Popularność techniki cyfrowej, a w szczególności mikroprocesorowej, w naturalny sposób wymusiła na elektronikach-konstruktorach potrzebę podjęcia wyzwania nauki programowania. O ile w przypadku języków maszynowych nie stanowiło to większego problemu, o tyle języki wyższego poziomu często nie były przyjmowane z dużym entuzjazmem. Dla początkującego elektronika, wciąż borykającego się problemami natury sprzętowej i niemającego zacięcia programistycznego, nauka języka np. C bywa bolesna. Wraz z nabieraniem doświadczenia w programowaniu systemów wbudowanych, wcześniej czy później zachodzi potrzeba połączenia urządzenia ze światem zewnętrznym. Jeśli będzie to komputer PC z zainstalowanym jednym z popularnych systemów operacyjnych, to co wtedy? Od strony sprzętowej nie ma problemu: elektronik-konstruktor doskonale zna specyfikację połączenia szeregowego czy równoległego; dodatkowo mikrokontroler, który wybrał, jest wyposażony w interfejs USB. Mogło zdarzyć się, że w urządzeniu zaimplementowano interfejs ethernetowy... elektronik-konstruktor dopiero co uporał się z opanowaniem języka C w stopniu pozwalającym na tworzenie aplikacji mikroprocesorowej, a już musi zastanawiać się jak obsłużyć te wszystkie metody komunikacyjne z poziomu aplikacji uruchamianej na komputerze PC. Podejmuje zatem kolejne wyzwanie: nauka programowania w środowisku Windows. Za chwilę pojawia się następne: nauka programowania interfejsów sprzętowych w środowisku Windows... Na końcu okazuje się, że z czasu poświęconego na zbudowanie kompletnego systemu złożonego z urządzenia, jego aplikacji wbudowanej oraz aplikacji służącej do komunikacji za pomocą komputera PC, 80% zajęła elektronikowi nauka programowania. Z definicji jednak elektronik nie jest programistą. Satysfakcję i przyjemność z wykonywania zawodu/hobby daje mu konstruowanie urządzenia. Do programowania (zwłaszcza aplikacji PC) został zmuszony. Sytuacja nie jest jednak beznadziejna, ponieważ cały proces można uprościć, modyfikując ostatni jego etap. Jeśli zamiast konwencjonalnych języków programowania (C++, Java, C#), zastosujemy język G, a środowiskiem deweloperskim będzie LabVIEW. Pomagamy wówczas elektronikowi-konstruktorowi dwojako: po pierwsze, dajemy mu do dyspozycji potężne narzędzie programistyczne, które od podstaw tworzone było z myślą o interakcji ze sprzętem, po drugie, alternatywę w sposobie podejścia do samego procesu programowania, proponując „rysowanie" kodu aplikacji.

Niniejsza praca stanowi próbę pokazania, jak środowisko to może zostać użyte do definitywnego rozwiązania problemu, z którym, w przekonaniu autora, boryka się spora część elektroników.

LabVIEW (**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) jest środowiskiem programistycznym używającym do tworzenia aplikacji graficznego języka programowania (potocznie nazywanym **G**). Pomimo tego, że obecnie LabVIEW może być uruchamiane pod kontrolą wszystkich popularnych systemów operacyjnych (Linux, Windows, Mac OSX), to jego pierwsza wersja (wprowadzona w roku 1986) pracowała jedynie na komputerach Apple Macintosh. Powód był prozaiczny: w tamtym czasie jedynie Apple dysponował graficznym interfejsem użytkownika systemu operacyjnego.

Idea programowania w języku G opiera się na przepływie danych. Nie występują tutaj w jawnej postaci zmienne znane z konwencjonalnych (tekstowych) języków programowania. Kod LV opiera się na komponentach prezentowanych w postaci

symboli graficznych, połączonych ze sobą przewodami. Przewody są odzwierciedleniem ścieżki przepływu danych, która jednocześnie wymusza wykonywanie się aplikacji. Model ten odróżnia LV od konwencjonalnych języków, których to komponenty sterują wykonywaniem się aplikacji. Jedne funkcje/komponenty wywołują kolejne, przekazując (lub nie) do nich dane. W LV komponent/funkcja zostanie wykonana tylko wtedy, kiedy dotrą do niej wszystkie dane wejściowe. Fakt ten jest często przytaczany (przede wszystkim przez „konwencjonalnych" programistów) jako argument na niekorzyść LabVIEW, bowiem że podejście takie nie gwarantuje kontroli nad kolejnością wykonywania się fragmentów programu. Argument ten jest bezpodstawny. Jeśli programista będzie stosował kilka podstawowych zasad kontroli przepływu danych, wówczas nie ma mowy o jakiejkolwiek dowolności wykonywania się kodu. Fakt, iż w kodzie LV nie znajdziemy linijki tekstu, jest również przytaczany jako powodujący nieczytelność czy trudność zrozumienia kodu. Również i ten argument jest nietrafiony. Tak naprawdę do przekazu logicznej informacji nie jest potrzebny tekst. Przykładem niech będą wszelkiego rodzaju algorytmy prezentowane w postaci grafów. W klasycznym podejściu implementacja takiego algorytmu sprowadza się do stworzenia kodu (w postaci tekstu) o treści zrozumiałej dla kompilatora. LV eliminuje krok pośredni: kod w postaci ciągu znaków ASCII. Co więcej, okazuje się, że do łatwego zrozumienia kodu zapisanego w postaci tekstowej nie jest istotna tylko jego treść, ale i wygląd. Zawodowi programiści używający języków tekstowych kładą ogromny nacisk na styl programowania (w sensie formatowania tekstu). Dlaczego? Ponieważ wcięcia i odstępy pomiędzy fragmentami kodu niosą cenną informację na temat jego struktury. W LV to wyłącznie wygląd kodu stanowi o jego treści.

Poza podstawowymi elementami programu, LV dostarcza wiele gotowych komponentów, zwłaszcza w domenach zbierania, analizy, przetwarzania i prezentacji danych, co powoduje, że jest to środowisko niezwykle efektywne. Z drugiej jednak strony często prowadzi to do błędnego stwierdzenia, iż poprawne programowanie w LV nie wymaga dużej wiedzy. W przypadku małych aplikacji często jest to prawdą, jednak aby stworzyć duży i skalowalny kod, potrzebne jest spore doświadczenie. Zorientowanie LV na interakcję ze sprzętem powoduje też częste zaszufladkowanie tego środowiska jako typowo laboratoryjnego czy testowego. Rodzi to dyskusję, czy G jest językiem ogólnego przeznaczenia? Poprawna odpowiedź mogłaby brzmieć: nie, G jest językiem, za pomocą, którego można stworzyć aplikację ogólnego przeznaczenia, nie jest jednak ogólnym językiem programowania. Wynika to również z faktu, że jedynym dostawcą kompilatora G jest jego twórca, a więc National Instruments. G nie jest też objęte standardami ANSI (co ma miejsce np. w przypadku C). Przeciwko klasyfikacji G, jako równoważnego z konwencjonalnymi językami programowania, przemawia brak możliwości stosowania rekurencji. Komponent (VI) stworzony w języku G nie może wywołać samego siebie. Z teoretycznego punktu widzenia jest to poważna wada, jednak w praktyce rekurencja w LV może być realizowana w postaci maszyny stanów, w której specyficzny stan może zapisać do kolejki jako akcję wykonanie siebie samego.

Dużą zaletą LabVIEW jest natomiast fakt, iż fragmenty kodu są kompilowane w czasie rzeczywistym podczas tworzenia aplikacji. Sprawia to, że nie występuje tu potrzeba kompilacji kodu przed rozpoczęciem jego debugowania/uruchamiania. Implikuje to sytuację, gdzie kod źródłowy programu jest łączony z elementem wykonywalnym do pojedynczego pliku z rozszerzeniem *.vi. Plik wykonywalny może

zostać uruchomiony za pomocą środowiska uruchomieniowego (ang. *Run-Time Engine*), które dostarcza aplikacji standardowe mechanizmy systemu operacyjnego. Z tego też powodu środowisko uruchomieniowe jest inne dla każdego systemu operacyjnego, natomiast kod VI jest zawsze taki sam. VI może być uruchamiany na maszynach, na których jest zainstalowane środowisko LabVIEW. Istnieje też możliwość przekształcenia go do samodzielnej aplikacji przy użyciu komponentu *Application Builder*, który np. w przypadku systemu Windows przekształci plik .vi do postaci wykonywalnego pliku .exe. Aplikacja taka nie jest jednak do końca samodzielna, jako że wciąż potrzebuje do pracy środowiska uruchomieniowego. Jednak w tym przypadku nie musi to już być całe LV, a jedynie *Run-Time Engine*, który jest darmowym komponentem udostępnianym przez NI.

Fakt, że LabVIEW oraz programowanie w języku G nie jest trudne do opanowania przez osoby niebędące programistami, nie oznacza, że środowisko nie ma niczego do zaoferowania doświadczonym (w konwencjonalnym programowaniu) programistom. LV ma wiele interfejsów programowych pozwalających na jego interakcję z innymi środowiskami deweloperskimi. Za pomocą LV możemy korzystać z metod/funkcji zgromadzanych w bibliotekach dołączanych dynamicznie, mamy możliwość umieszczania kontenerów *ActiveX* na panelu frontowym, możemy korzystać z prekompilowanych fragmentów kodu napisanego w konwencjonalnym języku programowania czy wreszcie tworzyć instancje obiektów .NET. Ponadto samo środowisko dostarcza wielu mechanizmów kontroli swojej pracy dostępnych wprost z kodu aplikacji.

Domeną LabVIEW nie są jedynie platformy PC. National Instruments dostarcza wielu dodatkowych modułów, które pozwalają na uruchamianie programów tworzonych w G na dedykowanych platformach. W roku 1999 NI zaprezentował odmianę środowiska LabVIEW nazwaną *Real-Time*. Jak nietrudno się domyślić, jest to wersja LabVIEW wspierająca kontrolery czasu rzeczywistego. Generalnie przeznaczona jest ona do współpracy ze sprzętem NI, jak np. kontrolery Compact FieldPoint (można by je porównać do bardzo rozbudowanych sterowników PLC) czy moduły CompactRIO (stanowiące klasę samą dla siebie) – bardzo szybkie wydajne kontrolery mające unikatową architekturę. Pierwsza wersja tego modułu była tak naprawdę niezależna od bazowej instalacji LabVIEW. W systemie instalowała się i była uruchamiana jako osobne środowisko (co było nieco uciążliwe). Obecnie jest to jedynie moduł instalowany jako dodatek do bazowej kopii LabVIEW, umożliwia on również zastosowanie komputera PC w charakterze kontrolera czasu rzeczywistego (rzecz jasna jest to inny komputer niż ten, na którym tworzymy aplikację) poprzez instalację na nim własnego systemu operacyjny (podobny do tego znajdującego się w kontrolerach NI), w którym mogą być uruchamiane aplikacje tworzone za pomocą modułu. Moduł ten jest dostępny wyłącznie dla wersji LabVIEW pracujących pod kontrolą systemów Windows 2000 i XP. Obecnie w zastosowaniach przemysłowych jest to najważniejsze rozszerzenie LabVIEW. Wraz z wersją „7 Express" LabVIEW (nazwa ta obejmuje wersje 7.0 i 7.1), zaprezentowano dwa nowe moduły: LabVIEW PDA Module oraz LabVIEW FPGA Module. Pierwszy z nich umożliwia tworzenie aplikacji uruchamianych na komputerach PDA (ang. *Personal Digital Assistant*) pracujących pod kontrolą takich systemów jak Palm OS, Windows CE czy Windows Mobile for Pocket PC 2003. Sposób rozwijania samej aplikacji jest nieco inny niż w przypadku systemów LV RT (gdzie w trybie de-

bugowania można załadować program do platformy docelowej i kontrolować jego działanie na komputerze nadrzędnym) i polega na testowaniu tworzonego kodu na symulatorze urządzenia uruchamianym w środowisku Windows (podobnie jak LV RT moduł dostępny jest wyłącznie dla systemów Win 2000 i XP). Jeśli chcemy przetestować aplikację na rzeczywistym PDA, to najpierw musimy ją poddać procesowi kompilacji na daną platformę, a następnie przekopiować na komputer przenośny. W tym celu LV współpracuje, np. z oprogramowaniem *ActiveSync*. Interesujące jest to, że w przypadku PDA aplikacje kompilują się do postaci natywnej, a więc mogą być uruchamiane bez kontroli systemu uruchomieniowego (ang. *Run-Time Engine*), jak to ma miejsce w przypadku „dużych" systemów operacyjnych. Naturalnie LabVIEW PDA Module wspiera technologię ekranów dotykowych. NI ma w swojej ofercie również kilka urządzeń DAQ (ang. *data acquisition*), które mogą współpracować z komputerami PDA. Są one wyposażone w interfejs USB lub mają postać karty rozszerzeń w formacie *CompactFlash* oraz *PCMCIA*. Pewną ciekawostką (być może odstraszającą potencjalnych użytkowników tego modułu) jest system licencjonowania polegający na wykupywaniu pakietów 10, 100 bądź nieskończonej liczby licencji, która pozwala na komercyjne stosownie oprogramowania stworzonego za pomocą modułu na pojedynczym urządzeniu PDA.

Drugim z modułów zaprezentowanych z siódmą wersją LabVIEW jest LV FPGA Module. Ponownie nazwa wiele wyjaśnia, ale zapewne rodzi jeszcze więcej pytań. Moduł ten po zainstalowaniu umożliwia tworzenie aplikacji z wykorzystaniem układów FPGA (NI wykorzystuje układy Xilinx). W pierwszej jego wersji moduł wspierał jedynie sprzęt w postaci kart PCI bądź PXI, które docelowo pracować miały w komputerach PC lub komputerach przemysłowych NI (zarówno czasu rzeczywistego, jak i pracujących pod kontrolą Windows). Obecnie moduł przeznaczony jest także dla kontrolerów CompactRIO. Moduł ten ma zastosowanie wszędzie tam, gdzie istnieje potrzeba stworzenia indywidualnego i unikalnego hardware'u. NI dostarcza bazową platformę wyposażoną w bloki analogowe oraz cyfrowe, jak również nadzorujący ich pracę układ FPGA. Za pomocą modułu możemy dokonać syntezy kodu, który będzie zapewniał synchronizację pracy modułów, implementację różnego rodzaju magistral czy złożone systemy wyzwalania pomiaru. Tak przygotowany kod zostaje następnie załadowany do pamięci flash platformy docelowej, którą możemy obsługiwać z poziomu LV. Niestety nie ma możliwości syntezy kodu bezpośrednio na jakikolwiek zewnętrzny układ PLD. Moduł ten dodaje jedynie elastyczność konfiguracji dla urządzeń akwizycji danych NI (z rodziny RIO – *Reconfigurable IO*). Z uwagi na to, że synteza kodu jest procesem długotrwałym, w trakcie rozwijania aplikacji stosuje się, podobnie jak w przypadku PDA, symulator.

Przy pojawieniu się wersji 8.20 LabVIEW, NI zaprezentował kolejny moduł przeznaczony tym razem dla zintegrowanych komputerów panelowych z dotykowym wyświetlaczem. *LabVIEW Touch Panel Module* pozwala na uruchamianie aplikacji tworzonych w LV na przemysłowych panelach pracujących pod kontrolą systemu *WindowsCE*. Przebieg rozwoju aplikacji wygląda podobnie jak przy używaniu modułu PDA. Zastosowano też podobne zasady licencjonowania powstałego za jego pomocą oprogramowania, z tym że licencję nabywa się tutaj pojedynczo, dla każdej platformy sprzętowej niepochodzącej z NI.

Bardziej interesujące z punktu widzenia elektronika moduły LV to trzy pakiety należące do rodziny *LabVIEW for Embedded Applications*: *LabVIEW DSP Module, LabVIEW Embedded Development Module oraz LabVIEW Embedded Module for ADI Blackfin.*

Pierwszy z nich przeznaczony jest dla trzech gotowych platform zawierających procesory sygnałowe *Texas Instruments*. Wspierane są układy serii C6000, a więc rodzina procesorów zmiennoprzecinkowych. Moduł tuż po instalacji umożliwia pracę z dwoma płytami DSK TI (DSP Starter Kit) oraz z jedną platformą NI o nazwie NI SPEEDY-33. Podstawowe parametry tych platform przedstawiono w **tabeli 1.1**. *LV DSP Module* dostarcza naturalnie szeregu gotowych funkcji służących do przetwarzania sygnałów. Znaleźć można funkcje transformat pomiędzy dziedziną czasową i częstotliwościową. Znajdują się w nim funkcje okien czasowych, funkcje splotu i obliczania korelacji sygnałów, funkcje opóźnień, funkcje obliczania wartości skutecznej sygnału oraz ogromny zasób funkcji służących do generowania sygnałów. Ponadto NI dostarcza specjalny pakiet oprogramowania (tzw. *toolkit*) służący do aplikowania przeróżnych filtrów cyfrowych, w tym filtrów adaptacyjnych. Poza dziedziną cyfrowego przetwarzania sygnałów moduł ma funkcję bezpośredniego dostępu do linii IO układów znajdujących się na płytach.

Moduł DSP stanowił pojedynczą próbę usadowienia LV na platformach stricte wbudowanych. Pomimo tego, że wspierał jedynie trzy płyty, to istniała teoretyczna możliwość zaprojektowania własnego urządzenia opartego na procesorze DSP rodziny C6000 i zaprogramowania go omawianym modułem. Prawdziwy przełom w tej kwestii stanowi jednak najnowszy moduł dla aplikacji wbudowanych nazwany *LabVIEW Embedded Development Module*. Pozwala on na uruchamianie kodu stworzonego w języku G teoretycznie na każdym procesorze 32-bitowym. Hasło promujące moduł brzmiało: „*Allow Embedded Developers to port LabVIEW to a platform that NI has never seen*" (w wolnym tłumaczeniu: „umożliwić programistom portowanie aplikacji na platformy, jakich NI nigdy nie widział na oczy", natomiast między wierszami: „uwolnić programistów LabVIEW od sprzętu NI"). Z jednej strony można powiedzieć, że NI podcina sobie skrzydła – inżynierowie programujący w LV nie będą więcej nabywać drogich kontrolerów NI, np. CompactRIO, aby zbudować system RT, mogą przecież zaprojektować własny kontroler oparty na tanich płytkach rozwojowych. Po głębszym zastanowieniu wydaje się jednak, że tak naprawdę intencje są zupełnie inne: programiści LabVIEW mogą zacząć promować ten sposób tworzenia kodu także dla platform, dla których jedyny zrozumiały język to C. Jest to zatem zabieg mający na celu popularyzację języka G we wszystkich obszarach systemów wbudowanych. Moduł abstrahuje od platformy sprzętowej tym, że wynikiem jego pracy jest czysty kod C zgodny ze standardem ANSI, wygenero-

Tab. 1.1. Podstawowe parametry płyt ewaluacyjnych wspieranych przez *LV DSP Module*

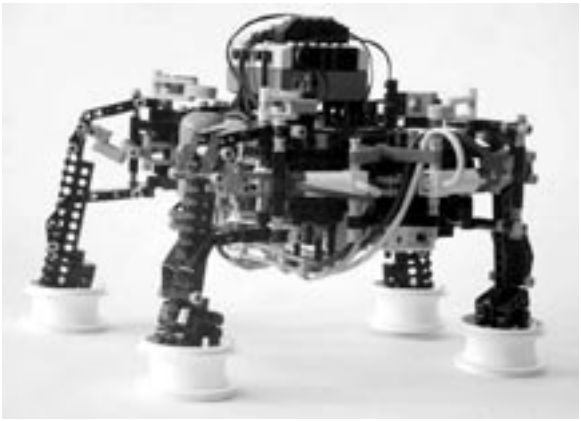
Płyta	TIC6711 DSK	TIC6713 DSK	NI SPEEDY-33
Rozdzielczość kodeka audio	16-bit	24-bit	16-bit
Liczba kanałów	Mono	Stereo	Stereo
Częstotliwość próbkowania	8 kHz	96 kHz	48 kHz
Liniowe wej/wyj	Tak	Tak	Tak
Wejścia cyfrowe	4 DIP switch	4 DIP switch	8 DIP switch
Wyjścia cyfrowe	4 LED	4 LED	8 LED

wany z VI stworzonych przez programistę. I tu w zasadzie kończy się działalność modułu. Dysponując takim kodem oraz kompilatorem C na daną platformę sprzętową, możemy dokonać procesu kompilacji i linkowania, a następnie uruchamiać do woli naszą aplikację. W teorii wygląda to dość prosto. W praktyce sprawy się nieco komplikują i moduł do poprawnie działającego „portu" na dany procesor/platformę wymaga kilku dni pracy. Praca ta to przede wszystkim tworzenie skryptów (jednak w języku G), które umożliwią automatyczne wywoływanie narzędzi (kompilatora, linkera asemblera i narzędzia do ładowania aplikacji do pamięci płyty). Czyli jest to po prostu integracja mająca na celu stworzenie sprawnie działającego środowiska. *LV Embedded Development Module* jest dostarczany z przykładowymi portami na kilka typów procesorów: między innymi PowerPC, oraz na dwa systemy operacyjne: VxWorks i eCos. Ma również szerokie możliwości w zakresie debugowania aplikacji, w tym (podobnie jak moduł DSP) z uaktualnianiem paneli czołowych aplikacji w czasie rzeczywistym. Tak jak pierwsza wersja *LabVIEW Real-Time*, moduł ten jest dostarczany z osobną instalacją LabVIEW.

NI przygotował też pakiet dla inżynierów, którzy muszą rozpocząć pracę od razu, nie mając czasu na integrację i dogrywanie narzędzi. Jest to *LabVIEW Embedded Module for ADI Blackfin Processors*, w skrócie – gotowy port *LV Embedded Development Module* właśnie dla procesorów *Blackfin*. Do kompilacji generowanego kodu C użyto dostarczanego przez Analog Devices – VisualDSP++, natomiast sam kod aplikacji pracuje pod kontrolą systemu operacyjnego *VisualDSP++ Kernel*. Dodatkowo Moduł ten zawiera przebudowane biblioteki analizy sygnałów (co nie ma miejsca w podstawowym module LV EDM, gdzie biblioteki musimy przebudowywać samodzielnie, każdorazowo gdy zmieniamy platformę bądź kompilator C). Od strony sprzętowej moduł współpracuje z platformą Analog Devices 573 EZ-KIT.

Oprócz „poważnych" zastosowań, LabVIEW odnalazło się także w dziedzinie rozrywkowej robotyki. Seria klocków LEGO MINDSTORMS umożliwia konstruowania inteligentnych zabawek-robotów. Część mechaniczną rozwiązano stosując klocki zbliżone do lego (nieco bardziej skomplikowane). Każdy robot ma na „pokładzie" jednostkę sterującą nazwaną NXT (opartą na 32-bitowym procesorze!). Kontroler ten umożliwia sterownie modułami ruchu, odczytywanie wskazań czujników czy różnorodną sygnalizację. Program wykonywany w tej jednostce jest tworzony w środowisku opartym na... LabVIEW, a następnie ładowanym do kontrolera poprzez połączenie *Bluetooth* bądź USB. Pakiet *LabVIEW Toolkit for LEGO MINDSTORMS NXP* jest udostępniany za darmo. Oprogramowanie to może pracować pod kontrolą systemów Windows XP oraz Mac OS.

Niniejsza książka ma na celu wyjaśnienie podstaw programowania w języku G. W szczególności jednak nacisk położono na zagadnienia obsługi popularnych interfejsów komunikacyjnych. Zaprezentowano wiele przykładów prostych aplikacji i bardziej złożonych programów pokazujących z jednej strony pryncypia obsługi interfejsów, z drugiej natomiast zastosowanie tej wiedzy do tworzenia samodzielnych aplikacji. Co będzie nam potrzebne, aby efektywnie analizować przytaczane przykłady kodu? Podstawowym oprogramowaniem, w jakie należałoby się zaopatrzyć, chcąc myśleć poważnie o nauce programowania bądź w ogóle o używaniu LabVIEW, jest LabVIEW for: Windows, Linux, MacOS (naturalnie są to trzy od-



Rys. 1.1. Robot kroczący zbudowany za pomocą zestawu LEGO MINDSTORMS

rębne pakiety oprogramowania). Nazwa mówi sama za siebie – są to środowiska programowania przeznaczone dla konkretnych systemów operacyjnych.

National Instruments oferuje trzy konfiguracje podstawowego środowiska LabVIEW, różniące się między sobą liczbą dodatkowych komponentów. Najbardziej podstawowy pakiet nosi nazwę *LabVIEW Base Development System for Windows*. Obecna wersja nosi numer 8.5. Pakiet ten pozwala na tworzenie aplikacji i uruchamianie jej w środowisku. Nie umożliwia natomiast budowania programów do wykonywalnych plików .exe. Chcąc kompilować samodzielne aplikacje należy do tego pakietu dołączyć *Application Builder for Windows* (nabywany osobno). Ta wersja LabVIEW jest najtańsza i dostępna jedynie dla systemu Windows. Pakiet ten wydaje się być idealny do celów szkoleniowych, bądź jako wyposażenie laboratorium.

Bardziej rozbudowana jest wersja nazwana LabVIEW *Professional Development System*. Pakiet ten jest przeznaczony do profesjonalnych zastosowań (czytaj: komercyjnych). Standardowo zawiera *Application Builder* oraz dodatkowe biblioteki dostarczające szeregu funkcji do przetwarzania oraz analiz sygnałów oraz do ogólnych zastosowań matematycznych. Zawiera również narzędzia wspierające tworzenie skomplikowanych programów sprawdzających wydajność i integralność kodu (nabierających znaczenia właśnie podczas kompilowania samodzielnych aplikacji). Ta wersja jest dostępna w jednakowej postaci dla wszystkich trzech podstawowych systemów operacyjnych: Windows, MacOS, Linux. Pakiet ten występuje też pod nazwą *LabVIEW Full Development System* i różni się (poza ceną) tym, że nie ma standardowo Application Buildera oraz wspierających go narzędzi. Można powiedzieć, iż powyższe pakiety dla komputerów osobistych stanowią o charakterze LabVIEW, które powinno być kojarzone właśnie z platformami PC. W każdym przypadku dodatkowego modułu czy jakiegokolwiek rozszerzenia i tak wszystko sprowadza się do tworzenia aplikacji na komputerze stacjonarnym, a dopiero w kolejnym kroku jej kompilacji na docelową platformę. Z tego też względu, aby używać jakiejkolwiek wersji LabVIEW (na jakąkolwiek platformę), musimy posiadać jeden z pakietów omówionych wyżej.

Wszystkie przykłady przytaczane w dalszej części napisano i przetestowano z użyciem LabVIEW w wersji 7.1 uruchamianym na komputerze z systemem operacyjnym Windows XP Professional.