

This is my thesis!

Spis treści

1	Introduction	1
1.1	Motivation	1
1.2	Organization	1
2	System eliminacji zakłóceń	3
2.1	Wstęp	3
2.1.1	Środowisko GnuRadio	3
2.2	Blok funkcjonalny	3
2.2.1	Założenia	3
2.2.2	Szablon projektu	4
2.2.3	Klasa obiektu bloku	4
2.2.4	Inicjalizacja	5
2.2.5	Implementacja	6
2.2.6	Instalacja	7
3	Modelowanie warunków propagacji	9
3.1	Wstęp	9
3.2	Strategia	9
3.3	Symulowanie zakłóceń	10
3.3.1	Sygnał sinusoidalny	10
3.3.2	Zjawisko wielodrogowości	10
3.4	Podsumowanie	12
4	Architektura odbiornika	13
4.1	Wstęp	13
4.2	Radio definiowane programowo	13
4.3	USRP	13
4.3.1	Architektura	13
4.3.2	Wybór właściwego USRP	14
5	Podsumowanie	15
6	Zakończenie	17

ROZDZIAŁ 1

Introduction

1.1 Motivation

I was motivated to write a Phd thesis because I did not want to work directly after finishing my study [4]

1.2 Organization

This thesis is organized as follows, ... We refer to Schön [7] for things you ...

2.1 Wstęp

Realizacja systemu eliminacji zakłóceń w odbiorniku polegała na stworzeniu elementu kompatybilnego z radiem USRP. Modułu, który pobiera sygnał z radia i dokonuje przetwarzania, a wynik tego przekierowuje do demodulatora. Przetwarzanie zakłada filtrację adaptacyjną w oparciu o algorytm LMS.

2.1.1 Środowisko GnuRadio

Głównym środowiskiem wspierającym radio USRP jest GnuRadio. GnuRadio pozwala na implementacje własnych bloków w oparciu o języki programowania Python i C++. W jego skład wchodzi narzędzie o nazwie `gnuradio-companion`, które służy do budowy diagramów z bloków o różnym przeznaczeniu. Czynniki te były kluczowe w podjęciu decyzji o doborze tego narzędzia do realizacji projektu.

Diagramy Diagram nazywany jest również projektem i łączy się z wielu bloków funkcjonalnych. Zawiera przynajmniej jedno źródło i jedno ujście sygnału. Tworzy łańcuch w którego ogniwach zachodzi przetwarzanie sygnału. Nadzoruje poprawność połączeń i zarządza wykonywaniem programu.

Bloki Mogą być źródłem sygnału (np. generator), ujściem (np. wyjście audio w PC) lub elementem pośredniczącym (np. filtry) w procesie przetwarzania. Każdy z bloków określa swój typ danych wejściowych i wyjściowych, co ogranicza możliwość połączeń, lecz gwarantuje kompatybilność typów.

2.2 Blok funkcjonalny

2.2.1 Założenia

Pierwszym założeniem projektowym bloku było zapewnienie interfejsu zgodnego ze schematem filtra. Powinien posiadać dwa wejścia: dla sygnału zakłóconego i sy-

gnału odniesienia. Wyjścia również dwa: pierwsze- sygnału odfiltrowanego, drugie- wektora błędu do monitorowania szybkości zbieżności algorytmu.

Kolejny parametr określa liczbę próbek wyjściowych do wejściowych. Blok nie będzie wpływał na liczbę próbek sygnału wejściowego, zatem stosunek n_{wej}/n_{wyj} powinien wynosić 1 : 1. W nomenklaturze środowiska GnuRadio takie przetwarzanie określa się jako synchroniczne.

Typem danych wejściowych jak i wyjściowych będą wartości zespolone próbek. Parametrami do ustawienia przez użytkownika w środowisku graficznym będą:

1. rząd N filtra w postaci liczby całkowitej
2. długość kroku μ w postaci liczby zmiennoprzecinkowej
3. liczba iteracji w postaci liczby całkowitej
4. początkowe współczynniki filtra w postaci wektora liczb o długości N

Parametry zostaną podzielone na wymagane (1, 2) i opcjonalne (3, 4)

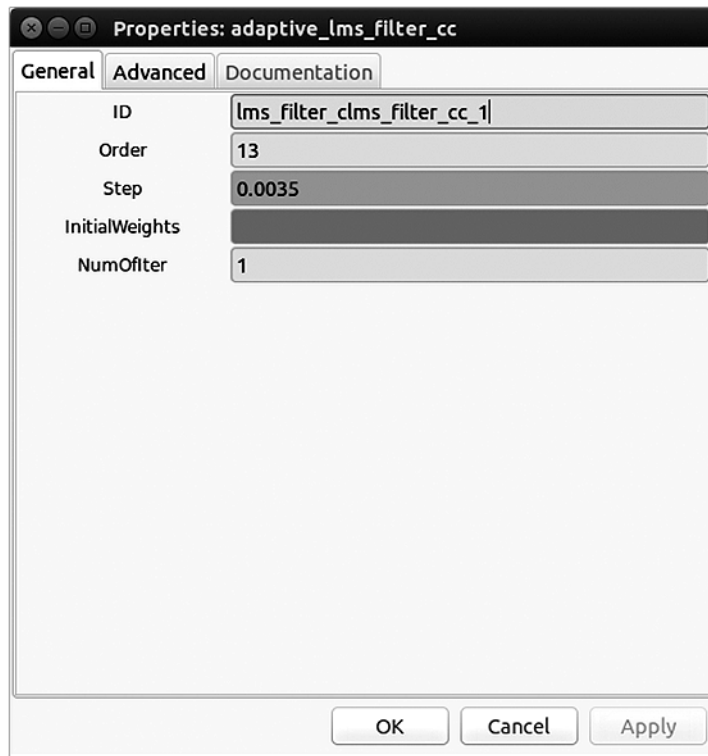
2.2.2 Szablon projektu

Środowisko GnuRadio dostarcza szereg narzędzi wspomagających pracę w rozwijaniu własnych modułów tak, aby były kompatybilne z już istniejącymi. Jednym z nich jest `gr_modtool`, który został wykorzystany do stworzenia struktury plików bloku filtra adaptacyjnego. Dzięki temu nasz blok będzie w stanie łączyć się z innymi, przekazywać i odbierać sygnały oraz będzie rozumiał polecenia wydawane przez środowisko w którym zostanie umieszczony. W projekt włącza się katalog z testami jednostkowymi, sprawdzającymi elementarne funkcjonalności bloku. Dobrze zaimplementowane i przemyślane testy jednostkowe informują o wykrytych błędach powstających podczas tworzenia algorytmu. Oprócz testów w projekcie tworzy się plik `.xml`, który określa reprezentację bloku w środowisku graficznym `gnuradio-companion`. W zależności od wybranego języka programowania, również plik z kodem źródłowym bloku.

2.2.3 Klasa obiektu bloku

Serce modułu odpowiedzialnego za przetwarzanie wejściowych strumieni danych znajduje się w pliku `adaptive_lms_filter_cc.py`. W tym miejscu zdefiniowana jest klasa bloku wraz z jej zmiennymi i funkcjami.

<code>adaptive_lms_filter_cc</code>
<code>-input_items</code>
<code>-output_items</code>
<code>-weights</code>
<code>-order</code>
<code>-step</code>
<code>-num_of_iter</code>
<code>+work()</code>
<code>+consume()</code>
<code>+set_history()</code>



Rysunek 2.1: Panel parametrów bloku

Funkcja `work` uruchamiana cyklicznie przez proces `scheduler` zawiera instrukcje odpowiedzialne za całe przetwarzanie wewnątrz bloku. Jest miejscem w którym należy umieścić implementację filtra. Pobiera z wejścia nadchodzące elementy i na ich podstawie oblicza produkty, które kieruje na wyjście.

Należy zauważyć, że w niektórych przypadkach do obliczenia wartości próbki wyjściowej potrzeba więcej niż jedną próbkę wejściową. Jest tak w przypadku filtracji, gdzie filtr o długości N potrzebuje $N - 1$ próbek minionych. Konieczne jest stworzenie bufora z którego można będzie pobierać te wartości. Funkcja `set_history` tworzy miejsce do przechowywania wcześniejszych próbek w pamięci, a jako parametr przyjmuje długość tego bufora.

2.2.4 Inicjalizacja

Następuje zawsze po uruchomieniu projektu. W każdym bloku (obiekcie) na diagramie tworzą się instancje do których przypisuje się wartości początkowe. Parametry mogą być dostosowywane przez użytkownika na etapie projektowania (w czasie przed uruchomieniem), lub zmieniane w trakcie wykonywania programu. Zmiana parametrów w biegu wiąże się z ponowną inicjalizacją bloku i może prowadzić do utraty danych przechowywanych wewnątrz tego obiektu. W środowisku `gnuradio-companion` okno właściwości pojawia się po podwójnym kliknięciu na blok. Zamiast stałej wartości parametru można przypisać referencję do jednego z dostępnych elementów sterujących np. suwaka lub pokrętła. Dzięki temu dostosowywanie bloku możliwe jest z poziomu graficznego panelu użytkownika (R).

Projekt pozwala na sterowanie dwoma parametrami podczas pracy filtra:

1. `order` - rząd $M = \{5, 6, \dots, 30\}$

2. step - krok adaptacji $\mu \in [0.0002, 0.005]$

2.2.5 Implementacja

Nieodłączną częścią modułu wykorzystywaną przez `gnuradio-companion` jest plik `adaptive_lms_filter_cc.xml`, który określa parametry wejściowe bloku, układ doprowadzeń i wyprowadzeń; gwarantuje kompatybilność połączeń z innymi blokami przez definiowanie typów danych dla wszystkich wejść, wyjść i parametrów. Pozwala również ustawić właściwości istotne dla poprawnego kategoryzowania w zasobach bibliotek.

Poniżej fragment struktury pliku `.xml`:

```
<?xml version="1.0"?>
<block>
  <name>adaptive_lms_filter_cc</name>
  <key>lms_filter_adaptive_lms_filter_cc</key>
  <category>[lms_filter]</category>
  <import>import lms_filter</import>
  <make>lms_filter.adaptive_lms_filter_cc
($order, $step, $weights $num_of_iter)</make>

  <param>
    <name>Order</name>
    <key>order</key>
    <type>int</type>
  </param>

  <param>
    <name>Step</name>
    <key>step</key>
    <type>float</type>
  </param>

  <sink>
    <name>reference</name>
    <type>complex</type>
  </sink>

  <source>
    <name>out</name>
    <type>complex</type>
  </source>
```

Nazwy parametrów zostały dobrane tak, aby blok dobrze integrował się ze środowiskiem i mógł zostać powtórnie wykorzystany w projektach innych użytkowników. Stąd język angielski, jako wspólny dla inżynierów na całym świecie.

Algorytm Aby zwiększyć efektywność i wygodę działań algebraicznych na macierzach, zastosowano narzędzia **numpy** będące rozszerzeniem języka Python.

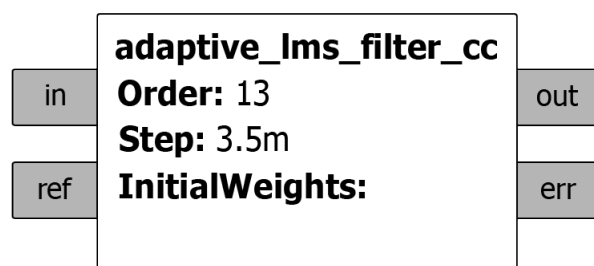
Algorytm pobiera z dwóch wejść próbki sygnałów i zapisuje je do bufora o wymiarze $b = 2 \times M$, którego każdy wiersz to jeden sygnał, a każda kolumna to kolejna próbka tego sygnału. Filtr odczytuje z bufora, wykonuje operacje algebraiczne, adaptacyjnie zmieniając swoje współczynniki. Odfiltrowany sygnał przekierowywany jest na wyjście, a z bufora wymazywane są próbki b_{1M}, b_{2M} .

```
Require:  $w \leftarrow w_0 \leftarrow [0, 0, \dots, 0]^T$ ,
buffer  $\leftarrow$  SET HISTORY( $2 \times M$ )
while Program running do
    repeat buffer  $\leftarrow$  CONSUME SAMPLES(reference input, noisy signal input)
    until buffer length < filter order
    FILTER(buffer)
    REMOVE FROM BUFFER(oldest  $x, d$ )
end while
function FILTER(samples)
    for all  $x, d \leftarrow$  samples do
         $y \leftarrow x \cdot w$ 
         $e \leftarrow d - y$ 
         $w \leftarrow w \cdot step \cdot x \cdot e$ 
         $y \leftarrow x \cdot w$ 
        return  $y, e$ 
    end for
end function
function CONSUME SAMPLES(reference input, noisy signal input)
     $x \leftarrow$  reference input
     $d \leftarrow$  noisy signal input
    buffer  $\leftarrow x, d$ 
    return buffer
end function
```

2.2.6 Instalacja

Kiedy wszystkie wymagane funkcjonalności zapisane w kodzie pomyślnie przechodzą etap testowania, implementację uznaje się za zakończoną. Moduł należy dołączyć do bibliotek GnuRadio, aby jego graficzna reprezentacja znalazła się w środowisku **gnuradio-companion**, a sam blok mógł zostać wykorzystany jako element w systemie odbiornika.

Proces instalacji nadzoruje narzędzie **cmake**, które wykonuje instrukcje z pliku **Makefile** powstałego podczas tworzenia szablonu projektu. Gotowy blok przedstawia Rysunek 2.2.



Rysunek 2.2: Reprezentacja graficzna bloku w `gnuradio-companion`

Modelowanie warunków propagacji

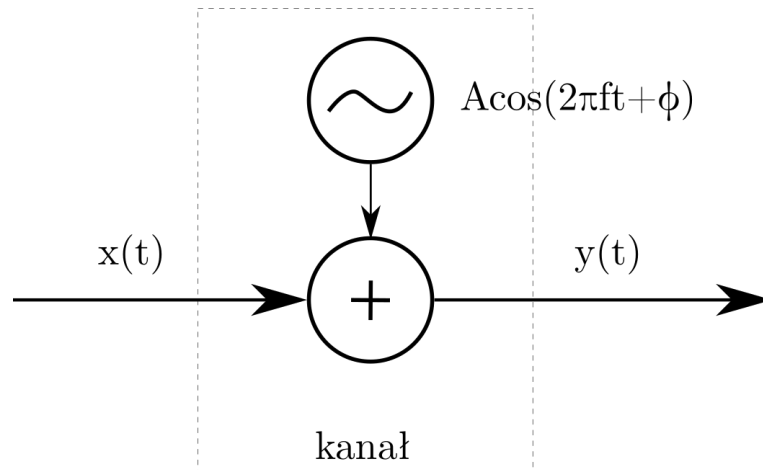
3.1 Wstęp

Kiedy proces implementacji osiąga pewien stopień zaawansowania nadchodzi czas testowania. Oznacza to sprawdzenie projektu pod kątem dostępnych funkcjonalności. Prototyp poddany zostaje testom, aby upewnić się czy nie posiada poważnych wad. Komercyjne ścieżki rozwoju produktu dopuszczają równoległy proces weryfikacji zarówno u producenta jak u klienta, aby lepiej dostosować się do jego potrzeb [1]. System odbiornika tworzony jest dla celów edukacyjnych, dlatego testy zostaną wykonane wyłącznie przez autora projektu.

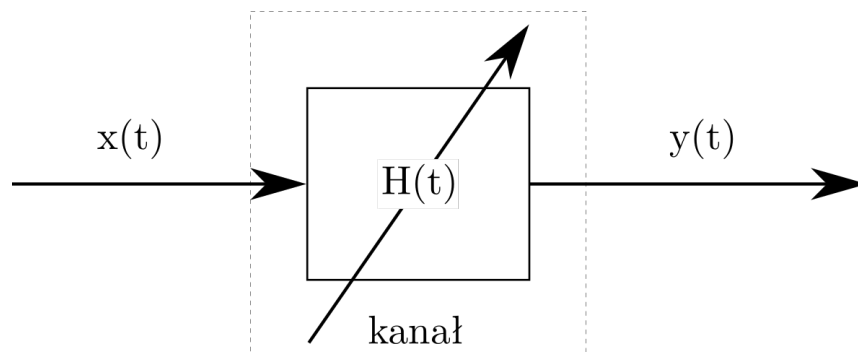
W tym rozdziale duży nacisk położony jest na poznanie sygnału, który dociera do odbiornika. Odtworzone zostaną warunki pracy odbiornika poprzez spreparowanie sygnału i poddanie go działaniom zjawisk występujących w kanale radiowym.

3.2 Strategia

Założenia projektowe nie określają jasno wymogów jakościowych i wydajnościowych pracy, a przyjęta strategia weryfikacji ma charakter eksperymentalny. Jako pierwsza przetestowana zostanie zdolność odbiornika do eliminacji zakłóceń pojedynczej częstotliwości. Scenariusz pokazany na Rysunku 3.1 zakłada, że w kanale transmisyjnym sygnał użyteczny silnie interferuje z falą sinusoidalną. Dodatkowo występuje nieznacznym dryft częstotliwości sygnału zakłócającego w czasie. Drugą próbą to modelowanie kanału jako filtra liniowego o zmiennej w czasie odpowiedzi impulsowej (Rysunek 3.2). Pozwoli to na przetestowanie odbiornika w warunkach zakłóceń związanych z propagacją wielodrogową [3].



Rysunek 3.1: Model kanału z zakłóceniem pojedynczej częstotliwości



Rysunek 3.2: Model kanału w propagacji wielodrogowej

3.3 Symulowanie zakłóceń

3.3.1 Sygnał sinusoidalny

Źródłem zakłóceń w postaci sygnałów sinusoidalnych mogą być radary dużych mocy, których częstotliwości harmoniczne zachodzą na pasmo użyteczne innych systemów [5]. Głównym powodem testowania działania odbiornika dla tego rodzaju zakłóceń jest popularna aplikacja, w której algorytm LMS jest w stanie śledzić pulsację wolnozmiennego sygnału i adaptacyjnie dostosowywać nastawy filtra [4]. Źródłem sygnału będzie blok **Signal Source**, który jako parametr przyjmuje amplitudę i częstotliwość fali.

3.3.2 Zjawisko wielodrogowości

Występuje wtedy, gdy sygnał z nadajnika, odbity od elementów otoczenia, dociera do odbiornika więcej niż jedną drogą. Czas propagacji jest różny dla różnych dróg sygnału. Tym samym każdy sygnał dociera do odbiornika opóźniony lub przyspieszony. Fale interferują, wzmacniają się lub osłabiają. W określonym czasie przy zachowaniu rozmieszczenia elementów otoczenia, nadajnika i odbiornika, kanał radiowy można traktować jako filtr o odpowiedzi impulsowej $h(t)$ [5].

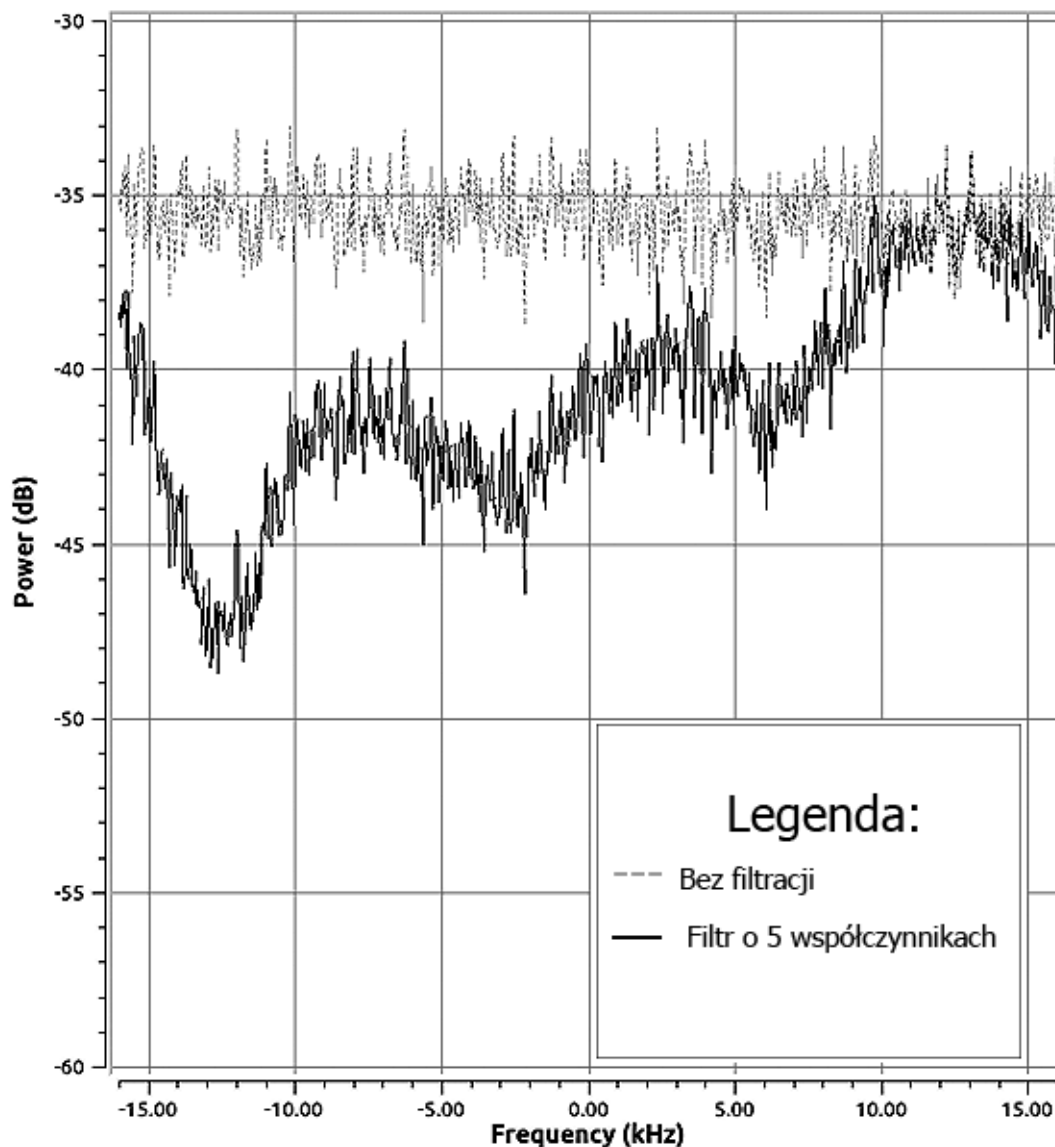
Jako model kanału z propagacją wielodrogową posłuży blok **Channel Model**. Istotnym argumentem wejściowym dla bloku jest wektor współczynników o długości

M . Współczynniki przyjmują wartości z zakresu $w_i = [0, 1]$, gdzie 0 to brak tłumienia, a dla 1 tłumienie $\rightarrow \infty$. Charakterystyka kanału określona jest w przedziale $[-\frac{f_s}{2} + f_c, \frac{f_s}{2} + f_c]$, gdzie f_s to częstotliwość próbkowania, a f_c częstotliwość nośna sygnału. Pasma kanału dzielone jest na odcinki o szerokości $\frac{f_s}{M}$. Za tłumienie na każdym z M odcinków odpowiada jeden współczynnik. Wartości współczynników powiązane zostały z suwakami do regulacji z panelu graficznego dla lepszej kontroli charakterystyki.

W celu zbadania charakterystyki, na wejście podłączono dwa sygnały szumu białego. Na Rysunku 3.3 przedstawiono widma tych sygnałów po przejściu przez blok kanału radiowego. Jedno wejście nie zostało poddane filtracji dla zachowania punktu odniesienia. W drugim, część częstotliwości została stłumiona przez zastosowanie filtra o regulowanych współczynnikach.

Efektem takich zakłóceń mogą być błędy w procesie decyzyjnym odbiornika, które prowadzą do zmniejszenia efektywnej przepustowości kanału.

Rysunek 3.3: Charakterystyka częstotliwościowa kanału radiowego



3.4 Podsumowanie

Przedstawione zostały dwa rodzaje zakłóceń, które posłużą do przeprowadzenia testów na odbiorniku.

1. Sygnał pojedynczej częstotliwości, pochodzący ze źródeł sąsiednich systemów telekomunikacyjnych.
2. Zwielokrotnienie tego samego sygnału w odbiorniku powodowane zjawiskiem propagacji wielodrogowej.

4.1 Wstęp

Wszechstronność systemu była jednym z wymogów projektowych architektury odbiornika. Dostosowanie do różnych rodzajów modulacji, pasm częstotliwości i mocy odbieranego sygnału nie powinno powodować zmian w konfiguracji sprzętowej. Jednostką zarządzającą działaniem odbiornika będzie komputer PC. Również na nim miejsce będzie miała eliminacja zakłóceń, zatem moduł radiowy powinien posiadać interfejs o wysokiej przepustowości.

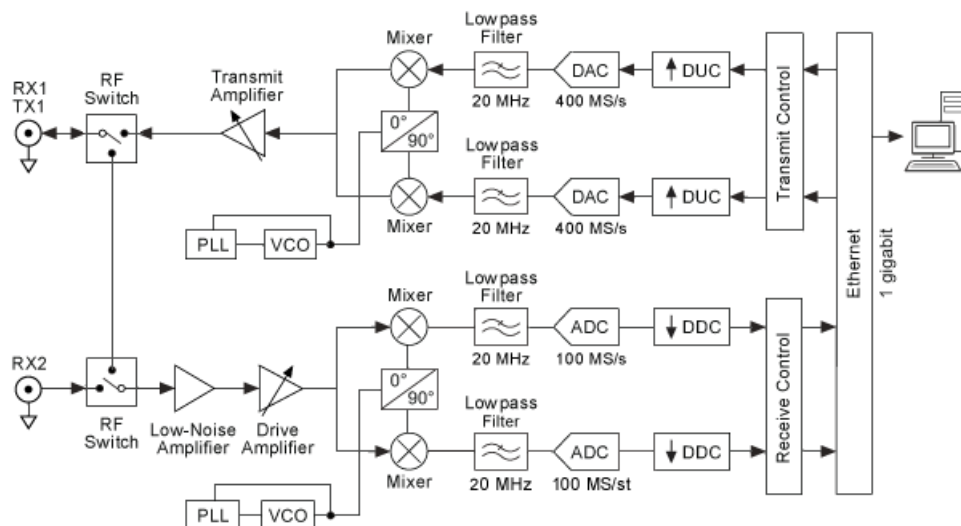
4.2 Radio definiowane programowo

Odpowiedzią na stawiane wymagania jest Radio Definiowane Programowo, w skrócie SDR od ang. *Software Defined Radio*. Dzięki zastosowaniu cyfrowych mieszaczy i filtrów urządzenie może zostać przeprogramowane do celów, których nikt nie przewidywał na etapie projektowania systemu telekomunikacyjnego. Stanowi pewne zabezpieczenie na wypadek zmiany koncepcji lub pojawienia się nowego standardu, ponieważ nie wymaga ingerencji w infrastrukturę sprzętową, której modernizacja może przewyższać koszt SDR. Radio programowalne pozwala na szybkie prototypowanie przy użyciu środowisk dostępnych na różnych systemach operacyjnych. Programy takie jak Matlab, Simulink, LabView lub wykorzystywany w tym projekcie GnuRadio, oferują łatwość użytkowania bez specjalistycznej wiedzy z zakresu architektury procesorów sygnałowych. [2]

4.3 USRP

4.3.1 Architektura

Architektura SDR przedstawiona na Rysunku 4.1 jest wspólna dla wszystkich produktów z rodziny USRP. Po przejściu przez wzmacniacz, sygnał analogowy trafia



Rysunek 4.1: Architektura USRP

na demodulator kwadraturowy, który rozdziela jego składowe I i Q . Widmo sygnałów w.cz. przeniesione zostaje do pasma podstawowego (BB) lub pośredniego (IF). Wtedy następuje konwersja analogowo- cyfrowa. USRP posiada po dwa przetworniki ADC na każdej linii TX/RX. Układ FPGA realizuje funkcję konwertera DDC (ang. *Digital Down Converter*), który ogranicza liczbę próbek wyprodukowanych przez ADC, zachowując informację niesioną przez sygnał. Mniejsza przepływność bitowa pozwala na efektywne przesyłanie danych z USRP do komputera w celu dalszego przetwarzania. [6]

4.3.2 Wybór właściwego USRP

USRP występuje w wielu modelach,

Pytania na które należało odpowiedzieć

1. Jaki jest wymagany zakres częstotliwości pracy odbiornika?
2. Jakiej szerokości pasmo odbiornik powinien odbierać?
3. Czy odbiornik będzie jednostką autonomiczną czy współpracującą z hostem (PC)?
4. Czy wymagana jest komunikacja dwukierunkowa (full duplex)?
5. Czy obsługiwany będzie tryb MIMO?

Pasmo

Interfejs

ROZDZIAŁ 5

Podsumowanie

ROZDZIAŁ 6

Zakończenie

Bibliografia

- [1] R. G. Cooper. A process model for industrial new product development. *IEEE Transactions on Engineering Management*, EM-30(1):2–11, Feb 1983.
- [2] Eugene Grayver. *Implementing software defined radio*. Springer, New York, 2013.
- [3] H. Hashemi. The indoor radio propagation channel. *Proceedings of the IEEE*, 81(7):943–968, Jul 1993.
- [4] Simon Haykin. *Adaptive Filter Theory (3rd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.
- [5] Simon Haykin and Michael Moher. *Modern Wireless Communication*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.
- [6] Pandeya Neel and Temple Nate. *About USRP Bandwidths and Sampling Rates*, 2016. https://kb.ettus.com/About_USRP_Bandwidths_and_Sampling_Rates.
- [7] Torsten Schoen and Co Author. Ten things you better not say to your wife. *Optimizing Husbands*, 21:85–91, 2013.

