Piotr Kaczorek, Marcin Waraksa
Akademia Morska w Gdyni

WDROŻENIE RADIA PROGRAMOWALNEGO Z WYKORZYSTANIEM GNU RADIO ORAZ USRP2920

Technika radia programowalnego jest coraz szerzej stosowana do celów dydaktycznych, badawczych i komercyjnych. W pracy przedstawiono pakiet GNU Radio i urządzenie USRP2920, które razem tworzą uniwersalną platformę radia programowalnego.

Słowa kluczowe: *radio programowalne, GNU Radio, USRP.*

WSTĘP

Technika radia programowalnego SDR (*Software Defined Radio*) polega *de facto* na takim przesunięciu punktu cyfrowej obróbki sygnału, aby znajdował się on możliwe najbliżej anteny – sprowadza zatem problem sprzętowego odbioru sygnału radiowego do problemu programistycznego. Przetwarzanie sygnałów w SDR istotnie różni się od klasycznego radia, w którym radiowe tory nadawczy i odbiorczy są realizowane w postaci układów analogowych bądź hybrydowych analogowo-cyfrowych. Domeną SDR jest zarówno formowanie sygnału wyjściowego w nadajniku, jak i jego późniejsza detekcja w odbiorniku, niemal wyłącznie za pomocą oprogramowania. Ze względu na ograniczenia technologiczne przetworników cyfrowo-analogowych, radio programowalne musi współpracować z modułem sprzętowym, którego zadaniem jest przeniesienie uformowanego już sygnału na odpowiednią częstotliwość radiową (w odbiorniku odwrotnie) i jego wzmocnienie. SDR otwiera nowe możliwości implementacji toru radiowego, niedostępne przy sprzętowej realizacji układów nadajnika i odbiornika. Technika ta jest coraz szerzej stosowana, zarówno do celów dydaktycznych i badawczych, jak i komercyjnych do budowy systemów komunikacji radiowej.

W dalszej części artykułu przedstawiono pakiet narzędzi GNU Radio, który w połączeniu z urządzeniami USRP firmy National Instruments tworzy uniwersalną platformę radia programowalnego.

1. GNU RADIO

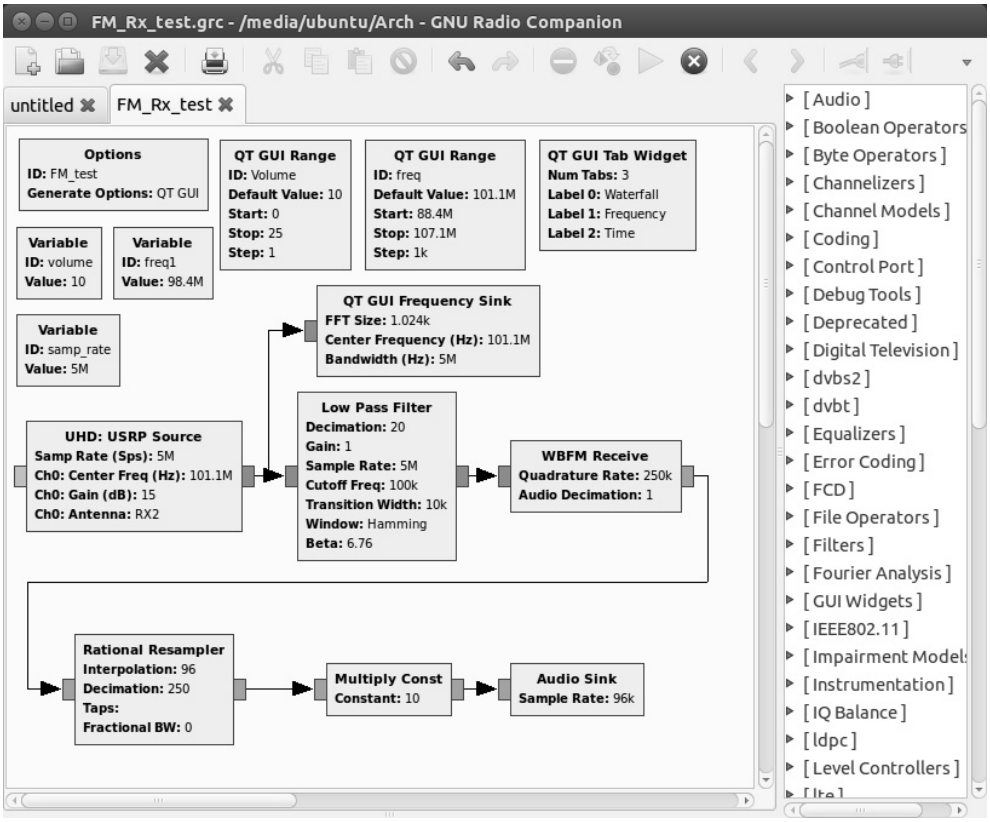
GNU Radio jest zestawem narzędzi programistycznych zorientowanych na tworzenie oprogramowania do cyfrowego przetwarzania sygnałów, a w połączeniu z odpowiednim modułem sprzętowym – do implementacji SDR. Zawiera wiele gotowych modułów (bloków), które realizują określone algorytmy cyfrowego przetwarzania sygnałów, np. filtrację, modulację i demodulację, synchronizację, kodowanie źródłowe i kanałowe oraz wiele innych. Co więcej, istnieje możliwość łączenia tych bloków i w konsekwencji przekazywania danych z jednego bloku do następnego w taki sposób, jak zwykle ma to miejsce w nadajniku i odbiorniku radiowym. GNU Radio umożliwia stworzenie aplikacji, która pobiera dane z jednego strumienia, przetwarza je i następnie umieszcza w innym strumieniu. Jeżeli dane są odczytywane i zapisywane do plików, to uzyskuje się rodzaj środowiska do prowadzenia symulacji, jednak głównym przeznaczeniem GNU Radio jest, jak już wspomniano, implementacja SDR. Niezbędny do tego jest kompatybilny moduł sprzętowy, przekształcający strumień danych na sygnał analogowy i przenoszący go na właściwą częstotliwość radiową [6].

GNU Radio jest oficjalnie częścią GNU Project, jest więc oprogramowaniem *open source*, rozprowadzanym na licencji GNU GPL. Pierwszym kierownikiem projektu był Eric Bossom, a pierwsze wydanie GNU Radio miało miejsce w roku 2001. Jako jeden z pierwszych do projektu GNU Radio dołączył Matt Ettus, który opracował uniwersalną platformę sprzętową do GNU Radio, USRP. Urządzenia te produkuje firma Ettus Research, obecnie należąca do National Instruments.

2. GNU RADIO COMPANION

GNU Radio Companion (GRC) jest jednym z najważniejszych narzędzi należących do pakietu GNU Radio. Jest to program z graficznym interfejsem użytkownika, służący do projektowania diagramów przepływu sygnałów (*signal flow graphs*), reprezentujących sposób przetwarzania sygnałów przez GNU Radio. Korzystając z GRC, użytkownik umieszcza na diagramie wybrane bloki i łączy je, wskazując w ten sposób kolejność przetwarzania danych. Większość bloków ma zdefiniowane parametry, których wartość można zmieniać, dopasowując je do aktualnych potrzeb [3].

Okno programu GRC oraz przykładowy diagram przepływu sygnałów przedstawiono na rysunku 1.



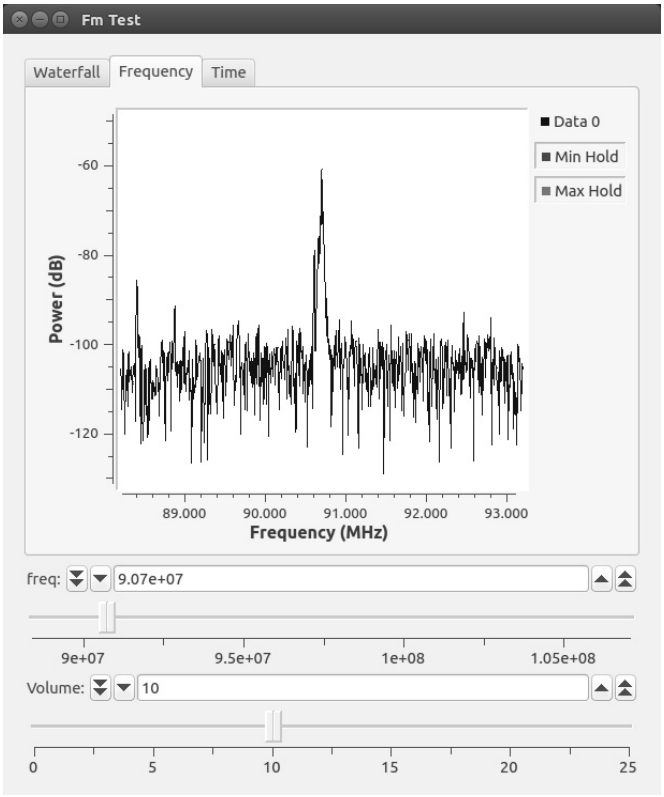
Rys. 1. Program GRC z przykładowym diagramem przepływu sygnałów

Fig. 1. *An example of the signal flow graph in GRC application program*

Pierwotnie program GRC jedynie zapisywał opracowany przez użytkownika diagram w pliku XML, natomiast do realizacji przetwarzania sygnałów według tego diagramu służyła inna aplikacja. Obecnie GRC, wykorzystując szablony języka Python (*Cheetach templates*), generuje kod źródłowy Python dla diagramu, dzięki czemu diagram zostaje przekształcony w samodzielną aplikację [3].

GNU Radio Companion może generować kod źródłowy aplikacji wyposażonych w graficzny interfejs użytkownika, do czego służą bloki tzw. widżetów, które można umieszczać na diagramach. Widżety mogą służyć do wizualizacji przetwarzanych sygnałów (np. wyświetlania przebiegu lub widma sygnału) oraz do dynamicznej zmiany parametrów implementowanego układu (np. zmiany częstotliwości nośnej za pomocą suwaka).

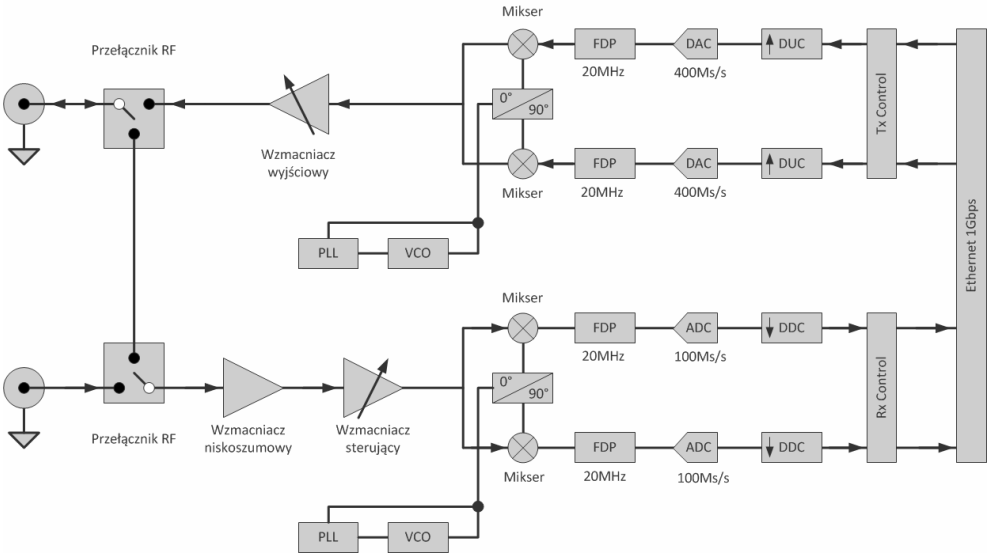
Przykład aplikacji utworzonej za pomocą GRC przedstawia rysunek 2. W tym przypadku jest to odbiornik radia FM, którego diagram sygnałowy został pokazany na rysunku 1. W oknie aplikacji (rys. 2) są widoczne wspomniane widżety – wykres widma odbieranego sygnału oraz suwak zmiany częstotliwości nośnej.



Rys. 2. Okno aplikacji utworzonej za pomocą GRC
***Fig. 2**. Window of application program created within GRC*

3. PLATFORMA USRP

USRP (*Universal Software Radio Peripheral*) jest uniwersalną, konfigurowalną programowo platformą sprzętową radia programowalnego, którą można wykorzystać zarówno do zadań badawczych, jak i do celów dydaktycznych. W połączeniu z oprogramowaniem SDR umożliwia efektywne prototypowanie nadajników i odbiorników radiowych. Dzięki otwartej architekturze możliwe jest dostosowanie USRP do wymagań związanych z konkretnym zadaniem projektowym lub obszarem badań. Dwa tory radiowe, RX1/TX1 oraz RX2 (rys. 1), pozwalają na opracowanie oraz walidację nadajnika i odbiornika z wykorzystaniem jednego tylko urządzenia USRP, po połączeniu wyjścia toru nadawczego TX1 z torem odbiorczym RX2 przez tłumik 30 dB. Dzięki modułowej budowie USRP może pracować w kilku zakresach częstotliwościowych (tab. 1), zależnie od zainstalowanego tranceivera [8].



Rys. 3 Architektura USRP NI2920 [8]
Fig. 3. *The USRP NI2920 architecture[8]*

Platforma USRP jest dostarczana z zainstalowanym oprogramowaniem *firmware* oraz sterownikami dedykowanymi dla środowiska LabVIEW. Aby zapewnić kompatybilność z GNU Radio, należy zainstalować *firmware* oraz obraz FPGA zgodne z używaną dystrybucją GNU Radio, a także kompatybilną wersję sterownika UHD (*USRP Hardware Driver*).

Tabela 1. Zakresy pracy USRP (w zależności od zainstalowanego tzw. frontendu)
Table 1. *USRP frequency bands (depending on RF front-end installed)*

RF Frontend	Pasmo [MHz]	Rx	Tx	Uwagi
Basic Tx	1 – 250		+	
Basic Rx	1 – 250	+		
LFTX	0 – 30		+	
LFRX	0 – 30	+		
TVRX2	50 – 860	+		
DBSRX2	800 – 2300	+		
WBX	50 – 2200	+	+	B = 40 MHz
SBX	400 – 4400	+	+	B = 40 MHz
CBX	1200 – 6000	+	+	B = 40 MHz
UBX40	10 – 6000	+	+	B = 40 MHz

Oznaczenia: + – funkcjonalność dostępna, B – szerokość pasma roboczego.

Pakiet UHD zawiera zarówno sterownik dla platformy USRP, jak i dedykowany interfejs API (ang. *Application Programming Interface*) do GNU Radio. Do instalacji oprogramowania *firmware* służy dostarczany z USRP program narzędziowy, który należy uruchomić, wpisując w oknie konsoli polecenie:

usrp_n2xx_simple_net_burner.exe --addr=x.x.x.x
--fw=..\usrp_n210_fw.bin --fpga=..\usrp_n210_r4_fpga.bin

przy czym opcja *--addr* służy do wskazania adresu IP urządzenia USRP, natomiast opcje *--fw* oraz *--fpga* pozwalają wskazać nazwy plików *firmware* i obrazu FPGA. Po aktualizacji oprogramowania platforma USRP jest kompatybilna ze sterownikiem UHD, dzięki czemu możliwa jest współpraca USRP z GNU Radio.

4. PRZEGLĄD BLOKÓW GNU RADIO

GNU Radio posiada imponującą liczbę kilkuset bloków, podzielonych na ponad 50 kategorii (lub modułów, jako że każda z kategorii jest związana z odrębnym modułem języka C++ lub Python). Poszczególne bloki mogą bardzo różnić się złożonością realizowanych funkcji: najprostsze realizują elementarne operacje matematyczne lub logiczne, te zaś najbardziej złożone pełnią rolę funkcjonalnych bloków zaawansowanych systemów radiokomunikacyjnych, takich jak telewizja cyfrowa DVBT lub system LTE [2].

Celem niniejszej pracy nie jest przedstawienie czy choćby wymienienie wszystkich bloków GNU Radio, warto jednak dokonać krótkiego ich przeglądu, ponieważ daje to lepsze wyobrażenie o możliwych zastosowaniach GNU Radio, zarówno w pracach naukowo-badawczych, jak i do celów dydaktycznych [2]:

- **Operatory.** Kilka kategorii bloków, tj. *Bool Operators, Byte Operators, Math Operators, Packet Operators, Stream Operators* oraz *File Operators*, realizuje różne przekształcenia danych, od najprostszych (np. blok AND, realizujący logiczną operację koniunkcji, podobnie ADD – dodawanie, *Integrate* – całkowanie), po bardziej złożone, np. w kategorii *Packet Operators* można znaleźć bloki realizujące enkapsulację i dekapsulację pakietów danych. Kategoria operatorów plikowych *File Operators* umożliwia m.in. zapisywanie do plików dowolnych danych przetwarzanych w GRC. Dane te mogą być następnie odczytane i przetwarzane przez GNU Radio albo wykorzystane do analizy za pomocą innych narzędzi.
- **Cyfrowe przetwarzanie sygnałów.** Kategorie *WaveformGenerators, Filters, Resamplers, Synchronizers, Fourier Analysis* oraz *Spectrum Estimation* zawierają szereg bloków realizujących różne algorytmy DSP ogólnego przeznaczenia, które mogą być wykorzystane jako elementy składowe modeli systemów radiokomunikacyjnych, ale też do analizy sygnałów, również w celach dydaktycznych.
- **Kodowanie źródłowe.** Kategoria *Audio* zawiera kodery i dekodery PCM (g711 a-Law oraz u-Law) g723.24, g.723.40, GSM FR, CVSD (odmiana różnicowego PCM o niewielkiej przepływności) oraz CODEC2 (opracowany na

potrzeby radiokomunikacji amatorskiej koder MBE o zasadzie działania zbliżonej do kodera AMBE2+, stosowanego w systemach DMR i dPMR). W kategorii *Audio* znalazły się również bloki *WaveFileSource* oraz *WaveFileSink*, dzięki którym można odczytywać i zapisywać pliki audio w różnych formatach.

- **Kodowanie kanałowe i modulacja.** Kategorie *Error Coding* zawierają bloki koderów i dekoderów różnych kodów korekcyjnych, m.in. kodu splotowego CC (*Convolutional Code*), turbokodu PCCC (*Parallel Concatenated Convolutional Code*) oraz kodu SCCC (*Serial Concatenated Convolutional Code*), który jest proponowany dla systemów telewizji satelitarnej drugiej generacji DVB-S2. Kategoria *Modulators* zawiera aż 30 bloków modulatorów i demodulatorów, implementujących wiele popularnych modulacji, zarówno analogowych (AM i FM), jak i cyfrowych (PSK, DPSK, GFSK, GMSK oraz QAM). Odrębna kategoria, *OFDM,* dostarcza szereg bloków potrzebnych do realizacji tego typu transmisji – oprócz bloków modulatora i demodulatora są m.in. bloki dodawania cyklicznego przedrostka, estymacji parametrów kanału oraz synchronizacji.
- **Modele kanałów.** Kategoria *Channel Models* zawiera kilka bloków modeli kanałów, które można wykorzystać zarówno jako modele kanału AWGN, jak i kanału z zanikami wielodrogowymi Rayleigha, dla kanału o profilu NLOS (*non line-of-sight*), oraz Rice'a, dla kanału LOS (*line-of-sight*). Profil kanału (AWGN, NLOS, LOS) można zmieniać przez parametry konfiguracyjne bloku, podobnie jak opóźnienia i względną moc dla poszczególnych dróg propagacji, dzięki czemu można np. uzyskać modele zgodne ze specyfikacją systemu GSM.
- **Elementy GUI.** GNU Radio posiada dwa rodzaje bloków graficznego interfejsu użytkownika – *QT* oraz *WX*, wykorzystujące dwie różne biblioteki klas, tj. *QT* oraz *wxWidgets*. Obie biblioteki są przenośne i dostępne dla różnych systemów operacyjnych. W diagramie budowanym w GRC nie można mieszać ze sobą widżetów należących do różnych bibliotek. Widżety (kategoria *GUI Widgets*) to m.in. *GUI Entry* – pole tekstowe, w którym użytkownik może wpisać wartość jakiegoś parametru, oraz *GUI Range* – suwak, za pomocą którego wartość parametru można ustawić, używając myszki. Inna kategoria, *Instrumentation*, zawiera kilka bloków do graficznej prezentacji sygnałów przetwarzanych w GNU Radio. Są to m.in. wykres wartości sygnału (*GUI Time Sink*), jego widma (*GUI Frequency Sink*), wykres konstelacji (*GUI Constellation Sink*) i inne.
- **DVBT, DVBS2, WiFi, LTE.** GNU Radio zawiera również kilka kategorii bloków realizujących wybrane funkcje kilku systemów radiokomunikacyjnych. Sama tylko kategoria LTE zawiera prawie 40 bloków implementujących funkcje specyficzne dla tego systemu.

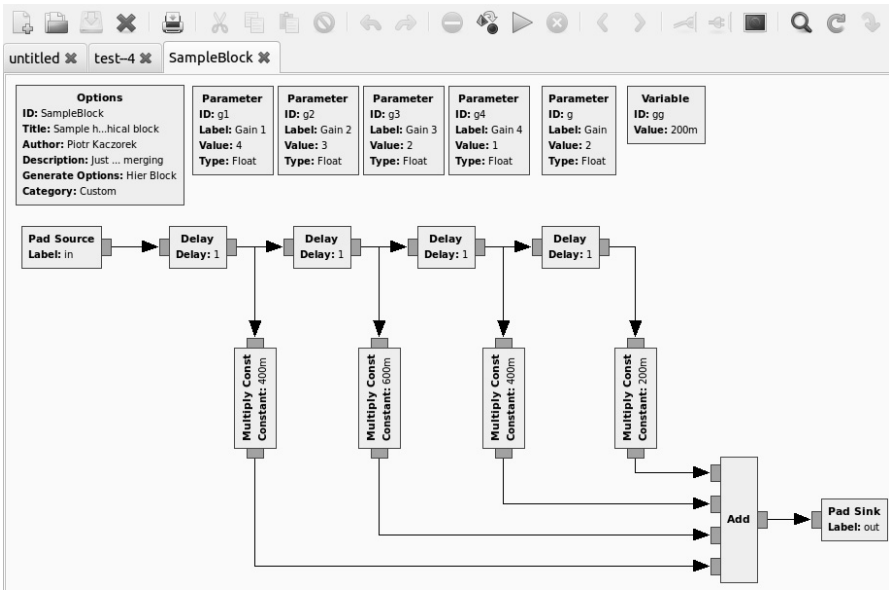
Reasumując, warto wspomnieć, że istnieje kilka ciekawych projektów opartych na GNU Radio, implementujących nawet bardzo złożone systemy radiokomunikacyjne. Do najciekawszych należą *OpenBTS* – implementacja systemów GSM i UMTS [7] oraz *Amarisoft LTE eNodeB* – implementacja LTE eNodeB [1]. Oba projekty są typu *open source*, można zatem pobrać i wykorzystać gotowe moduły GNU Radio wraz z kodem źródłowym.

5. TWORZENIE BLOKÓW GNU RADIO

Istnieją dwie metody tworzenia bloków GNU Radio. Pierwsza polega na połączeniu istniejących już bloków w tzw. blok hierarchiczny (*hierarchical block*), druga, nazywana w GNU Radio *out-of-tree module*, wymaga utworzenia modułu w języku C++ lub Python, który po skompilowaniu może być dołączony do GNU Radio [6]. Obie metody są dobrze znane z popularnych narzędzi do symulacji i analizy sygnałów – pakietu Simulink do programu Matlab oraz programu LabVIEW. Możliwości GNU Radio w tym zakresie są, co prawda, znacznie skromniejsze, jednak są warte zainteresowania, ponieważ umożliwiają dodanie do GNU Radio dowolnych funkcji cyfrowego przetwarzania sygnałów, dzięki czemu można wykorzystać GNU Radio do implementacji, badania i wdrażania praktycznie dowolnych systemów radiokomunikacyjnych.

5.1. Tworzenie bloków hierarchicznych

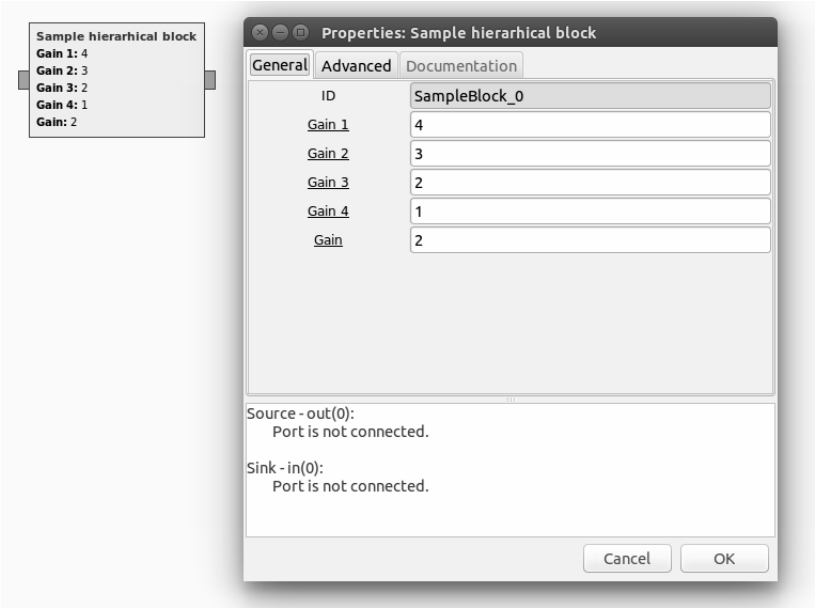
Bloki hierarchiczne tworzy się z istniejących bloków GNU Radio, bezpośrednio z poziomu aplikacji GRC, podobnie jak zwykłe diagramy przepływu sygnałów przez umieszczenie na diagramie, konfigurację i łączenie wybranych bloków (rys. 4). O utworzeniu bloku hierarchicznego decyduje odpowiednie użycie czterech bloków specjalnych: opcji, parametrów oraz portów wejścia i wyjścia [3].



Rys. 4. Tworzenie bloku hierarchicznego w GRC.
Widoczne bloki opcji, parametrów oraz portów wejścia i wyjścia

Fig. 4. *Hierarchical block definition within GRC.*
Option, parameter, input and output ports blocks are shown

Blok opcji (*option block*) to blok automatycznie dodawany do każdego diagramu budowanego w GRC. W typowych diagramach, służących do cyfrowego przetwarzania sygnałów, blok opcji umożliwia m.in. wybór rodzaju aplikacji (z graficznym interesem użytkownika lub bez), źródła sygnałów itp. Dla bloku hierarchicznego należy ustawić dla parametru *generate* wartość *hierarchical block*, a ponadto wpisać odpowiedni identyfikator tego bloku – jest wymagane, aby był unikatowy w całej bibliotece bloków GNU Radio.



Rys. 5. Nowo utworzony blok hierarchiczny i okno jego parametrów w GRC

Fig. 5. *An example of the hierarchical block and its properties window in GRC*

Blok parametru (*parametr block*) pełni funkcję podobną, jak maskowanie podsystemu w pakiecie Simulink, tzn. umożliwia parametryzację tworzonego bloku. Każdy blok parametru definiuje jeden parametr – jego nazwę, opis, typ i domyślną wartość. Zdefiniowanych w ten sposób parametrów bloku hierarchicznego można następnie użyć do skonfigurowania składających się nań bloków. Kiedy blok hierarchiczny zostanie utworzony i umieszczony w innym diagramie, jego wszystkie parametry będą dostępne w oknie opcji bloku (rys. 5), a użytkownik będzie mógł zmienić ich wartości.

Bloki portu wejścia (*pad source*) i portu wyjścia (*pad sink*) tworzą wejścia i wyjścia bloku hierarchicznego. Można wybrać m.in. typ danych (np. liczby całkowite lub zmiennoprzecinkowe) oraz liczbę wejść lub wyjść. Blok hierarchiczny może posiadać oba porty albo tylko jeden z nich.

Po skonfigurowaniu bloku opcji, zdefiniowaniu parametrów oraz dodaniu portów, do bloku hierarchicznego można dodawać dowolne bloki, łącząc je i konfigurując w taki sposób, aby realizowały założone funkcje. Do konfiguracji

poszczególnych bloków można wykorzystać zdefiniowane wcześniej parametry bloku hierarchicznego. Po zakończeniu pracy należy model skompilować (wystarczy kliknąć w odpowiedni przycisk na pasku narzędzi GRC), a nowy blok pojawi się w bibliotece bloków GRC, w kategorii *Custom*.

5.2. Tworzenie bloków *out-of-tree*

Tworzenie bloków *out-of-tree* jest znacznie bardziej skomplikowane, wymaga dobrej znajomości przynajmniej jednego z dwóch dostępnych języków programowania (C++ lub Python) w zakresie programowania obiektowego oraz, w zależności od przeznaczenia bloku, również znajomości odpowiednich zagadnień cyfrowego przetwarzania sygnałów. Dla osób nieobeznanych z charakterystycznym dla systemu Linux podejściem do tworzenia i instalacji oprogramowania dodatkowym utrudnieniem może być także konieczność korzystania z narzędzia *make* i plików *makefile*. Zadanie to znacznie ułatwia program *gr_modtool*, który obecnie jest włączony do pakietu GNU Radio [4, 5].

Tworzenie nowego bloku należy rozpocząć od utworzenia, za pomocą *gr_modtool*, nowego modułu. W tym celu należy w linii poleceń wpisać polecenie:

% gr_modtool newmod DemoMod,

gdzie *newmod* jest opcją oznaczającą utworzenie modułu, *DemoMod* zaś jest nazwą modułu. Program tworzy folder o nazwie *gr-DemoMod* i umieszcza w nim, w kilku podfolderach, wymagane pliki, a w tym pliki źródłowe w językach C++ i Python, pliki SWIG (*Simplified Wrapper and Interface Generator*), umożliwiające włączenie do GNU Radio bloków napisanych w języku C++ i plików dokumentacji.

W kolejnym kroku należy dodać pliki źródłowe nowego bloku i odpowiednio zmodyfikować plik *CMakeList.txt*, aby kod źródłowy został później poprawnie skompilowany, a blok zainstalowany i włączony do GNU Radio. To zadanie również wykonuje *gr_modtool*, po wprowadzeniu polecenia:

gr-DemoMod% gr_modtool add –t general –l cpp DemoGain,

gdzie: *add* to opcja oznaczająca utworzenie nowego bloku, *general* to rodzaj bloku, *cpp* oznacza wybór języka C++, *DemoGain* zaś to nazwa bloku.

Do modułu można dodać więcej bloków. Program pyta o listę parametrów nowego bloku oraz proponuje utworzenie testów jednostkowych. Twórcy GNU Radio, zgodnie z podejściem obowiązującym w tzw. zwinnych metodykach programowania [9], zalecają wszechstronne sprawdzanie poprawności tworzonego kodu za pomocą testów jednostkowych. Podejście takie nosi nazwę *Test-Driven Development* i wywodzi się z metodyki programowania ekstremalnego (*Extreme Programming*). Sposób tworzenia testów jednostkowych wykracza poza zakres niniejszej pracy i nie zostanie opisany.

Następnie należy zmodyfikować kod źródłowy bloku, w przypadku bloków tworzonych w języku C++ są to pliki *DemoGain_impl.h* oraz *DemoGain_impl.cc*, zawierające definicję klasy implementującej funkcje nowego bloku. W pliku

nagłówkowym *DemoGain_impl.h* można dodać deklaracje potrzebnych pól i metod prywatnych, natomiast w pliku źródłowym *DemoGain_impl.cc* należy odpowiednio zmodyfikować przynajmniej trzy metody: konstruktor klasy, metodę *forecast* (dla bloku typu *general* metoda ta podaje, jaka jest relacja pomiędzy liczbą próbek wyjściowych i wejściowych) oraz *general_work*, która odpowiada za przetwarzanie sygnałów [5]. Fragmenty pliku nagłówkowego *DemoGain_impl.h* oraz źródłowego *DemoGain_impl.cc* zostały przedstawione na wydrukach 1 i 2.

Kolejny etap pracy stanowi kompilacja modułu z nowym blokiem. W tym celu należy utworzyć folder *build* i wprowadzić polecenia:

gr-DemoMod/build% cmake ../
gr-DemoMod/build% make

```
class DemoGain_impl : public DemoGain
{
 private:
  float fGain;

 public:
  DemoGain_impl(float Gain);
  ~DemoGain_impl();

  void forecast (int noutput_items, gr_vector_int &ninput_items_required);

  int general_work(int noutput_items,
     gr_vector_int &ninput_items,
     gr_vector_const_void_star &input_items,
     gr_vector_void_star &output_items);
};
```

Wydruk 1. Fragment pliku nagłówkowego DemoGain_impl.h
Listing 1. *Fragment of the DemoGain_impl.h header file*

```
int
DemoGain_impl::general_work (int noutput_items,
                   gr_vector_int &ninput_items,
                   gr_vector_const_void_star &input_items,
                   gr_vector_void_star &output_items)
{
    const float *in = (const float *) input_items[0];
    float *out = (float *) output_items[0];

    for (int i=0; i<noutput_items; i++) {
      out[i] = fGain * in[i];
    }

    consume_each (noutput_items);
    return noutput_items;
}
```

Wydruk 2. Fragment pliku źródłowego DemoGain_impl.cc, metoda general_work
Listing 2. *Fragment of DemoGain_impl.cc file containing general_work method*

Dodatkowo należy zmodyfikować plik *DemoMod_DemoGain.xml*. Jest to plik XML wymagany przez GNU Radio, zawierający opis bloku, w tym jego parametrów oraz portów wejścia i wyjścia (wydruk 3). Zadanie to również może wykonać *gr_modtool*, jednak jego możliwości w tym zakresie są ograniczone i wystarczą jedynie w przypadku prostych bloków. Należy użyć polecenia:

gr-DemoMod% gr_modtool makexml DemoGain

gdzie *makexml* to opcja oznaczająca aktualizację pliku XML, a *DemoGain* jest nazwą bloku [5].

```
<block>
  <name>Demogain</name>
  <key>DemoMod_DemoGain</key>
  <category>DEMOMOD</category>
  <import>import DemoMod</import>
  <make>DemoMod.DemoGain($Gain)</make>
  <param>
    <name>Gain</name>
    <key>Gain</key>
    <type>float</type>
  </param>
  <sink>
    <name>in</name>
    <type>float</type>
  </sink>
  <source>
    <name>out</name>
    <type>float</type>
  </source>
</block>
```

Wydruk 3. Plik DemoMod_DemoGain.xml
Listing 3. *The DemoMod_DemoGain.xml file*

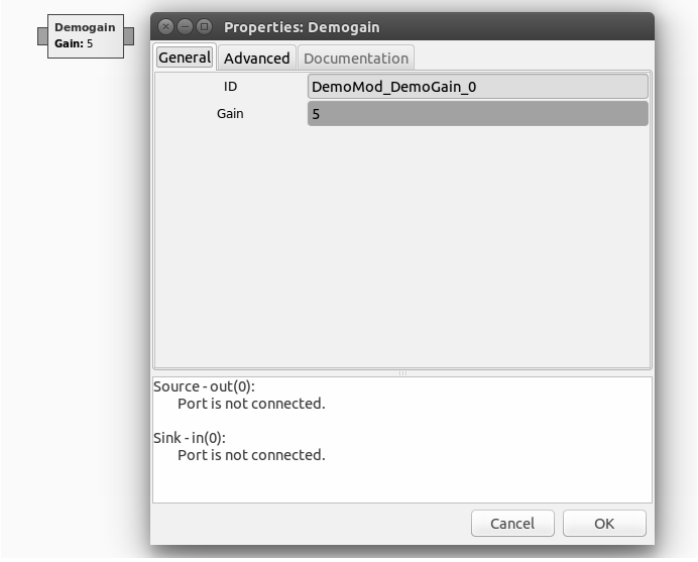
Ostatnim etapem jest instalacja modułu, do czego służą polecenia:

gr-DemoMod/build% sudo make install

gr-DemoMod/build% sudo ldconfig

W rezultacie ich wykonania blok zostanie dodany do biblioteki bloków GNU Radio, jak to pokazano na rysunku 6. Jeżeli program GRC był uruchomiony podczas kompilacji modułu, to należy go zrestartować lub odświeżyć listę bloków, wybierając odpowiednie polecenie z menu [5].

Tworzenie bloków *out-of-tree* do GNU Radio jest znacznie bardziej skomplikowane, niż np. w pakiecie Simulink tworzenie bloków z osadzoną funkcją Matlab (*Embedded MATLAB function*), czy nawet S-funkcją (*Level-2 M-file S-Function*), ze względu na niewielkie wsparcie ze strony GRC. Tworząc bloki *out-of-tree* do GNU Radio, trzeba korzystać z wielu zewnętrznych programów, a bez wsparcia *gr_modtool* nawet utworzenie podstawowych plików modułu byłoby nie lada wyzwaniem. Ponadto, w przypadku bardziej złożonych bloków, niezbędna jest szeroka wiedza na temat kodu źródłowego GNU Radio i znajomość obu używanych w nim języków programowania (C++ i Python) oraz języka XML.



Rys. 6. Blok *out-of-tree* DemoGain i okno jego parametrów w GRC

Fig. 6. *The out-of-tree block DemoGain and its properties window in GRC*

Sporym wyzwaniem może być również lokalizacja i poprawianie błędów w kodzie źródłowym, dlatego właśnie twórcy GNU Radio podkreślają konieczność stosowania testów jednostkowych, a w bibliotece GRC można znaleźć specjalne bloki ułatwiające debugowanie. W przeciwieństwie do GRC tworzenie nowych bloków w Simulinku odbywa się w całości z poziomu programu Matlab – zarówno edycja kodu źródłowego, jak i kompilacja, możliwe jest również debugowanie.

PODSUMOWANIE

Do niewątpliwych zalet zestawu narzędzi programistycznych GNU Radio należy z pewnością duży wybór bloków, dzięki którym można zbudować modele wielu różnych systemów radiokomunikacyjnych – zarówno prostych demonstracji łącza radiowego z modulacją analogową lub cyfrową dla celów dydaktycznych, jak i implementacji łącza radiowego systemów takich jak GSM, UMTS czy LTE dla celów badawczych. Nie bez znaczenia jest też fakt, iż GNU Radio jest oprogramowaniem darmowym. Do wad GNU Radio można zaliczyć swego rodzaju bałagan, jaki jest widoczny w dostępnych modułach i blokach (niektóre funkcje są zdublowane, np. modele kanałów, innych brakuje, jak np. modulacji DQPSK), a wynikający zapewne stąd, że GNU Radio jest rozwijane w części przez entuzjastów, z których każdy implementuje potrzebne mu funkcje. Przejawem tego są również znaczne różnice w jakości dokumentacji poszczególnych bloków – niektóre z nich są dobrze opisane, dzięki czemu relatywnie łatwo jest je skonfigurować i dopasować do konkretnego zastosowania, inne w ogóle nie mają żadnej dokumentacji.

Dla wielu osób duży problem może też stanowić tworzenie własnych bloków – ze względu na relatywnie słabe wsparcie ze strony GRC – do tworzenia bloków trzeba korzystać z zewnętrznych narzędzi (do edycji kodu źródłowego, kompilacji oraz instalacji bloków), wymagane są też zaawansowane umiejętności z zakresu programowania, znajomość mało popularnego języka Python, a wiele czynności trzeba wykonywać, korzystając z linii poleceń, czego nie ma w nowoczesnych środowiskach programistycznych.

Opierając się na własnych doświadczeniach z GNU Radio, autorzy potwierdzają stwierdzenie zawarte na stronie internetowej projektu: istnieją lepsze niż GNU Radio narzędzia do symulacji systemów radiokomunikacyjnych (chociażby wspomniany już pakiet Simulink), jednak wszędzie tam, gdzie przewidywana jest praca z rzeczywistymi sygnałami i transmisją sygnałów drogą radiową, GNU Radio będzie z pewnością dobrym wyborem – zarówno w celach dydaktycznych, jak i badawczych.

LITERATURA

1. Amarisoft, *Amari LTE 100 / Amari OTS 100 Software 4G network on PC*, http://www.amarisoft.com/?p=amarilte [25.06.2015].
2. GNU Radio, *GNU Radio 3.7.7 documentation*, http://gnuradio.org/doc/sphinx/ [2.07.2015].
3. GNU Radio, *GNU Radio Companion*, http://gnuradio.org/redmine/projects/gnuradio/ wiki/GNU RadioCompanion [2.07.2015].
4. GNU Radio, *GNU Radio Manual and C++ API Reference*, http://gnuradio.org/doc/doxygen-3.7.7/ [2.07.2015].
5. GNU Radio, *Out-of-tree modules*, http://gnuradio.org/redmine/projects/gnuradio/ wiki/OutOfTree Modules [2.07.2015].
6. GNU Radio, *Welcome to GNU Radio!*, http://gnuradio.org/redmine/projects/gnuradio/wiki [25.06.2015].
7. Iedema M., *Getting Started with OpenBTS*, O'Reilly, 2014.
8. National Instruments, *USRP-2920 Specifications.*
9. Martin R., Martin M., *Agile. Programowanie zwinne. Zasady wzorce i praktyki zwinnego wytwarzania oprogramowania w C#,* Helion, Gliwice 2008.

IMPLEMENTATION OF SOFTWARE DEFINED RADIO BASED ON GNU RADIO AND USRP2920

Summary

Software-defined radio is increasingly used for teaching, research and commercial applications. This paper presents GNU Radio and the USRP2920, which together form a universal software-defined radio platform.

***Keywords**: Software-defined radio, GNU Radio, USRP.*