sterowania systemami kontrolno – pomiarowymi oraz przetwarzania i analizy danych. Ma biblioteki umożliwiające komunikację pomiędzy urządzeniami, komputerami, biblioteki realizujące zaawansowane obliczenia matematyczne statystyczne i wiele innych.

Programowanie w tym środowisku jest zupełnie odmienne od standardowego podejścia, w którym linie kodu zapisane w postaci tekstowej wykonywane są kolejno linia po linii. W języku graficznym kod programu przedstawiony jest w postaci diagramu, procedury reprezentowane są przez ikony natomiast połączenia między nimi decydują o przepływie danych. Ten sposób programowania zwalnia nas z konieczności poznania składni języka, troszczenia się o prawidłowe wstawienia przecinków czy średników. Kompilowanie programu przebiega automatycznie po wstawieniu funkcji czy wykonaniu połączenia pomiędzy elementami diagramu. Dzięki temu natychmiast wiemy czy wykonana operacja jest źródłem błędu. Informacje obłędach są jasne i czytelne oraz wskazują dokładnie miejsce ich wystąpienia. Nauka programowania polega na poznaniu kilku zasad i struktur specyficznych dla tego języka, oraz dostępnych funkcji i umiejętności posługiwania się nimi.

Na Potrzeby kursu wystarczy wersja ewaluacyjna można ją pobrać ze strony producenta (http://sine.ni-.com/np/app/main/p/docid/nav-104/lang/pl/fmid/1765/), jest ona w pełni funkcjonalna. Jej wadą jest krótki czas działania, bo tylko 7 dni, ale po zainstalowaniu istnieje możliwość przedłużenia go o kolejne 45 dni. Można również skorzystać ze starszych wersji LabView, które będą służyć przez 30 dni. Poza wyglądem okna dialogowego nie zauważymy istotnych zmian. Wadą w tym wypadku jest to, że nie można otworzyć plików z wyższych wersji oprogramowania.

Pracując z LabView będziemy posługiwali się pojęciem przyrządu wirtualnego (*virtual instruments*), które nie jest dokładnie zdefiniowane w rzeczywistości i może oznaczać wirtualny instrument znajdujący się tylko na ekranie komputera lub program służący do sterowania urządzeniami zewnętrznymi. Każdy program przygotowany w LabView jest nazywany instrumentem wirtualnym (*virtual instruments - VI*), ponieważ składa się z dwóch elementów: panelu przedniego (*front panel*) i diagramu (*block diagram*). Rozszerzenie Vi plików przygotowanych w LabView pochodzi od nazwy *virtual instruments.*

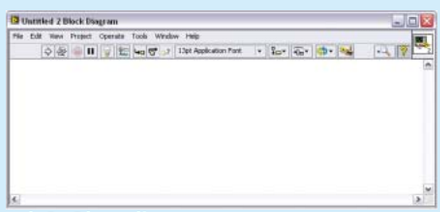
Rysunek 1. Okno panelu czołowego

Okna edycji

Po uruchomieniu pakietu widoczne jest okno dialogowe, w którym z menu *File* wybieramy *New VI.* Otworzą się dwa okna. To mające szare tło nosi nazwę *Front panel* (**rysunek 1**). W tym oknie będziemy umieszczać kontrolki *(Controls)* służące do sterowania programem i przekazywania wartości wejściowych oraz wskaźniki *(Indicator)*, przez które program zwraca dane do nadrzędnych instrumentów wirtualnych lub na panel główny w celu wizualizacji wyników na monitorze.

Na **rysunku 2** pokazano okno diagramu. Będziemy w nim umieszczali ikony symbolizujące funkcje, instrukcje strukturalne sterujące wykonaniem programu i połączenia pomiędzy nimi decydujące o przepływie danych.

Każdy z programów ma podobną budowę: zawiera panel czołowy i diagram, ale nie oznacza to, że wywołanie każdego z podprogramów spowoduje wyświetlenie panelu czołowego. Podczas edycji możemy zobaczyć pa-



Rysunek 2. Okno diagramu

nel każdego z podprogramów, ale podczas uruchamiania obserwujemy jedynie panel nadrzędnej aplikacji.

Rysunek 3. Paleta narzędziowa (Tools palette)

Palety narzędzi

Tworząc programu korzystamy z palet narzędziowych. Do dyspozycji mamy trzy palety. Paletę narzędziową (*Tools palette,* **rysunek 3**) zawierająca narzędzia do edycji diagramu i panelu czołowego służące do:

- automatycznego doboru narzędzi,
- zmiany nastaw obiektów,
- zmiany wymiarów, położenia i zaznaczania obiektów,
- wstawiania napisów,
- narzędzie do łączenia obiektów, działające tylko na diagramie.
- rozwijania menu, wskazywanych obiektów
- przesuwania okna
- do wstawiania punktów zatrzymania programu, działające tylko na diagramie.
- wstawiania sond wyświetlających wartości wybranych sygnałów, działające tylko na diagramie.
- pobierania koloru
- zmiany koloru (za pomocą poprzedniego narzędzia możemy wskazać kolor, aby nie różnił się od wymaganego).

Paleta jest dostępna w oknie panelu czołowego i diagramu. Jeśli jest nieaktywna, możemy ją włączyć wybierając z menu *View -> Tools Palette*.

Przygotowując panel czołowy korzystamy z palety kontrolek (*Controls*) **rysunek 4**.

Wyświetlenie jej następuje po kliknięciu prawym przyciskiem myszy wewnątrz okna panelu czołowego. W lewym górnym rogu znajduje się symbol pineski, kliknięcie w niego powoduje pozostawienie palety aktywnej do czasu jej zamknięcia. Po wybraniu odpowiedniego elementu klikamy na niego i przeciągamy go myszką do okna panelu. W ten sposób umieszczamy element na panelu czołowym i jednocześnie ikonę z nim związaną na diagramie. Wszystkie elementy są posegregowane według wyświetlanych typów i stylu.

Jako pierwsza zwykle znajduje się kontrolka (*Controls*) przekazująca dane do programu a obok niej taki sam wskaźnik (*Indicator*) wyświetlający wartości zwracane przez program. Elementy podzielone są ze względu na typ przekazywanych zmiennych oraz na styl (wygląd) kontrolki. Domyślną grupą są elementy dostępne w zakładce *Modern*. Polecam stosowanie elementów z grupy *System*. Funkcjonalnie są takie same, ale ich wygląd zmienia się w zależności od systemu, pod kontrolą którego jest uruchomiony program. Dzięki temu panel przedni jest zawsze aktualny.

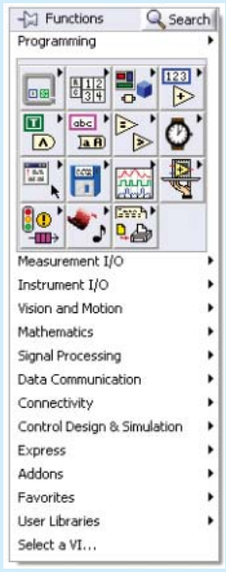
Najczęściej używane grupy kontrolek to:

- numeryczne (*Numeric*) – elementy pozwalające na przekazywanie zmiennych numerycznych np. za pomocą suwaków czy pokręteł.
- logiczne (*Boolean*) – elementy przekazujące zmienne logiczne np. jako przełączniki czy wskaźniki w postaci diody LED.
- tekstowe (*string & patch*) – elementy przekazujące zmienne tekstowe oraz ścieżki dostępu do pliku.
- tablice i klastry danych (*Array & Cluster*) - elementy przekazujące tablice i klastry danych (*odpowiednik struktury w języku C*).
- Graficzne (*Graph*) – elementy wyświetlające dane w postaci graficznej, wykresy.

Rysunek 4. Paleta kontrolek (Controls).

Przygotowując diagram będziemy się posługiwali paletą funkcji (*Functions*) pokazaną na **rysunku 5**.

Wyświetlenie jej następuje również poprzez kliknięcie prawym klawiszem myszy, ale tym razem w oknie diagramu. Paleta ta nie jest dostępna w oknie panelu czołowego. Podobnie jak poprzednio, za pomocą pineski możemy pozostawić ją widoczną przez cały czas. Wybrane funkcje umieszczamy na diagramie przeciągając je myszką. Najczęściej będziemy posługiwali się funkcjami z zakładek:

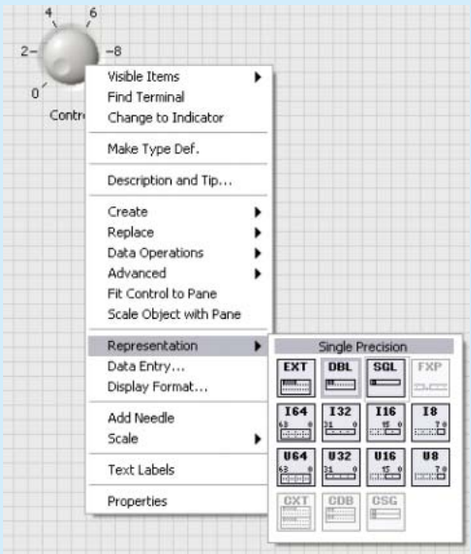


Rysunek 5. Paleta funkcji (Function)

- struktury (*structures*) – zawiera funkcje strukturalne, zmienne globalne i lokalne.
- tablice (*Array*) - zawiera funkcje do obsługi tablic.
- klaster, klasa (*Custer, Class & Variant*) – zawiera funkcje do obsługi elementów typu klaster.
- numeryczne (*Numeric*) – funkcje operujące na liczbach.
- logiczne (*Boolean*) – funkcje operujące na zmiennych logicznych.
- łańcuch (*String*) – funkcje operujące na zmiennych tekstowych.
- porównanie (*Comparison*) – operacje porównania.
- (*Timing*) – zawiera funkcje związane z odmierzaniem czasu.
- pliki (*File I/O*) – zawiera funkcje do obsługi plików.

Reprezentacja zmiennych na diagramie

W języku graficznym – podobnie jak w innych językach – zdefiniowano kilka typów danych. Określają one, jakiego rodzaju informacje mogą przechowywać zmienne i jaki jest ich zakres. Postarano się, aby w łatwy sposób można było rozpoznać, jaki typ reprezentuje zmienna. Zrobiono to poprzez przypisanie zmiennym kolorów. Dzięki temu, patrząc na diagram, możemy od razu zauważyć, z jaką zmienną mamy do czynienia. O jej rodzaju informuje nas również wygląd ikon na diagramie. Po wyglądzie zmiennej możemy również rozpoznać czy jest to kontrolka **(źródło danych**, jej obrys jest pogrubiony i terminal połączeniowy znajduje się z prawej strony), czy wskaźnik (element wyświetlający dane, obrys jest cieńszy a termi-



Rysunek 6. Zmiana reprezentacji danych zmiennej numerycznej.

nal znajduje się z lewej strony). Istnieją również stałe, które są pokazane tylko na diagramie.

Wygląd zmiennej można ustalić na dwa sposoby: może ona być ikoną lub terminalem. W pierwszym wypadku ikona reprezentująca zmienną jest większa, ale niesie więcej informacji, ponieważ możemy rozpoznać związany z nią element na panelu czołowym. W drugim wypadku jest mniejsza i nie zaciemnia nam diagramu. Zmianę wyglądu uzyskujemy klikając na niej prawym klawiszem myszy i zaznaczając lub odznaczając w menu lokalnym opcję *View As Icon*. W **tabeli 1** umieszczono ikony i typy kilku podstawowych zmiennych.

Definiowanie zmiennych

Zmienną definiujemy poprzez wstawienie odpowiedniego elementu na panelu czołowym. Wybierające element reprezentujący dane logiczne, na przykład przycisk, tworzymy zmienną logiczną przekazującą jedną z dwóch wartości: 0 lub 1. Wybierając element zawierający dane numeryczne np. pokrętło czy suwak tworzymy zmienną numeryczną reprezentującą liczby zmiennoprzecinkowe, całkowite lub zespolone. W przypadku zmiennych numerycznych należy zdefiniować, które dane są reprezentowane. Możemy to zrobić na panelu czołowym lub na diagramie. W pierwszym wypadku, klikamy prawym przyciskiem myszy na wstawionym elemencie wybieramy opcję *Representation* i wskazujemy właściwy typ. To samo można zrobić na diagramie, klikając w tym wypadku na ikonie.

Linijka przycisków narzędziowych

Zarówno na panelu czołowym oraz na diagramie u góry znajduje się linijka przycisków narzędziowych. Większość z nich jest taka sama, w oknie diagramu znajduje się kilka dodatkowych umożliwiających śledzenie wykonania programu:

- (*Run*) przycisk rozpoczyna wykonywanie programu (biała strzałka sygnalizuje gotowość do uruchomienia, zaczerniona – wykonywanie programu, przełamana – błąd na diagramie, kliknięcie w tym momencie w strzałkę wyświetla listę błędów).

- (*Run Continuously*) **rozpoczyna wykonanie programu w trybie ciągłym. Program wykonuje się tak, jakby pracował w pętli.**

Tabela 1. Ikony i podstawowe typy zmiennych

Ikona	Terminal	Typ
	SGL	Typ numeryczny, liczby zmiennoprzecinkowe: - pojedynczej precyzji - podwójnej precyzji - rozszerzonej precyzji
	DBL	
	EXT	
	I8	Typ numeryczny, liczby całkowite ze znakiem - 8 bitowe - 16 bitowe - 32 bitowe - 64 bitowe
	I16	
	I32	
	I64	
	U8	Typ numeryczny, liczby całkowite bez znaku - 8 bitowe - 16 bitowe - 32 bitowe - 64 bitowe
	U16	
	U32	
	U64	
	TF	Typ logiczny, przyjmuje wartości FALSE(0) lub TRUE (1)
	abc	Typ znakowy
	[SGL]	Tablica, która może zawierać elementy różnego typu
		Klaster danych – jest odpowiednikiem struktury w języku C, grupuje dane różnych typów

- (*Abort Execution*) zatrzymanie wykonywania programu.

- (*Pause*) wstrzymanie/wznowienie wykonywania programu.

- (*Highlighting Execution*) włącza/wyłącza animację danych na diagramie (można zaobserwować przepływ danych i kolejność wykonywania instrukcji; program wykonuje się znacznie wolniej i można go obserwować nie używając pułapek; aktywna funkcja jest podświetlona, tak samo jak przekazywane dane).

- (*Step Into*) praca krokowa, po naciśnięciu przycisku wykonywana jest jedna instrukcja. Jeśli na diagramie znajdują się instrukcja zagnieżdżone, to program otwiera je i wykonuje również krok po kroku.
- (*Step Over*) praca krokowa, po naciśnięciu przycisku jest wykonywany pojedynczy krok programu; podprogramy znajdujące się na diagramie nie są otwierane, a wykonywane jak jedna instrukcja.
- (*Step Out*) zakończenie pracy krokowej, aktualny diagram jest wykonywany do końca.
- (*Text Settings*) ustawienia czcionki.
- (*Align Objects*) wyrównanie elementów.
- (*Distribute Object*) wyrównanie odległości pomiędzy elementami.
- (*Resize Object*) zmiana wymiarów zaznaczonych obiektów.
- (*Reorder*) zmiana kolejności nakładających się na siebie obiektów.
- (*Show Context Help*) włączenie/wyłączenie okienka pomocy kontekstowej.

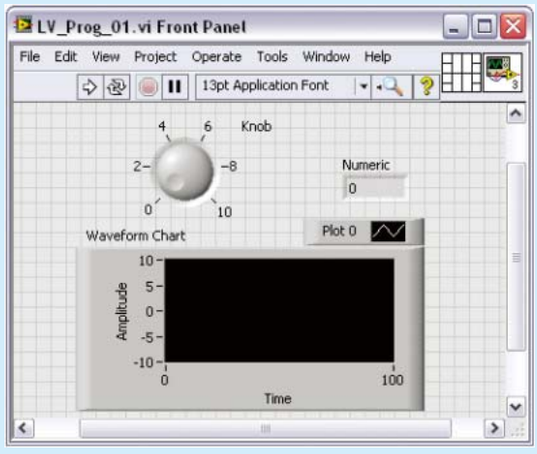
Pierwszy program: przygotowanie, uruchamianie, praca krokowa

Przygotowując pierwszy program nie skupimy się na złożoności programu, tylko przećwiczymy umiejętność posługiwania się LabView. Dlatego będzie to nieskomplikowany program, pobierający wartość z kontrolki na panelu czołowym, zwiększający ją o 1 i zwracający wynik na kilku wskaźnikach.

Po uruchomieniu LabView z menu *File* wybieramy *New VI*. Otworzą się dwa okna panelu czołowego i diagramu. W oknie panelu klikamy prawym przyciskiem myszy i wybieramy z palety *Numeric* na przykład *knob* (pokrętło) i przeciągamy je do panelu czołowego. Teraz możemy przejść do okna diagramu klikając w białym oknie lub wybierając z menu *Windows* ***-> Show Block Diagram***. Znajdziemy tam ikonę oznaczającą pokrętło. Nazwa ikony jest taka sama, jak elementu znajdującego się na panelu czołowym. Możemy wrócić do panelu czołowego klikając na nim myszką lub wybierając z menu *Windows* opcję *Show Front Panel*. Umieśćmy jeszcze dwa elementy: z palety ***Numeric – Numeric Indicator***, który będzie wyświetlał wartości zwracane przez program oraz z palety ***Graph – Waveform Chart*** tj. wskaźnik gromadzący w tablicy przekazane mu wartości i wyświetlający je w postaci wykresu. Panel czołowy powinien wyglądać jak na **rysunku 7**.

Możemy przejść teraz na diagram i przygotować nasz program. Tutaj również klikamy prawym przyciskiem myszy i z palety *Numeric* wybieramy funkcję *Increment*, przeciągamy ją na diagram. Zwróćmy uwagę na przycisk *Run* – biała strzałka zmieniła się na przełamaną sygnalizując błąd na diagramie. Oczywiście, nie popełniliśmy żadnego błędu, ale program po każdej operacji dokonuje sprawdza czy wykonanie kodu jest możliwe.

Ponieważ wejście funkcji nie zostało z niczym podłączone nie przekazujemy do niej żadnych parametrów. Musimy podłączyć je do źródła danych. W naszym wypadku będzie to kontrolka *Knob*. Jeśli mamy włączony automatyczny wybór narzędzi, wystarczy najechać na terminal funkcji lub kontrolki – kursor powinien zmienić wygląd na „szpulkę". Wtedy klikamy i ciągniemy połączenie do drugiego elementu, a po poprowadzeniu

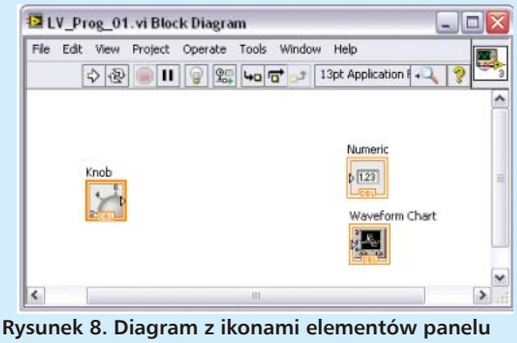


Rysunek 7. Panel czołowy programu

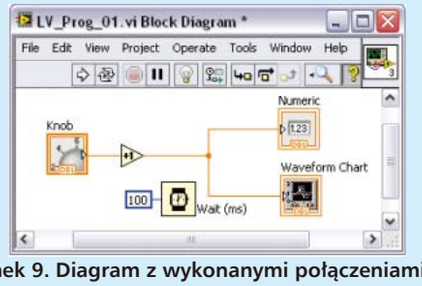
połączenia klikamy ponownie kończąc je. Jeśli wygląd kursora nie zmienił się, to musimy z menu *View* wybrać *Tools Palette*, a następnie włączyć narzędzie *Wiring Tool* i wykonać połączenie. Po tej operacji widok przycisku *Run* znów zmieni się na białą strzałkę, co oznacza, że program jest gotowy do wykonania.

Spostrzegawczy Czytelnicy, że wskaźniki po prawej stronie nie są dołączone, a program nie sygnalizuje błędu. Otóż LabView tego nie wymaga – po prostu wyświetlą wartości domyślne lub ostatnie wskazywane. Pozostało podłączyć wyjście funkcji z naszymi wskaźnikami. Mamy już gotowy program. Ale aby program nie wykonywał się zbyt szybko proponuję dodać jeszcze opóźnienie czasowe. W tym celu z palety *Timing* wstawiamy funkcję *Wait*. Ustawiamy kursor na terminalu z lewej strony funkcji, gdy jego wygląd zmieni się na „szpulkę" klikamy prawym przyciskiem myszy i wybieramy *Create* **->** *Constant*. W ten sposób na diagramie pojawi się stała od razu podłączona do naszej funkcji, wpisujemy do niej wartość 100 wprowadzi to opóźnienie 100 ms.

Oczywiście zamiast stałej można wybrać *Control* i wówczas pojawi się również element na panelu czołowym pozwalający na zmianę opóźnienia podczas



Rysunek 8. Diagram z ikonami elementów panelu czołowego.



Rysunek 9. Diagram z wykonanymi połączeniami

wykonywania programu. Na **rysunku 9** pokazano kompletny diagram. Sprawdzimy teraz, jak można go uruchomić i debugować. Proponuję na ekranie monitora umieścić okna w taki sposób, aby były widoczne jednocześnie. Najprostszym sposobem uruchomienia jest kliknięcie na przycisk *Run* powodujące natychmiastowe wykonanie programu, ponieważ program nie zawiera pętli głównej wykona się bardzo szybko i zobaczymy tylko efekt końcowy.

Przed następnym uruchomieniem zmieńmy wartość w kontrolce *Knob,* pozwoli to na zaobserwowanie zmiany wartości wskaźników. Włączmy teraz animacje przepływu danych na diagramie przyciskiem *Highlight Execution*. I znów *Run* nas program wykonuje się znacznie wolniej a na diagramie widoczne są przepływające po połączeniach „kulki" symbolizujące przepływ danych pomiędzy elementami diagramu. Możemy również zaobserwować wartości, które są przekazywane. Dzięki temu w łatwy sposób można prześledzić działanie nieznanych aplikacji, jak również znaleźć błędy logiczne w własnym programie.

Wyłączmy teraz animację, i uruchommy program przyciskiem *Run Continuously***.** Zawartość diagramu wykonuje się z pełną prędkością, ale po wykonaniu, program rozpoczyna się od nowa tak jak by znajdował się w pętli. Teraz przydatna jest funkcja *Wait* znajdująca się na diagramie. Dzięki niej program nie wykonuje się z zbyt durzą prędkością i zmianę wartości w kontrolce *Knob* obserwujemy w postaci wykresu na *Waveform Chart*. Używając tego trybu możemy przetestować fragmenty kodu, które później mogą być umieszczone w pętli. Oczywiście można również tutaj włączyć animację i śledzić przepływ danych tak jak poprzednio. Możliwe jest wstrzymanie i wznowienie wykonania kodu w dowolnym miejscu korzystając z przycisku *Pause.* Aby zakończyć, należy wcisnąć przycisk *Abort Execution.*

LabView umożliwia również testowanie aplikacji krok po kroku. Do tego celu służą trzy przyciski. Proponuję uruchomić program korzystając z przycisku *Step Into*. Po jego naciśnięciu zostanie wykonana jedna instrukcja, będzie to pobranie wartości z kontrolki *Knob* i przekazanie jej do funkcji inkrementacji. Migająca w czarnym kwadracie funkcja będzie wykonana jako następna. Gdy miga cała ramka diagramu, możemy za pomocą przycisku *Abort Execution* **lub** *Step Out* zakończyć pracę krokową.

Korzystając z funkcji *Step Into* możemy prześledzić nawet zagnieżdżone fragmenty kodu. Gdy dojdziemy do funkcji zagnieżdżonej, LabView przełączy śledzenie na diagram tej funkcji. Aby powrócić na wyższy poziom, należy użyć przycisku *Step Out*.

Jeśli zamierzamy debugować tylko aktualny diagram lub pominąć debugowanie zagnieżdżonej procedury, należy skorzystać z przycisku *Step Over*. Funkcje podrzędne wykonywane są w całości, bez otwierania diagramu i przechodzenia przez wszystkie jego funkcje.

Podobnie jak inne środowiska programistyczne LabView pozwala na ustawianie w dowolnym miejscu programu pułapek i podglądanie zmiennych. Podglądanie zmiennych jest możliwe z użyciem sondy (*Probe*). Sprawdźmy jak działają pułapki. Aby ustawić pułapkę, klikamy prawym przyciskiem myszy na funkcji inkrementacji i z menu wybieramy *Breakpoints* **->** *Set Breakpoints.* Ustawienie pułapki jest sygnalizowane poprzez obrysowanie funkcji czerwoną ramką. Dodamy sobie jeszcze dwie sondy, aby podglądać wartości zmiennych. Pierwszą ustawimy na połączeniu pomiędzy kontrolką *Knob* a funkcją inkrementacji. W tym celu klikamy prawym przyciskiem myszy na połączeniu i wybieramy *Probe*. Na połączeniu pojawi się numer sondy oraz otworzy się dodatkowe okienko, w którym będzie wyświetlana aktualna wartość zmiennej. Analogicznie wstawmy sondę na wyjściu funkcji.

Wciskamy teraz przycisk *Run* – zostanie wykonana część diagramu – do funkcji, która ma ustawioną pułapkę (*Breakpoint*). W okienku *Probe Watch Windows* można zobaczyć wartość pierwszej zmiennej, druga nie została jeszcze odświeżona, ponieważ program nie wykonał się do końca. Miejsce zatrzymania się programu jest sygnalizowane migającym czarnym prostokątem. Aby przejść dalej, należy wcisnąć przycisk *Pause,* co spowoduje wykonanie programu do końca. Jeśli mamy większą liczbę pułapek, program wykona się do następnej. Aby usunąć pułapkę, należy kliknąć na niej prawym przyciskiem myszy i wybierać *Breakpoints -> Clear Breakpoints*.

Podsumowanie

W pierwszej części poznaliśmy podstawowe informacje na temat środowiska oraz sposoby uruchamiania i śledzenia programu. W części drugiej przedstawię najważniejsze instrukcje strukturalne języka programowania graficznego G.

Wiesław Szaj
wszaj@prz.edu.pl