This is my thesis!

Spis treści

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 [9] for things you ...

ROZDZIAŁ 2

System eliminacji zakłóceń

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 filtru 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)

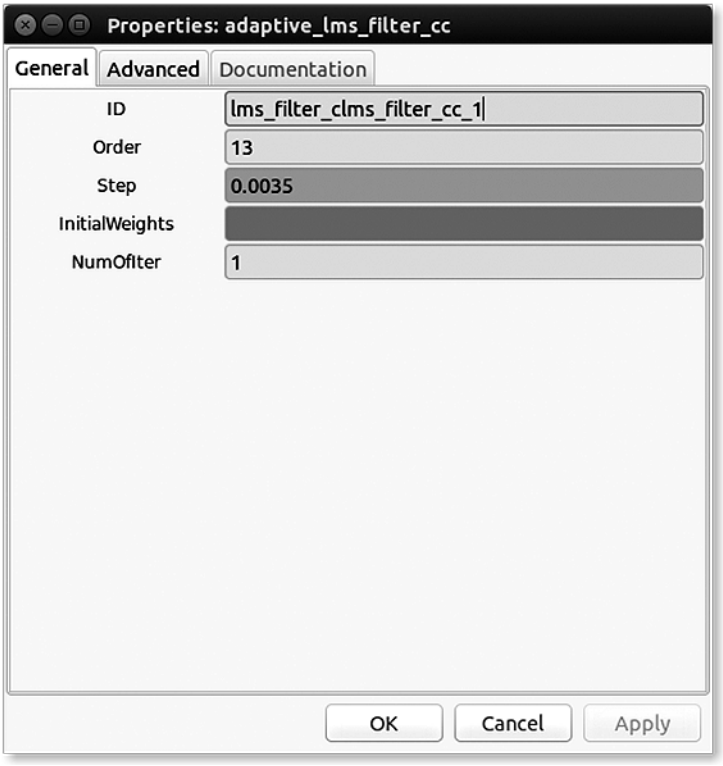
2.2.2 Szablon projektu

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.

adaptive_lms_filter_cc
-input_items -output_items -weights -order -step -num_of_iter
+work() +consume() +set_history()



Rysunek 2.1: Panel parametrów bloku

Funkcja `work` uruchamiana cykliczne 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 postawie 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 $M = \{5, 6, ..., 30\}$

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.

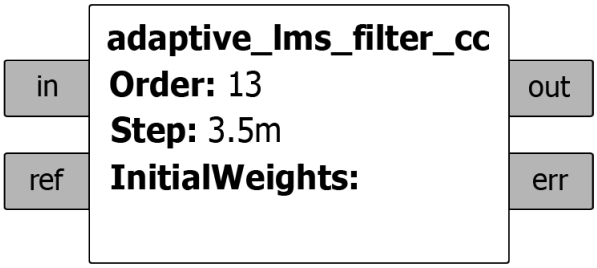
Algorytm pobiera z dwóch wejść próbki sygnałów i zapisuje je do bufora o wymiarze $b = 2 \times M$, którego każdy wiersz to jeden sygnał, a każda kolumna to kolejna próbka tego sygnału. Filtr odczytuje z bufora, wykonuje operacje algebraiczne, adaptacyjnie zmieniając swoje współczynniki. Odfiltrowany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

Require: $w \leftarrow w_0 \leftarrow [0, 0, ..., 0]^T$,
  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`

ROZDZIAŁ 3

Modelowanie warunków propagacji

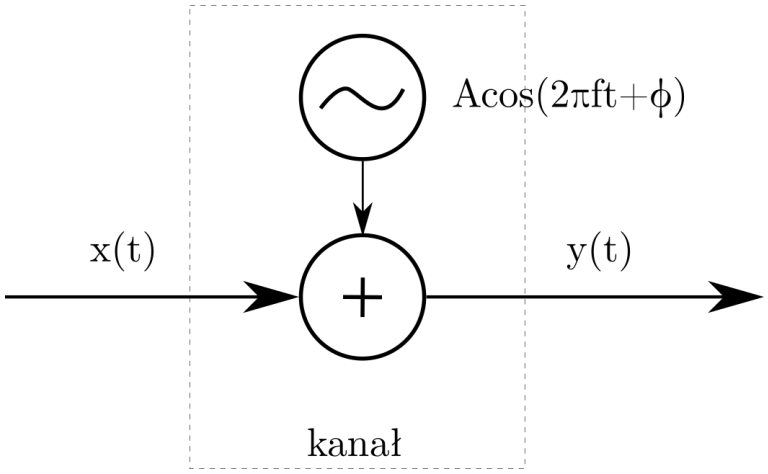
3.1 Wstęp

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.

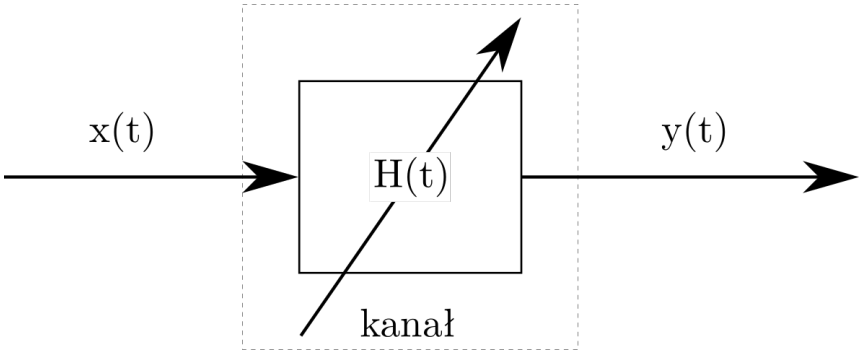
W tym rozdziale duży nacisk położony jest na poznanie sygnału, który dociera do odbiornika. Odtworzone zostaną warunki pracy odbiornika poprzez spreparowanie sygnału i poddanie go działaniom zjawisk występujących w kanale radiowym.

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 nieznaczny dryft częstotliwości sygnału zakłócającego w czasie. Druga próba to modelowanie kanału jako filtra liniowego o zmiennej w czasie odpowiedzi impulsowej (Rysunek 3.2). Pozwoli to na przetestowanie odbiornika w warunkach zakłóceń związancych 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 przyjmuję 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].

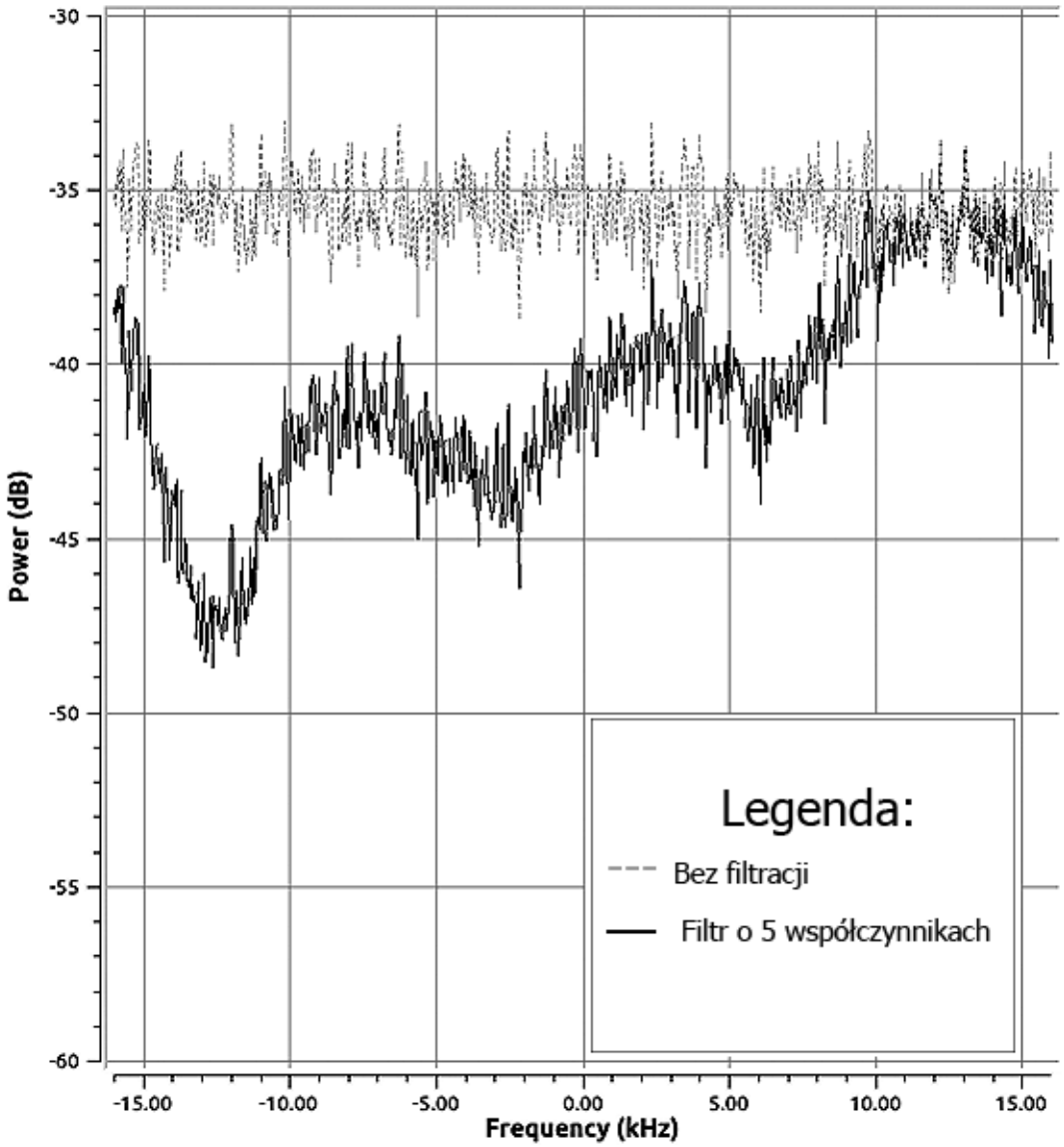
Jako model kanału z propagacją wielodrogową posłuży blok `Channel Model`. Istotnym argumentem wejściowym dla bloku jest wektor współczynników o długości

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. Pasmo 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.

ROZDZIAŁ 4

Architektura odbiornika

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.

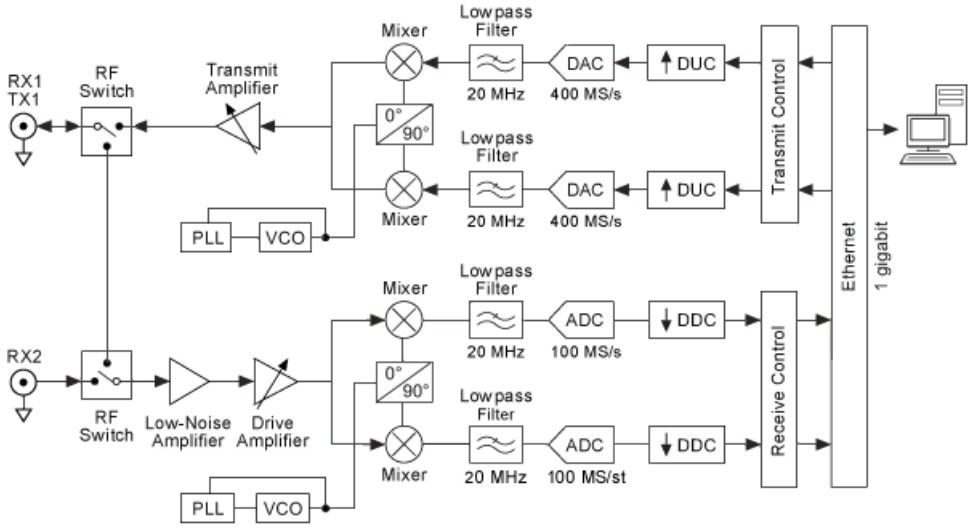
4.2 Radio definiowane programowo

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 przewidział na etapie projektowania systemu telekomunikacyjnego. Stanowi pewne zabezpieczenie na wypadek zmiany koncepcji lub pojawienia się nowego standardu, ponieważ nie wymaga ingerencji w infastrukturę 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 projeckie GnuRadio, oferują łatwość użytkowania bez specjalistycznej wiedzy z zakresu archiektury procesorów sygnałowych. [2]

4.3 USRP

4.3.1 Architektura

Architektura SDR przedstawiona na Rysunku 4.1 jest wspólna dla wszytkich 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 lini 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 Istnieją trzy typy pasma:

1. analogowe - to szerokość użytecznego (3dB) pasma pomiędzy wejściem w.cz., a interfejsem radiowym częstotliwości pośredniej lub podstawowej. Zazwyczaj ograniczany przez filtry pasmowe na płycie rozszerzeń.
2. przetwarzania układu FPGA- to częstotliwość próbkowania przetworników ADC na płycie bazowej. Jest to maksymalna hipotetyczna szerokość pasma dla USRP.

Tabela 4.1: Zestawienie parametrów USRP współpracujących z PC

Model	Interfejs	Host BW (MS/s)	ADC (MS/s)	MIMO
N210	GigE	50@16b 100@8b	100	Tak
N200	GigE	50@16b 100@8b	100	Tak
B210	USB 3.0	61.44	61.44	Tak
B210	USB 3.0	61.44	61.44	Nie

3. hosta - Interfejs do wymiany strumieni danych pomiedzy USRP a hostem również posiada swoje ograniczenia. Przepustowość zależy od rodzaju transmisji (simplex lub duplex) oraz liczby bitów przypadającej na jedną próbkę (8 lub 16 bitów).

Za pasmo całego systemu przyjmuje się najwęższe z wymienionych powyżej. [6]

Interfejs Dane pomiędzy radiem a hostem można przesyłać przez jeden spośród kilku interfejsów do wyboru. Najbardziej popularne to:

1. USB 3.0 - posiada przepustowość ok. 61.44 MS/s. Ze względu na to, że USB wykorzystuje wspólną linię do nadawania i odbierania danych zezwala jedynie na połączenie half duplex.

2. Gigabit Ethernet - przepustowość 25 MS/s w trybie full duplex. Dodatkową zaletą jest możliwość włączenia urządzenia do sieci i komunikowania się na dużą odległość.

Odpowiadając na pytania 1 i 2: Większość popularnych naziemnych systemów telekomunikacyjnych pracuje w paśmie do 5 GHz, zatem nie ma potrzeby przekraczania tej częstotliwości. Górną granicę szerokości kanału dla pojedynczej technologii stanowi WiFi z maksymalną szerokością 40 MHz. [8]

Odnosząc się do pytania 3. - Ze względu na łatwość prototypowania i dostępność komputera PC, radio będzie współpracowało z komputerem stacjonarnym. Komunikacja full duplex nie jest konieczna, co jest odpowiedzią na pyt. 4., lecz mobilność komputera jest ograniczona na tyle, że konieczne będzie zarządzanie zdalne poprzez interfejs Gigabit Ethernet. Projekt nie przewiduje również pracy w trybie MIMO (pyt. 5)

Tabela 4.1 przedstawia krytyczne z punktu widzenia projektu parametry radia. Jeżeli ograniczymy wybór do urządzenia z interfejsem GigE, pozostanie rozstrzygnąć pomiędzy modelem N210 a N200. Różnią się one jedynie ilością bramek w układzie FPGA, których to N210 ma więcej. Wykonanie projektu nie przewiduje przeprogramowania tego układu, zatem rozszerzanie FPGA nie jest konieczne. Wybrany został model N200 (Rysunek 4.2).

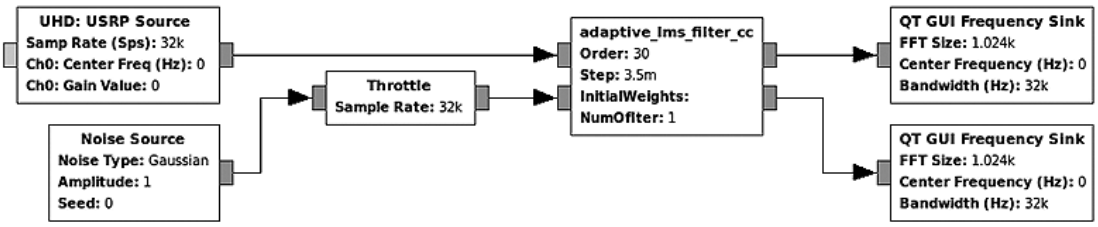
Rysunek 4.2: Model USRP N200 w obudowie.
Źródło: *https://www.ettus.com/product/details/UN200-KIT*

ROZDZIAŁ 5

Testy Funkcjonalne

5.1 Integracja bloku filtra w projekcie

W celu sprawdzenia integralności wykonanego bloku, zestawiono układ jak na Rysunku 5.1. Test integracyjny ma wykazać czy występuje zgodność typów danych we/wyj oraz zgodność panelu konfiguracyjnego użytkownika. Jeżeli nastąpi próba łączenia niekompatybilnych ze sobą bloków, wyświetli się błąd podobny do tego z Rysunku 5.2. Po podłączeniu bloków, środowisko nie zgłasza alarmów co oznacza, że filtr może przyjmować dane ze źródła USRP. Panel parametrów wyświetla się prawidłowo i wymaga od użytkownika podania kluczowych parametrów.

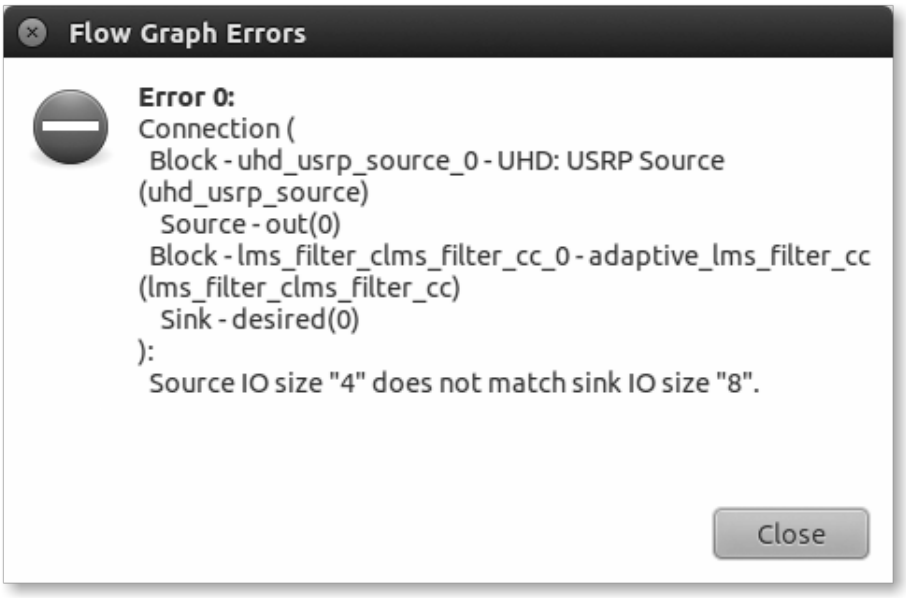


Rysunek 5.1: Układ do testu integracyjnego

5.2 Eliminacja zakłóceń sinusoidalnych

5.2.1 Weryfikacja poprawności działania

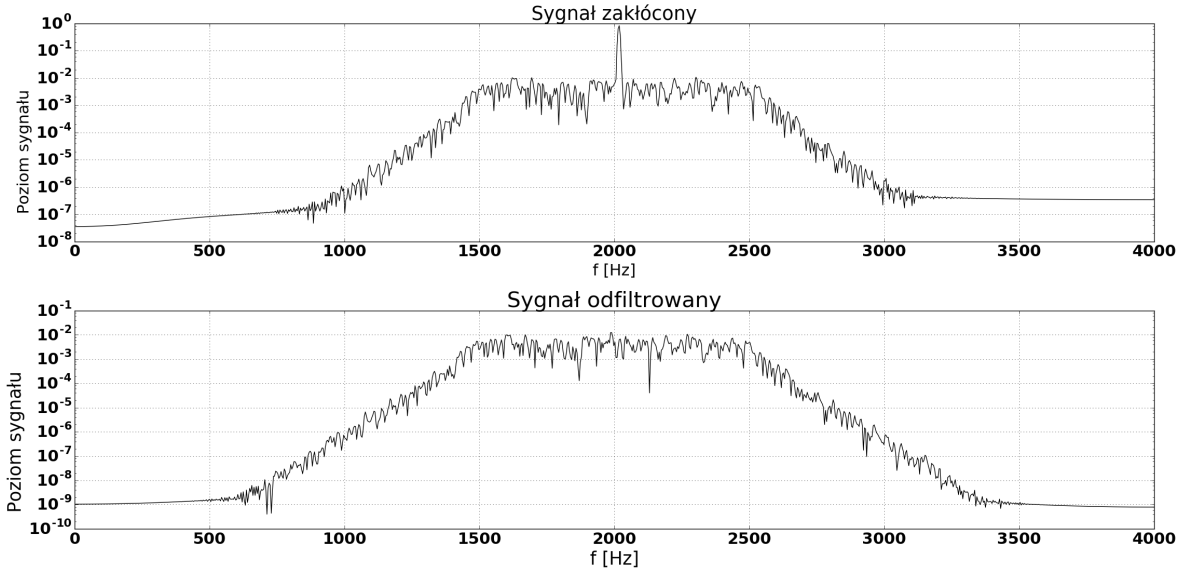
Test 1. Wygenerowano 1kHz szum pasmowy, który imituje sygnał danych. Sygnał został zsumowany z falą zakłócającą o częstotliwości $2kHz$. Suma została podana



Rysunek 5.2: Błąd projektu informujący o niezgodności typów danych.

na wejście główne filtra, a do wejścia odniesienia podłączono źródło sygnału sinusoidalnego o tej samej częstotliwości co składowa zakłócająca, lecz innej fazie. Częstotliwość sinusoid zwiększano stopniowo.

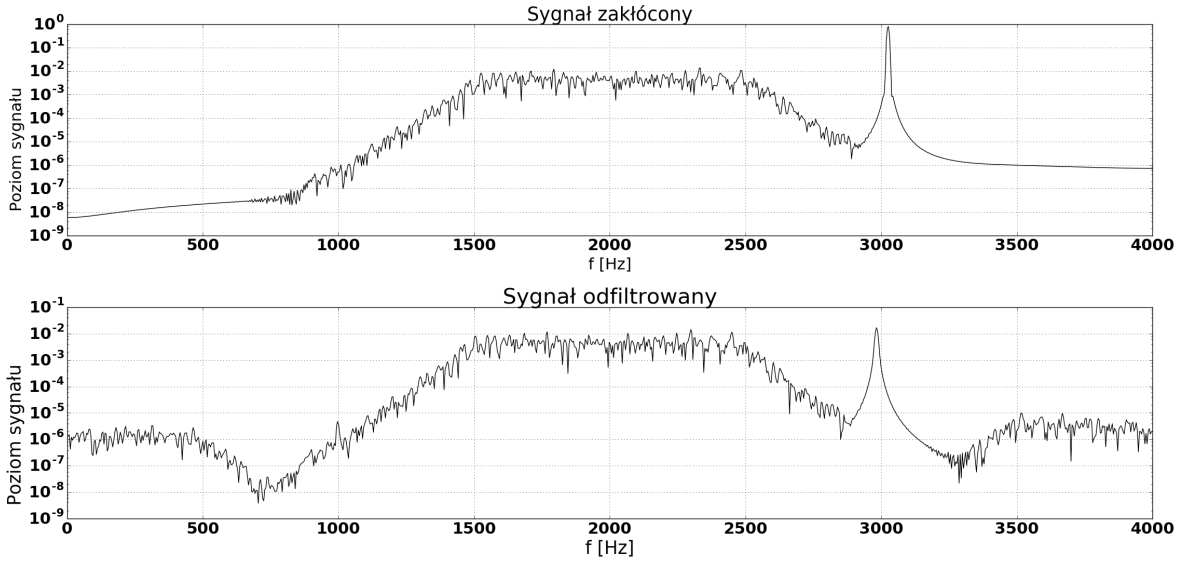
Obserwacja 1. Na Wykresie 5.3 przedstawiono efekt filtracji sygnału. Zakłócenie zostało pomyślnie odfiltrowane. Moc w paśmie użytecznym została wyrównana.



Rysunek 5.3: Filtracja zakłóceń o częstotliwości 2kHz

Obserwacja 2. Kiedy częstotliwość zakłócenia osiągnęła wartość wykraczającą poza pasmo użyteczne sygnału, zakłócenie znacznie wyróżniało się na wykresie widma. (Rysunek 5.4) Moc sinusoidy o częstotliwości 3kHz została stłumiona względem

niefiltrowanego sygnału o ok. 20dB. Świadczy to o tym, że algorytm podąża za zmianami sygnału interferującego.



Rysunek 5.4: Filtracja zakłóceń o częstotliwości 3kHz

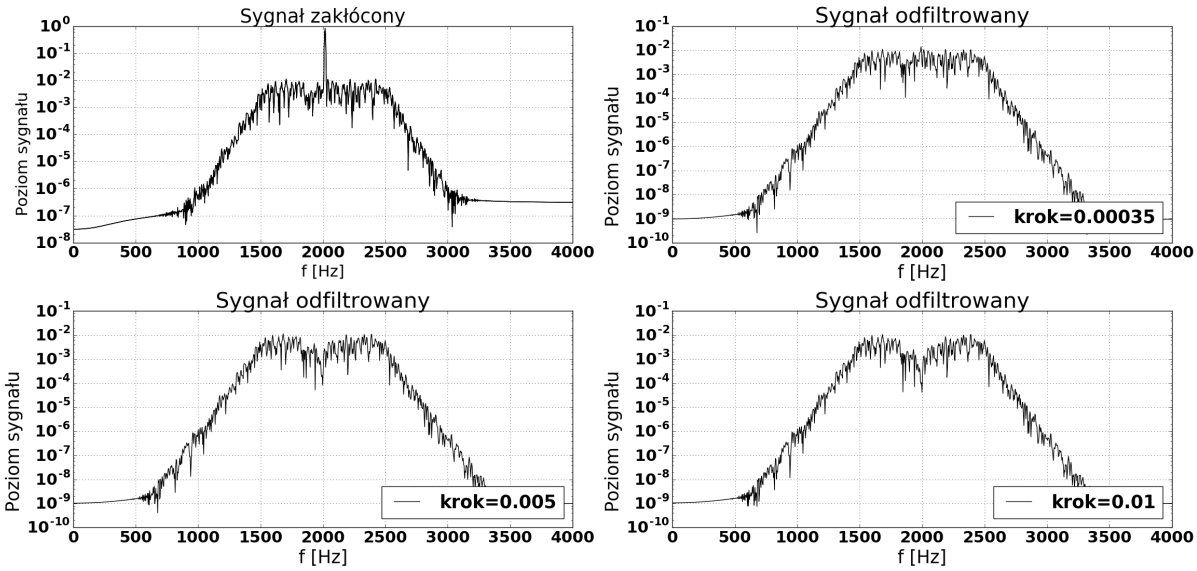
Test 2. Podobnie jak w teście 1. wykorzystano szum pasmowy i sinusoidalny sygnał zakłócający. Zmieniając wartość kroku adaptacji filtra zbadano jego wpływ na kształt widma. Dla każdej z wartości kroku obliczono wartość bezwzględną błędu estymacji.

Obserwacja 3. Im większy jest parametr kroku μ, tym szybsze są możliwości śledzenia przez filtr adaptacyjny (Rysunek 5.6). Duży parametr kroku μ może sprowadzać niakceptowalnie duży błąd średniokwadratowy. [**?**] Przykładem tego są wyraźnie obserwowalne na rys. 5.5 zaniki w widmie, na częstotliwości odpowiadającej cz. sygnału zakłócającego.

5.3 Korekcja adaptacyjna

Przedstawione w Tabeli 5.1 schematy konfiguracji odbiornika zostały stworzone do testów. Jedna z korektorem na bazie algorytmu LMS, druga bez korektora w celu porównania wydajności modułu `LMS DD Equalizer`. Skrót `DD` pochodzi od ang. *Decision Directed*, co oznacza że punktem odniesienia dla algorytmu jest obiekt konstelacji zawierający mapę symboli sygnału. [7]

Test 1. Zbadana została skuteczność eliminacji zakłóceń w postaci addytywnego białego szumu. Wartość amplitudy szumu jest jednym z parametrów bloku `Channel Model`. Celem było przesłanie pliku graficznego ze źródła do ujścia z wykorzystaniem modulacji QPSK i modelu kanału radiowego.



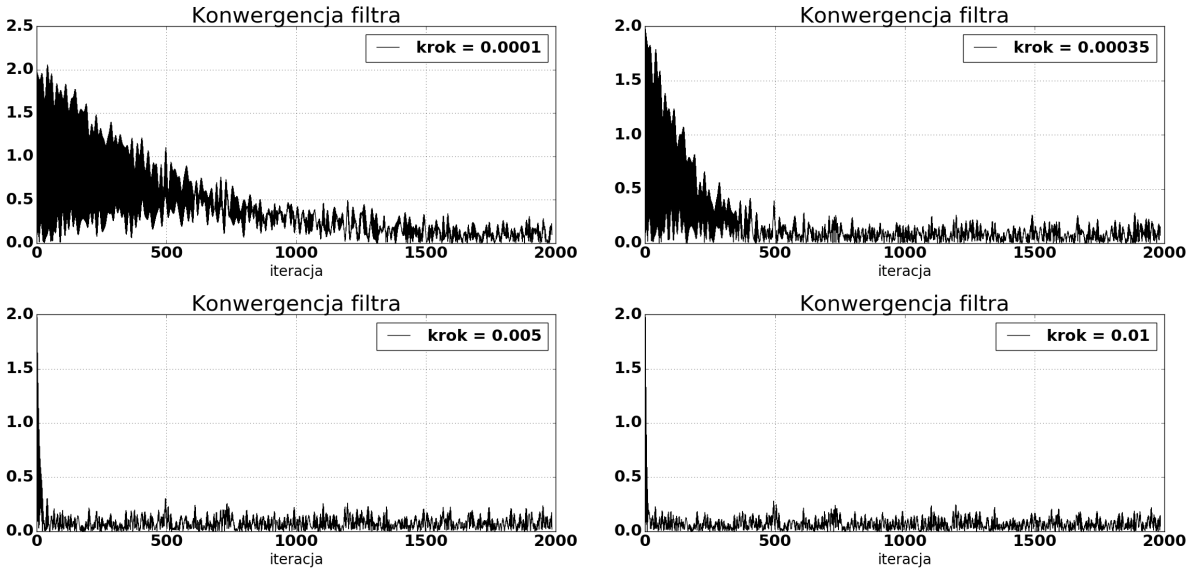
Rysunek 5.5: Wpływ parametru kroku na widmo sygnału.

Obserwacja 1. Na podstawie wyników zawartych w Tabeli 5.2 stwierdzono, że im większy poziom szumu addytywnego w kanale, tym gorszy stosunek S/N. Skutkuje to rozmyciem się konstelacji i błędnym rozpoznawaniem symboli w dekoderze. Efektem tego jest zniekształcony obraz na wyjściu.

Obserwacja 2. Tabela 5.3 zawiera wyniki próby z korekcją szumu. Korektor wpływa na wykres konstelacji, lepiej separując symbole przy tym samym poziomie szumu. Dekoder z korektorem przestaje poprawnie interpretować dane przy poziomie szumu powyżej $250mV$.

Test 3. Zbadano jakość korekcji modelu kanału przez blok `LMS DD Equalizer`. W tabeli 5.4 umieszczono wykresy charakterystyk kanału dla przypadku z korekcją i bez. Model kanału jest filtrem o pewnej odpowiedzi impulsowej. Zadaniem korektora jest zniwelowanie efektów kanału oddziałujących na sygnał. Wykres charakterystyki powstał poprzez wpuszczenie białego szumu na wejście kanału i obliczenie dyskretnej transformaty Fouriera z jego wyjścia.

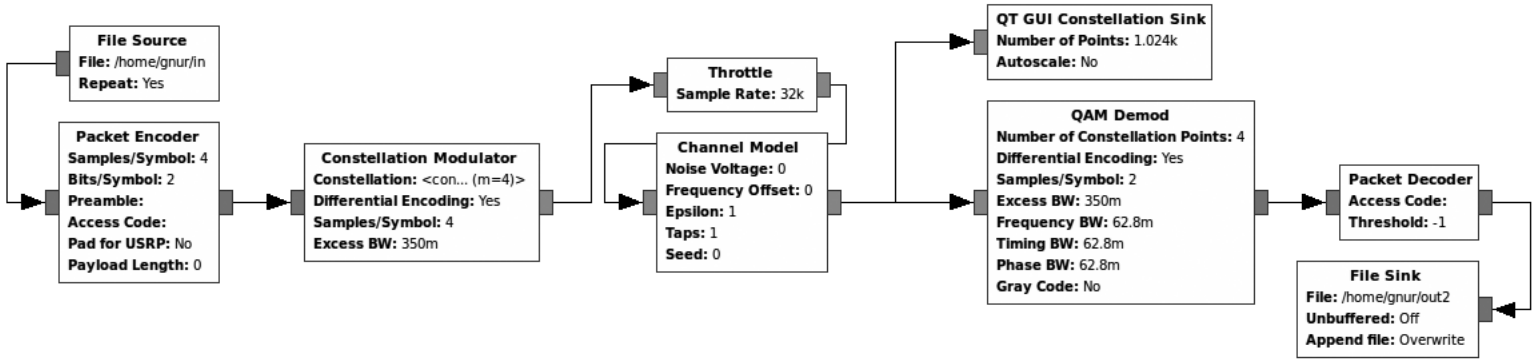
Obserwacja 1. Wykres konstelacji informuje o wielkości zniekształceń sygnału, które dla kanału bez korekcji są znacznie większe niż dla kanału z korekcją. Algorytm LMS efektywnie eliminuje zakłócenia związane z degradacją sygnału w kanale radiowym.



Rysunek 5.6: Wpływ parametru kroku na szybkość zbieżności algorytmu.

Tabela 5.1: Diagramy konfiguracji odbiornika

Transmisja bez korekcji



Transmisja z korekcją

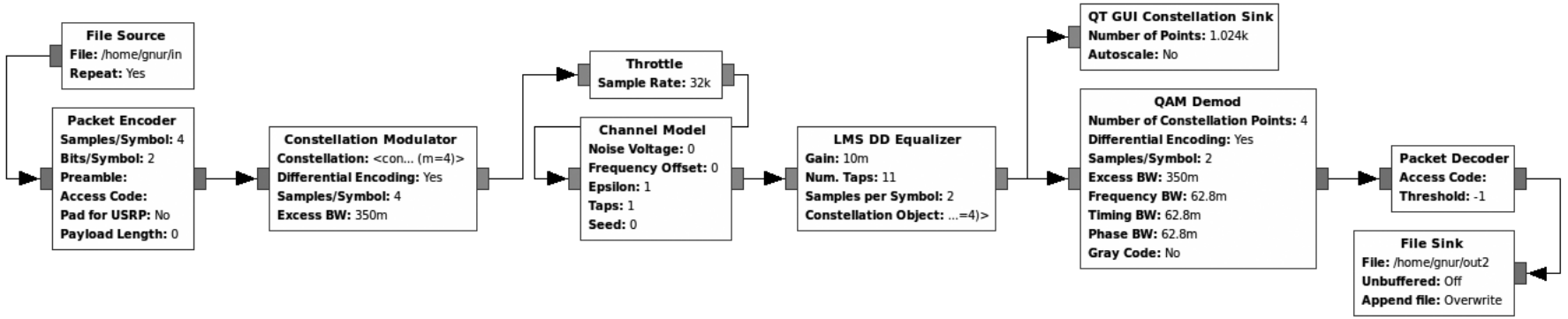


Tabela 5.2: Konfiguracja bez korektora

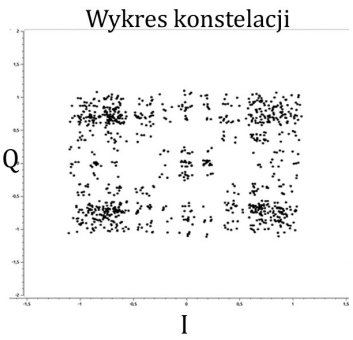
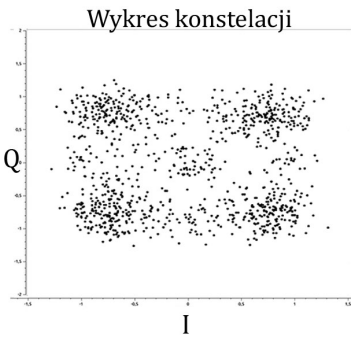
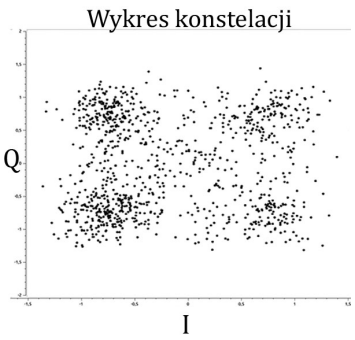
0mV	150mV	200mV
		
		

Tabela 5.3: Konfiguracja z korektorem

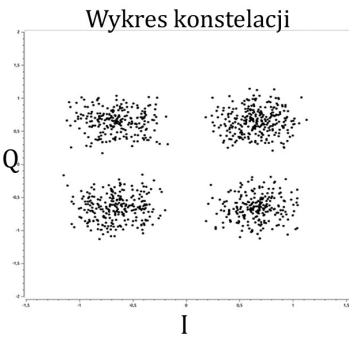
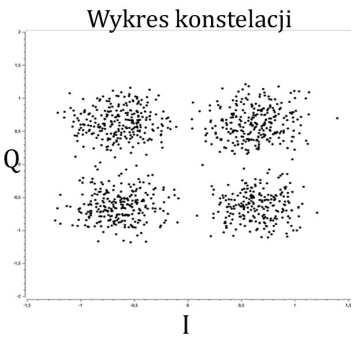
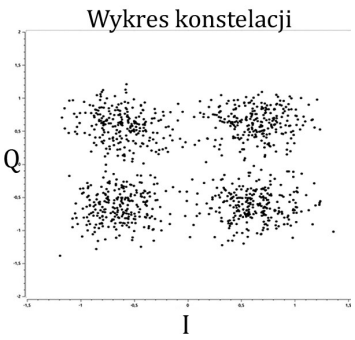
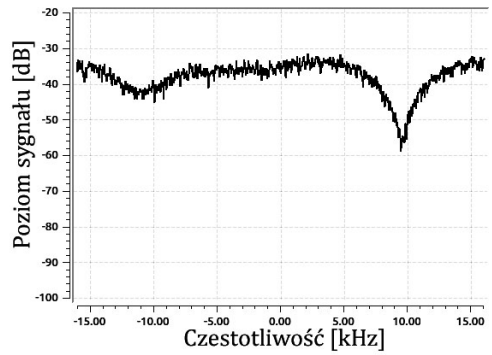
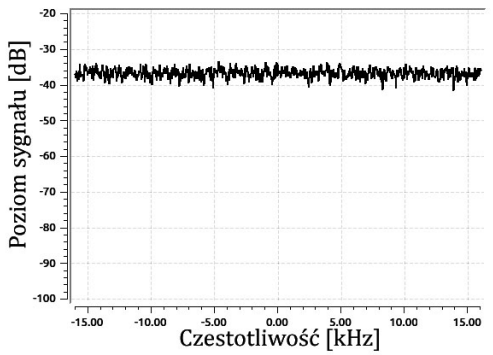
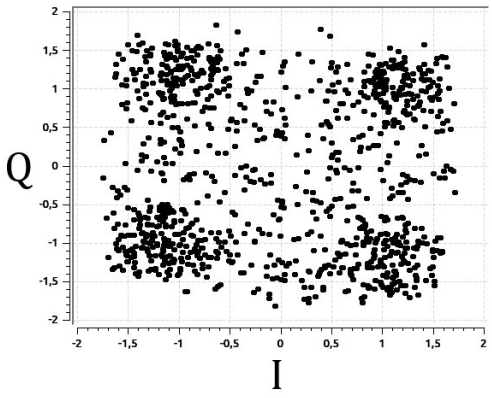
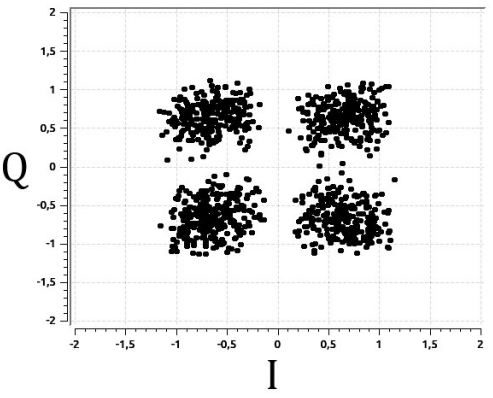
0mV	200mV	250mV
		
		

Tabela 5.4: Korekcja kanału

Kanał bez korekcji	Kanał z korekcją
	
	

ROZDZIAŁ 6

Podsumowanie

ROZDZIAŁ 7

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] S. J. Nowlan and G. E. Hinton. A soft decision-directed lms algorithm for blind equalization. *IEEE Transactions on Communications*, 41(2):275–279, Feb 1993.

[8] Minyoung Park. Ieee 802.11 ac: Dynamic bandwidth channel access. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[9] Torsten Schoen and Co Author. Ten things you better not say to your wife. *Optimizing Husbands*, 21:85–91, 2013.