Kurs:
Niezawodność i diagnostyka układów cyfrowych - Projekt

Implementacja i testy modulacji cyfrowych FSK i QAM

Autor:
Aleksandra Chrustek, 263900

Prowadzący:
dr inż. Maciej Nikodem

23.06.2023

Spis treści

1 Założenia projektowe

Założeniem projektu była implementacja modulatorów i demodulatorów wybranych modulacji cyfrowych, symulacja transmisji sygnału (zaszumianie) i demodulacji w odbiorniku. Na podstawie symulacji należało ocenić wpływ zakłóceń na niezawodność transmisji informacji dla różnych modulacji i zakłóceń.

2 Cel projektu

Cele projektu były następujące:
-Implementacja modulacji FSK.
-Implementacja modulacji 4-QAM oraz 16-QAM.
-Implementacja demodulatora FSK.
-Implementacja demodulatora QAM.
-Symulacja transmisji sygnału cyfrowego - dodanie zaszumień.
-Przeprowadzenie testów dla różnych parametrów i zakłóceń.
-Wnioski na podstawie wykonanych testów.

3 Informacje teoretyczne

3.1 FSK-Frequency-Shift Keying

3.1.1 Opis modulacji

Modulacja FSK (Frequency Shift Keying) jest techniką modulacji cyfrowej, w której informacja jest kodowana poprzez zmianę częstotliwości nośnej sygnału. W modulacji FSK, dwie lub więcej różnych częstotliwości nośnej są używane do reprezentacji różnych symboli danych. W przypadku dwóch częstotliwości, znanej jako FSK dwupunktowe, które zostało zaimplementowane w tym przypadku, każda częstotliwość reprezentuje jeden bit informacji. Na przykład, jeśli mamy częstotliwość f1 dla bitu 0 i częstotliwość f2 dla bitu 1, to zmiana częstotliwości między f1 a f2 oznacza zmianę bitu. W przypadku FSK wielopunktowego, więcej niż dwie częstotliwości jest używanych do kodowania większej liczby symboli. W przypadku modulacji FSK, sygnał nośny jest zazwyczaj sinusoidalnym sygnałem o stałej amplitudzie, który jest przełączany między różnymi częstotliwościami w zależności od wartości symboli danych.

3.1.2 Wpływ zakłóceń

Zakłócenia mogą wprowadzać błędy w odbiorze i dekodowaniu symboli, co prowadzi do błędnej interpretacji danych. Szumy, interferencje, zaniki sygnału mogą prowadzić

do zwiększenia Bit Error Rate, czyli liczby błędnie odebranych bitów w stosunku do wszystkich odebranych.

3.2 QAM-Quadrature Amplitude Modulation

3.2.1 Opis modulacji

Modulacja QAM (Quadrature Amplitude Modulation) jest jedną z technik modulacji cyfrowej, w której informacja jest przesyłana przez kombinację zmian amplitudy i fazy nośnego sygnału. Jest szeroko stosowana w systemach komunikacji bezprzewodowej, w tym w standardzie Bluetooth. W modulacji QAM sygnał wejściowy jest podzielony na dwie niezależne składowe - składową rzeczywistą (I) i składową urojoną (Q). Te dwie składowe są modulowane na nośnym sygnale o różnych fazach, a następnie są sumowane, tworząc sygnał QAM. Podstawową jednostką modulacji QAM jest symbol, który koduje pewną ilość bitów informacji. Symbol QAM składa się z kombinacji amplitudy i fazy, które reprezentują konkretne wartości bitów. Na przykład, w QAM-16, który używa 16 różnych kombinacji amplitudy i fazy, każdy symbol reprezentuje 4 bity informacji. Proces modulacji QAM polega na przypisaniu każdej kombinacji bitów do odpowiadającego symbolu QAM, a następnie modulacji tego symbolu na nośnej. Amplituda i faza nośnej są odpowiednio zmieniane, aby reprezentować wybrane kombinacje bitów. Następnie, zmodulowany sygnał QAM jest transmitowany przez kanał komunikacyjny. Na odbiorniku, proces demodulacji QAM polega na odzyskaniu symboli QAM z odebranego sygnału. Sygnał jest poddawany procesowi demodulacji, który rozdziela składowe I i Q. Następnie, na podstawie amplitudy i fazy tych składowych, przypisywane są odpowiednie kombinacje bitów. Ostatecznie, zdemodulowane bity są używane do odtworzenia pierwotnej informacji.

3.2.2 Opis modulacji 4-QAM

4-QAM koduje 2 bity informacji. Dwie składowe, rzeczywista (I) i urojona (Q), są modulowane na nośnej sygnału z odpowiednimi przesunięciami fazowymi, aby reprezentować te kombinacje. 4QAM jest stosowane w wielu systemach komunikacji, w tym w standardach WiFi

3.2.3 Opis modulacji 16-QAM

16QAM korzysta z 16 różnych kombinacji amplitudy i fazy dla każdego symbolu. Oznacza to, że każdy symbol QAM w 16QAM koduje 4 bity informacji. Działa to na podobnej zasadzie jak w 4QAM, ale oferuje większą przepustowość informacji dzięki większej liczbie kombinacji. 16QAM jest stosowane w systemach komunikacji szerokopasmowej.

3.2.4 Wpływ zakłóceń

Wpływ zakłóceń na modulację QAM jest istotnym zagadnieniem w systemach komunikacji. Zakłócenia, takie jak szumy, interferencje, zaniki sygnału, mogą negatywnie wpływać na jakość i niezawodność transmisji QAM. W przypadku QAM, zakłócenia mogą prowadzić do błędów w odczycie symboli, co z kolei prowadzi do błędnej interpretacji bitów informacji. Im większe zakłócenia, tym większe prawdopodobieństwo wystąpienia błędów. Wzrost zakłóceń może prowadzić do zwiększenia Bit Error Rate, czyli liczby błędnie odebranych bitów w stosunku do wszystkich odebranych.

4 Opis implementacji

4.1 FSK

1. Importowanie niezbędnych bibliotek: numpy i matplotlib.pyplot.
2. Definiowanie częstotliwości nośnej (fc) - odpowiada częstotliwości nośnej Bluetooth w paśmie 2,4 GHz - oraz szybkości bitowej (bit-rate) - jest standardową prędkością dla Bluetooth.
3. Definiowanie wiadomości do przesłania (message). Jest to ciąg znaków, który zostanie przekonwertowany na postać binarną.
4. Konwertowanie wiadomości na postać binarną. Każdy znak w wiadomości jest zamieniany na jego reprezentację binarną (o długości 8 bitów). Wynikowy ciąg binarny jest przechowywany w zmiennej binary-message.
5. Definiowanie dewiacji częstotliwości dla każdego bitu (delta-f). W przypadku Bluetooth, stosuje się technikę GFSK (Gaussian Frequency-Shift Keying), w której częstotliwość nośna jest przesuwana w górę lub w dół w zależności od wartości bitu. Odchylenie częstotliwości jest wyliczane na podstawie prędkości transmisji bitowej. W tym przypadku dewiacja jest równa 1/8 wartości szybkości bitowej.
6. Tworzenie osi czasu na podstawie długości wiadomości oraz wartości szybkości bitowej.
7. Tworzenie sygnału nośnego (carrier) jako fali sinusoidalnej o częstotliwości nośnej (fc) i zdefiniowanej osi czasu (time).
8. Tworzenie sygnału zmodulowanego (modulated-signal) na podstawie wiadomości binarnej. W pętli iteruje się po każdym bicie wiadomości. Jeśli bit jest równy 0, to odpowiedni fragment czasu sygnału zmodulowanego zostaje zainicjalizowany jako

fala sinusoidalna o częstotliwości (fc - delta-f). Jeśli bit jest równy 1, to odpowiedni fragment czasu sygnału zmodulowanego zostaje zainicjalizowany jako fala sinusoidalna o częstotliwości (fc + delta-f).

9. Wyświetlenie wykresu sygnału zmodulowanego bez zakłóceń.
10. Tworzenie zakłóceń (interference) jako fali sinusoidalnej o podwójnej częstotliwości dewiacji (2 * delta-f) i zdefiniowanej osi czasu (time).
11. Dodawanie zakłóceń do sygnału zmodulowanego.
12. Wyświetlanie wykresu sygnału zmodulowanego z zakłóceniami.
13. Wykonywanie demodulacji koharentnej. Tworzenie sygnału zdemodulowanego na podstawie sygnału zmodulowanego z zakłóceniami. Pętla iteruje po każdym bicie wiadomości. Jeśli bit jest równy 0, to odpowiedni fragment czasu sygnału zdemodulowanego jest mnożony przez fale sinusoidalne o częstotliwości (fc - delta-f). Jeśli bit jest równy 1, to odpowiedni fragment czasu sygnału zdemodulowanego jest mnożony przez fale sinusoidalne o częstotliwości (fc + delta-f).
14. Wykonanie filtracji dolnoprzepustowej. Definiowanie częstotliwości odcinającej dla filtru dolnoprzepustowego. Częstotliwość odcinania jest ustawiana na połowę prędkości transmisji bitowej, aby wyeliminować komponenty wysokoczęstotliwościowe, które mogą wynikać z procesu demodulacji. Tworzenie filtru dolnoprzepustowego o rzędzie 5 i zdefiniowanej częstotliwości odcinającej - te zmienne reprezentują współczynniki filtru dolnoprzepustowego Butterwortha, który jest używany do wygładzania demodulowanego sygnału.
15. Zastosowanie filtru dolnoprzepustowego do sygnału zdemodulowanego.
16. Obliczenie odebranej wiadomości na podstawie sygnału zdemodulowanego. Iterowanie po każdym bicie wiadomości. Jeśli wartość sygnału po filtracji w odpowiednim fragmencie czasu jest większa od zera, to dodawane jest '1' do otrzymanej wiadomości. W przeciwnym przypadku dodawane jest '0'.
17. Obliczenie błędu bitowego. Sumowanie różnic pomiędzy bitami wiadomości oryginalnej a otrzymanej i obliczenie błędu bitowego. Obliczenie błędu bitowego jako stosunek liczby błędów do długości wiadomości. Wyświetlenie wyniku błędu bitowego.
18. Wyświetlanie wykresu zdemodulowanego i przefiltrowanego sygnału.

4.2 QAM

1. Importowane są niezbędne biblioteki numpy i matplotlib.
2. Zdefiniowana jest częstotliwość nośnej (fc) i okres symbolu (Tsym).
3. Zdefiniowane są punkty konstelacji dla 4-QAM (constellation-4QAM) i 16-QAM (constellation-16QAM).
4. Zdefiniowane są dane do przesłania dla 4-QAM (data-4QAM) i 16-QAM (data-16QAM).
5. Wybierana jest konstelacja i dane, które zostaną użyte (w tym przypadku 4-QAM).
6. Tworzony jest wektor czasu na podstawie danych i okresu symbolu.
7. Tworzony jest sygnał zmodulowany o długości odpowiadającej długości wektora czasu. W pętli iteruje się po danych i przypisuje odpowiednie symbole z konstelacji do odpowiednich odcinków czasowych w sygnale zmodulowanym.
8. Do sygnału dodawany jest szum. Obliczane są moc sygnału i moc szumu na podstawie stosunku sygnału do szumu (SNR) podanego w dB. Generowany jest szum gaussowski (noise) o odpowiednim poziomie mocy i dodawany do sygnału zmodulowanego.
9. Generowane są wykresy sygnału zmodulowanego i sygnału zaszumionego.
10. Wykonywana jest demodulacja QAM. Tworzona jest pusta lista (demodulated-data) na zdemodulowane dane. W pętli iteruje się po danych i dla każdego symbolu wykonuje się następujące czynności:
    -Pobierany jest odpowiadający mu symbol z konstelacji.
    -Wybierany jest odpowiedni segment sygnału zaszumionego.
    -Wykonywane jest obliczenie korelacji między symbolem a segmentem.
    -Znajdowany jest indeks najbliższego punktu konstelacji.
    -Indeks ten jest dodawany do listy zdemodulowanych danych.
11. Obliczany jest współczynnik błędów bitowych poprzez porównanie oryginalnych danych z zdemodulowanymi danymi.
12. Wyświetlany jest współczynnik błędów bitowych.
13. Wykonywany jest wykres zdemodulowanych danych.

5 Badania

5.1 FSK

5.1.1 Wybrane zakłócenia

Wpływ zakłóceń na sygnał w tym przypadku transmisji sygnału jest badany przez dodanie sinusoidalnego sygnału zakłócającego do sygnału modulowanego oraz analizowanie wpływu tych zakłóceń na proces demodulacji i filtrowania. Sinusoidalny sygnał zakłócający (interference) ma częstotliwość dwukrotnie większą od odchylenia częstotliwości (2 * delta-f). Jest on dodawany do sygnału modulowanego, tworząc sygnał modulowany z zakłóceniami. Obliczane jest error bit rate poprzez porównanie każdego bitu wiadomości oryginalnej z otrzymaną wiadomością.

5.1.2 Testy

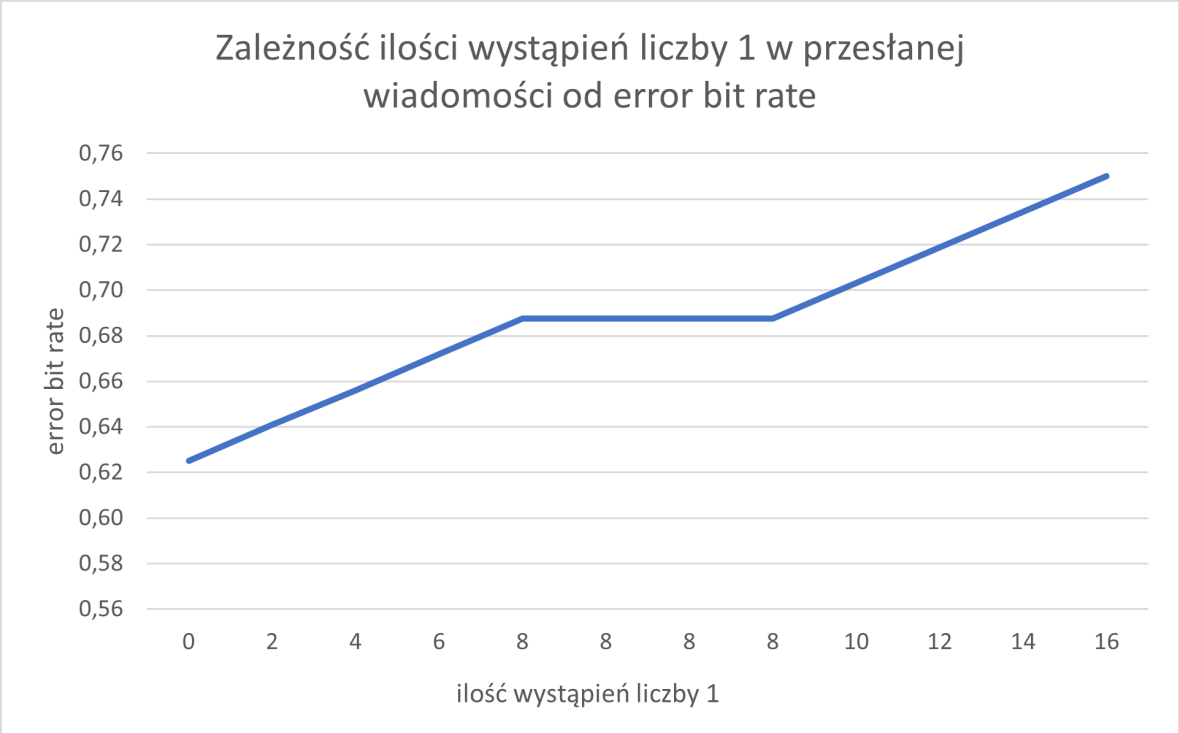
W modulacji FSK testom został poddany wpływ przesyłanej wiadomości na błąd bitowy. Zostało to wykonane na przykładzie liczby 16- bitowej. W liczbie zmieniano ilość wystąpień 0 i 1, aby zaobserwować jaki daje to efekt. Wyniki pomiarów przedstawia tabela nr 2.

5.1.3 Tabele wyników

Tabela 1: Tabela zależności wystąpień 0 i 1 w wiadomości od błędu bitowego

Wybrany ciąg bitów	Ilość wystąpień liczby 1	Error bit rate
0000000000000000	0	0,625
1000000000000000	2	0,641
1111000000000000	4	0,656
1111110000000000	6	0.672
1111111100000000	8	0,688
0000000011111111	8	0,688
1010101010101010	8	0,688
0101010101010101	8	0,688
1111111111000000	10	0,703
1111111111110000	12	0,719
1111111111111100	14	0,734
1111111111111111	16	0,750

5.1.4 Wykresy



Rysunek 1: Wykres przedstawiający wpływ wystąpień 0 i 1 w wiadomości na błąd bitowy

5.2 QAM

5.2.1 Wybrane zakłócenia

Wpływ zakłóceń na sygnał jest badany poprzez dodanie szumu do sygnału modulowanego i analizowanie wpływu tego szumu na proces demodulacji QAM. SNRdB (Signal to Noise Ratio) określa stosunek sygnału do szumu w dB. Na jego podstawie obliczana jest moc sygnału i moc szumu. Wygenerowany szum o odpowiedniej mocy jest dodawany do sygnału modulowanego, tworząc sygnał z zakłóceniami. Tak samo jak w przypadku FSK, obliczane jest error bit rate poprzez porównanie każdego bitu wiadomości oryginalnej z otrzymaną zdemodulowaną wartością.

5.2.2 Testy

Testy zakłóceń w tym przypaadku transmisji sygnału koncentrowały się na badaniu wpływu wartości SNR na błąd bitowy. Pierwszy test został przeprowadzony dla konstelacji 4-QAM dla wartości SNR równych kolejno 1, 5, 7, 10, 13. Wyniki przedstawia tabela nr 3. Następnie takiemu samemu testowi zostało poddane 16-QAM. Wyniki znajdują się w tabeli nr 4.

5.2.3 Tabele wyników

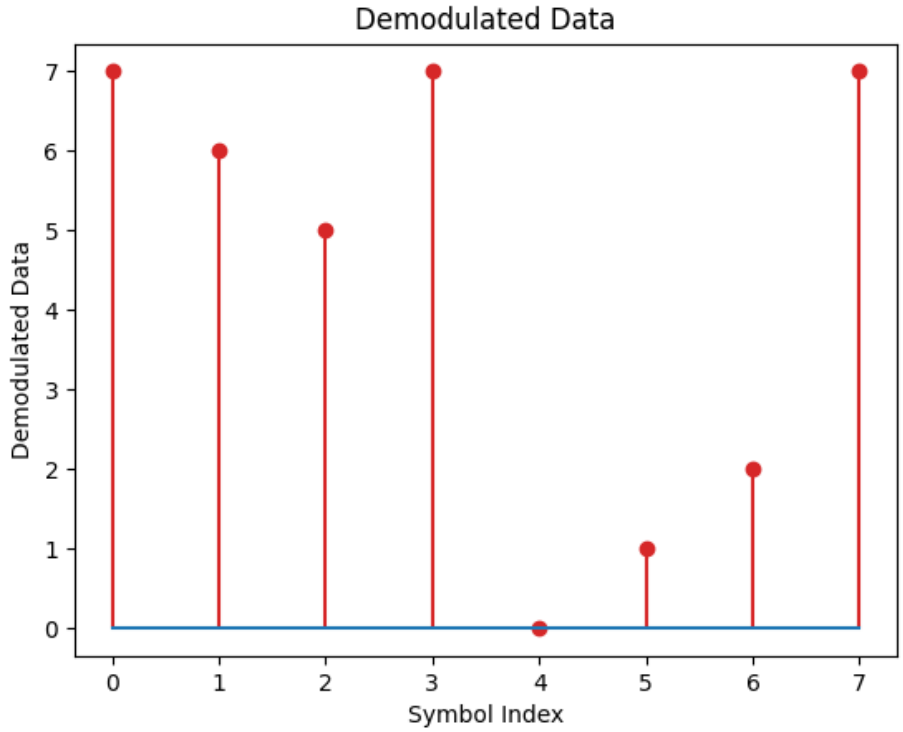
Tabela 2: Tabela zmian wartości błędu bitowego w zależności od stosunku sygnału do szumu dla 4-QAM

SNRdB	1	5	7	10	13
bit error rate	0,85	0,825	0,8	0,775	0,75

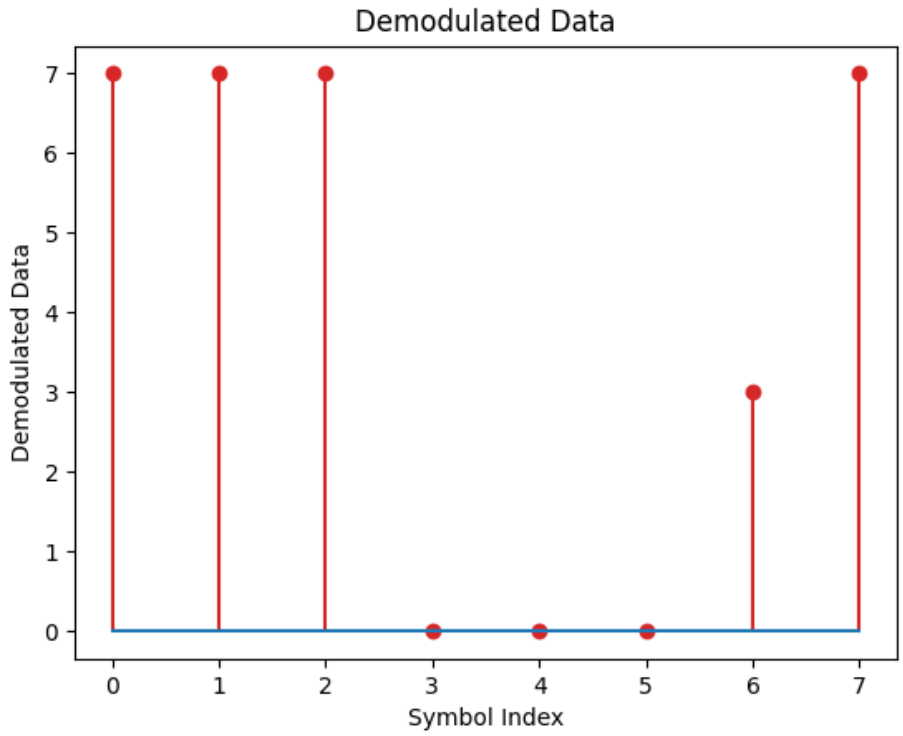
Tabela 3: Tabela zmian wartości błędu bitowego w zależności od stosunku sygnału do szumu dla 16-QAM

SNRdB	1	5	7	10	13
bit error rate	0,969	0,925	0,913	0,906	0,885

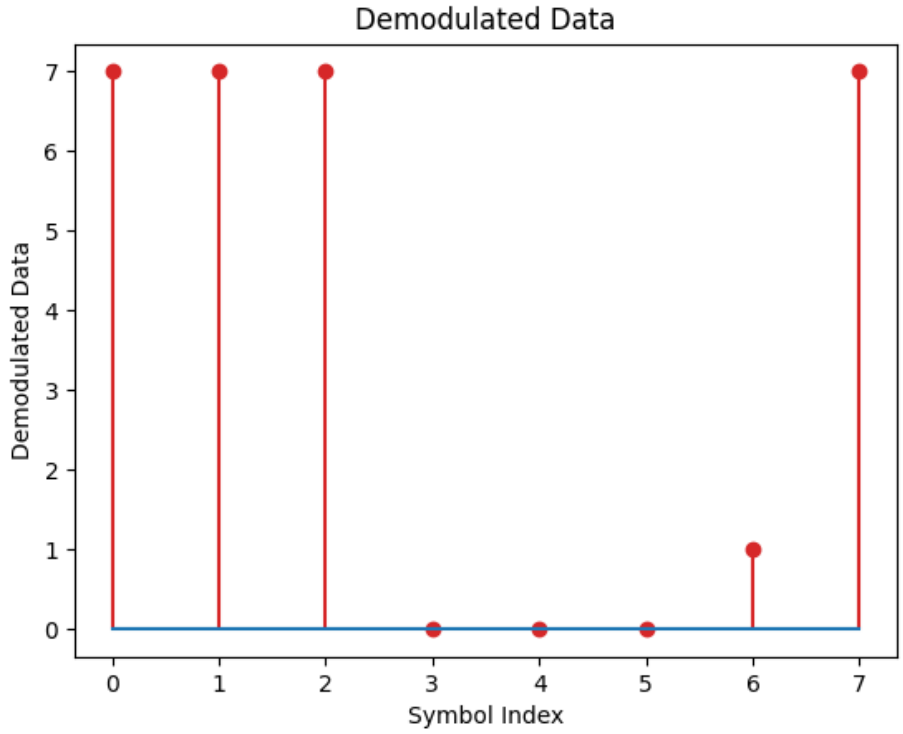
5.2.4 Sygnał zdemodulowany 4QAM dla różnych SNR



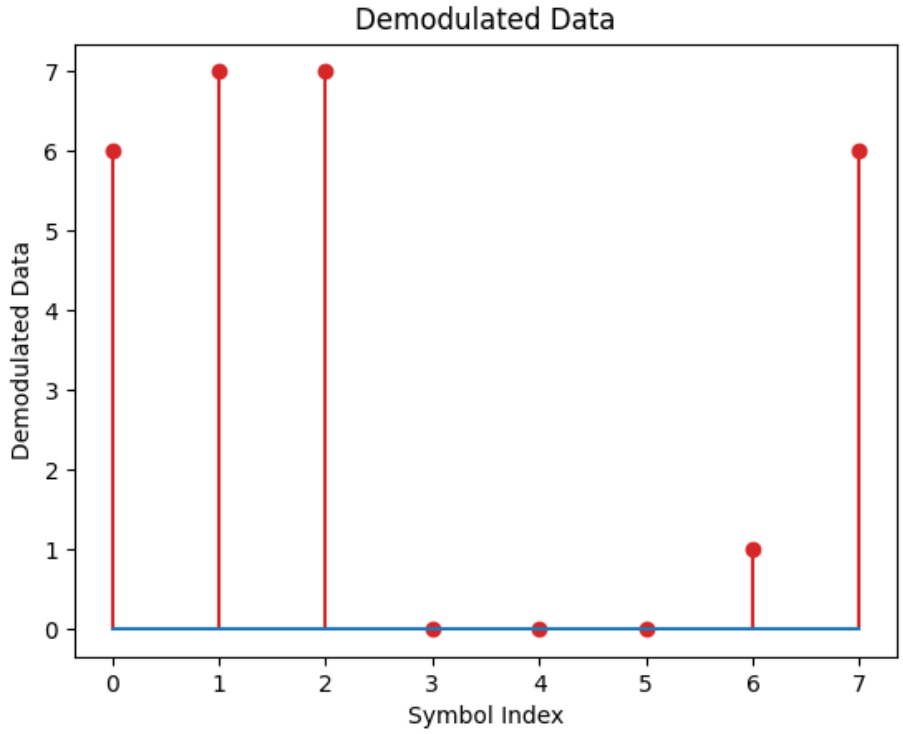
Rysunek 2: 4-QAM: SNR=1dB



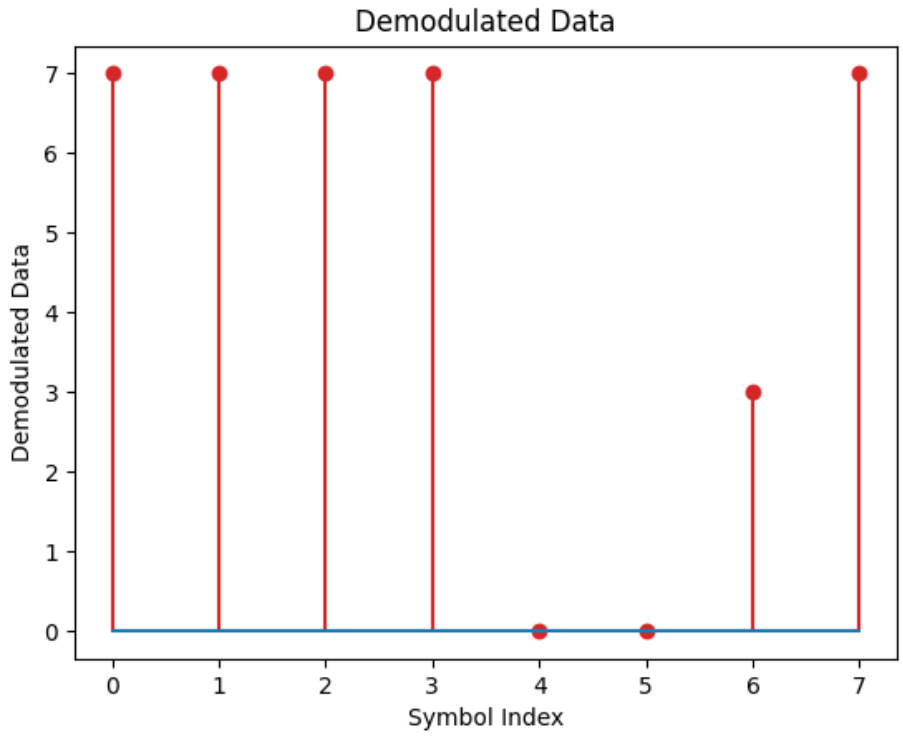
Rysunek 3: 4-QAM: SNR=5dB



Rysunek 4: 4-QAM: SNR=7dB

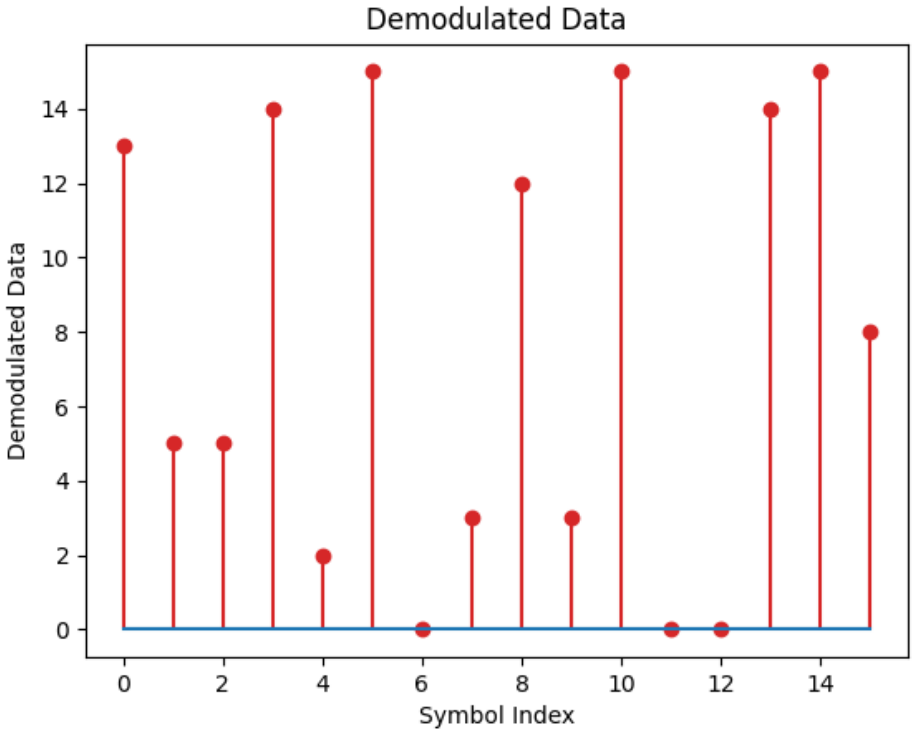


Rysunek 5: 4-QAM: SNR=10dB

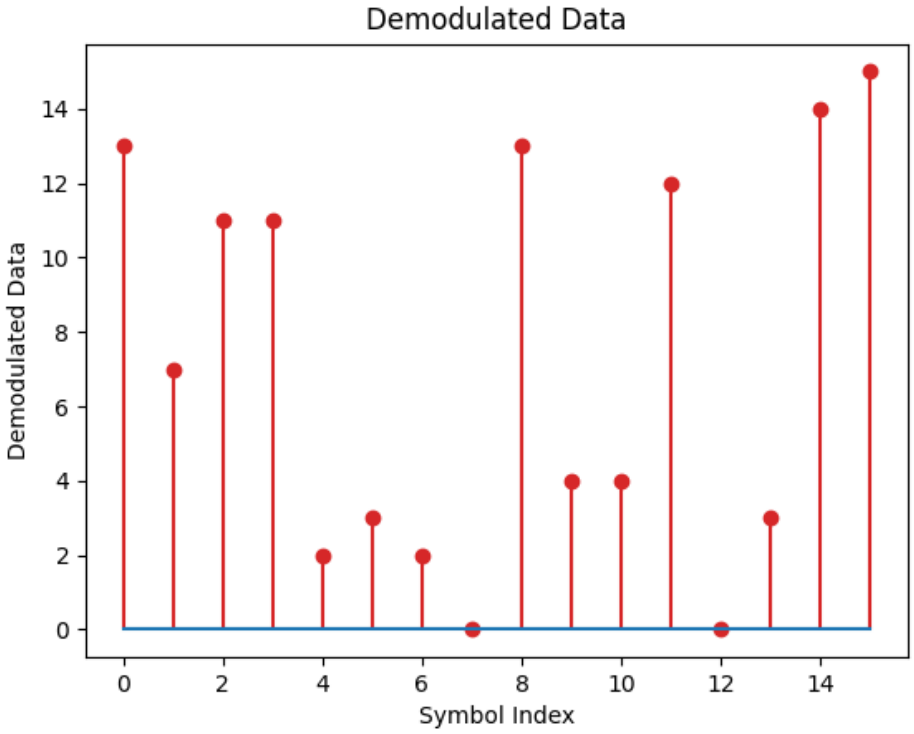


Rysunek 6: 4-QAM: SNR=13dB

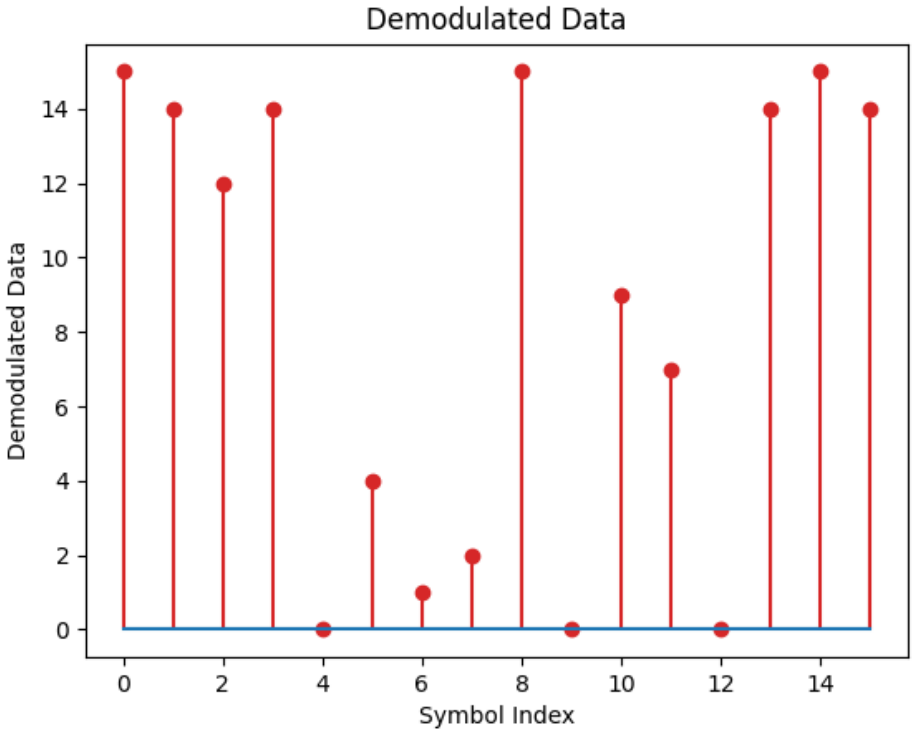
5.2.5 Sygnał zdemodulowany 16QAM dla różnych SNR



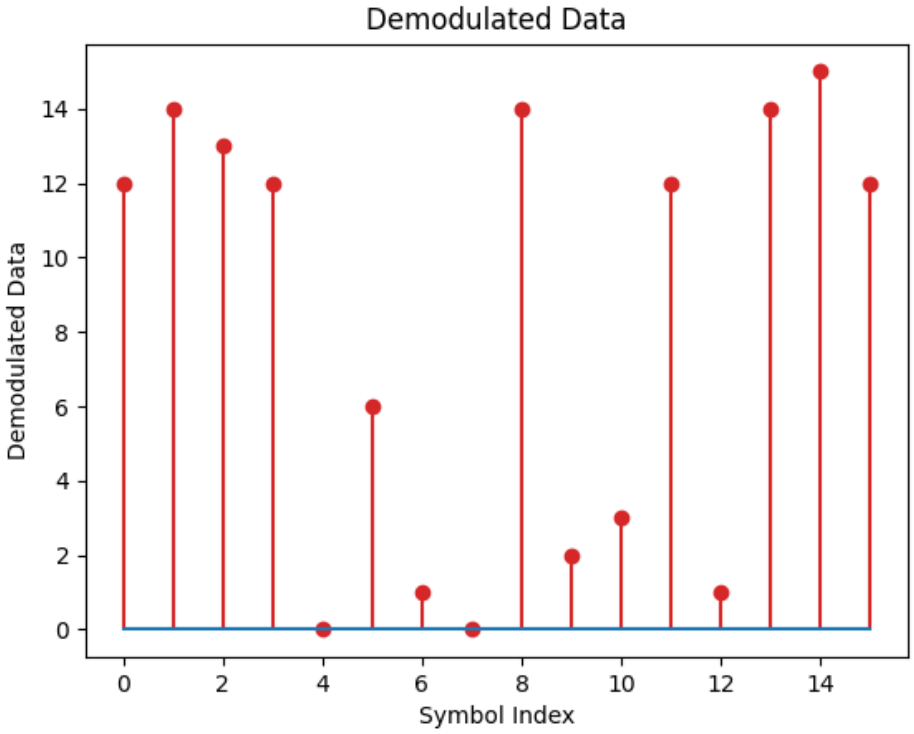
Rysunek 7: 16-QAM: SNR=1dB



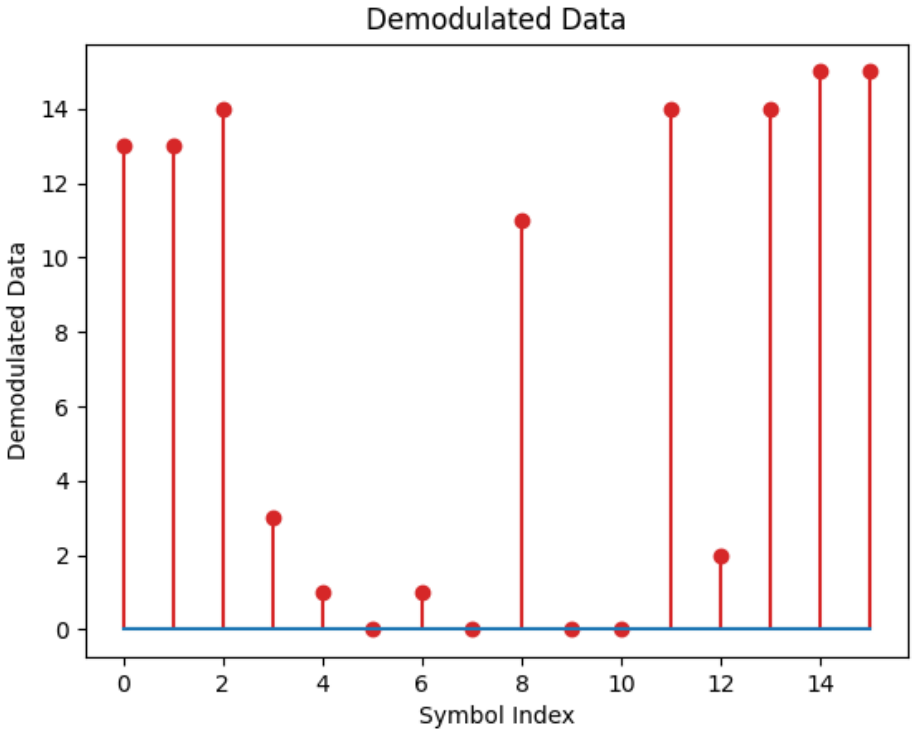
Rysunek 8: 16-QAM: SNR=5dB



Rysunek 9: 16-QAM: SNR=7dB

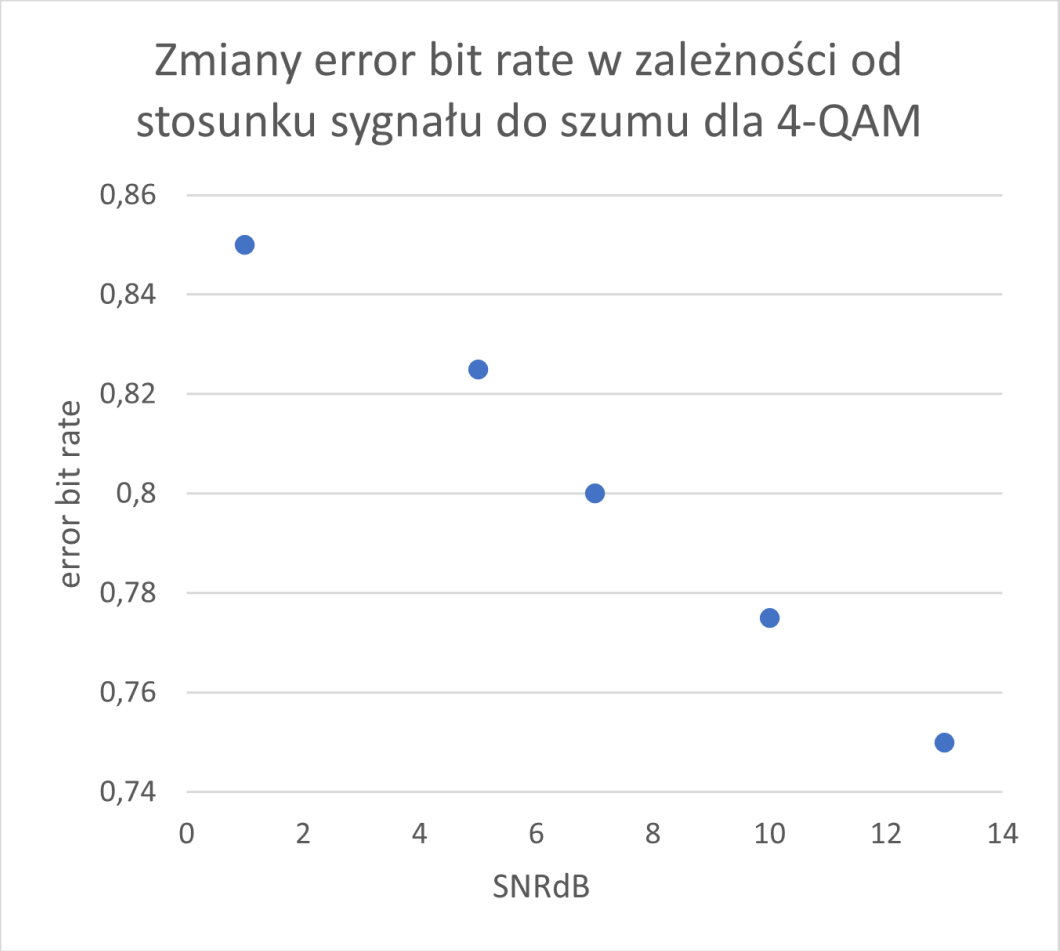


Rysunek 10: 16-QAM: SNR=10dB

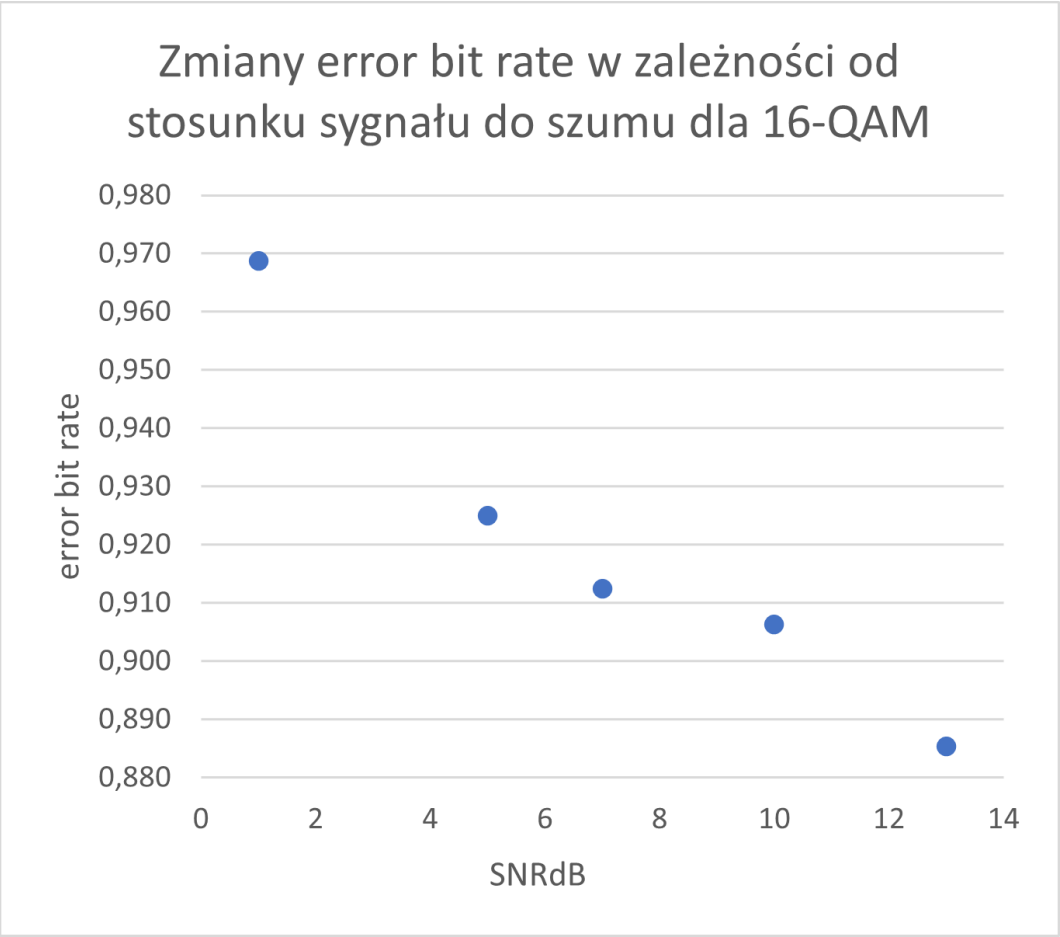


Rysunek 11: 16-QAM: SNR=13dB

5.2.6 Wykresy



Rysunek 12: Wykres zmian wartości błędu bitowego w zależności od stosunku sygnału do szumu dla 4-QAM



Rysunek 13: Wykres zmian wartości błędu bitowego w zależności od stosunku sygnału do szumu dla 16-QAM

6 Wnioski

Przeprowadzone badania pozwoliły na wyciągnięcie wielu wniosków. Testy wpływu zakłóceń na sygnał modulowany FSK wykazały, że error bit rate rośnie wraz ze wzrostem wystąpień cyfry 1 w przesyłanym ciągu bitów. Czterokrotny test dla ośmiu wystąpień bitu 1 udowodnił jednak, że na błąd bitowy nie ma wpływu kolejność rozmeiszczenia bitów o wartości 1 w ciągu wiadomości, a jedynie ich ilość. Oznacza to, że dla niezbalansowanych ciągów bitów, gdzie występuje duża liczba bitów 1 w porównaniu do bitów 0, może występować większe ryzyko błędów. Może być to spowodowane tym, że jeżeli system transmisji ma problemy z dokładnym odtworzeniem częstotliwości nośnej, różnica w częstotliwości między stanami logicznymi może być mniej wyraźna dla większej liczby wystąpień cyfry 1, co może prowadzić do większego błędu w demodulacji. Powód może leżej także po stronie dodanej interferencji: jeśli sygnał z interferencją ma wartości bliskie granicy między stanami logicznymi, większa liczba wystąpień cyfry 1 może prowadzić do większej podatności na błędy w procesie demodulacji.
Badania zarówno dla 4-QAM, jak i 16-QAM potwierdziły oczekiwaną tezę: wraz ze wzrostem SNR maleje błąd bitowy. Im lepszy stosunek sygnału do szumu, tym mniejsze ryzyko błędów w odbiorze i dekodowaniu symboli. Jest to kluczowe przy projektowaniu systemów QAM, gdzie wysoki SNR jest pożądany dla uzyskania wysokiej jakości transmisji.
Badania potwierdziły, że zakłócenia mają negatywny wpływ na jakość i niezawodność transmisji sygnału w modulacjach FSK i QAM. Zakłócenia mogą powodować błędy w odbiorze i dekodowaniu symboli, co prowadzi do niepoprawnej interpretacji przesyłanych danych. W związku z tym, konieczne jest zastosowanie strategii redukcji zakłóceń i technik korekcji błędów w systemach transmisyjnych w celu zapewnienia niezawodnej komunikacji. Wyniki przeprowadzonych testów wskazują na potrzebę zastosowania odpowiednich technik redukcji zakłóceń i korekcji błędów w systemach komunikacyjnych opartych na modulacjach FSK i QAM. Włączenie mechanizmów takich jak kodowanie korekcyjne, filtracja, czy równoważenie mocy sygnału może pomóc w minimalizacji wpływu zakłóceń na transmisję i poprawie jakości odbieranych danych.
Otrzymane wnioski mogą być wykorzystane przy projektowaniu i optymalizacji systemów komunikacyjnych, takich jak sieci telekomunikacyjne czy transmisje danych. Dodatkowo, projekt przyczynił się do poszerzenia mojej wiedzy na temat charakterystyki modulacji FSK i QAM w obecności zakłóceń.

7 Implementacja modulacji i demodulacji -kod

7.1 FSK

```
import numpy as np
import matplotlib.pyplot as plt

#Define the carrier frequency and bit rate
fc = 2402e6 #bluetooth carrier frequency
bit_rate = 1e6 #bluetooth bit rate

#Define the message to be transmitted
message = '1010101010101010'

#Convert the message to binary
binary_message = ''
for char in message:
    binary_message += '{0:08b}'.format(ord(char))

#Define the frequency deviation for each bit
delta_f = 1/8* bit_rate

#Create the time base of the signal
time_step = 1/(1 * fc)
time = np.arange(0, len(binary_message) * 1/bit_rate, time_step)

#Create the carrier signal
carrier = np.cos(2 * np.pi * fc * time)

#Create the modulated signal
modulated_signal = np.zeros_like(carrier)
for i in range(len(binary_message)):
    if binary_message[i] == '0':
        modulated_signal[i*int(1/bit_rate/time_step):
        (i+1)*int(1/bit_rate/time_step)] =
        np.cos(2 * np.pi * (fc - delta_f)
        * time[i*int(1/bit_rate/time_step):(i+1)
        *int(1/bit_rate/time_step)])
    else:
        modulated_signal[i*int(1/bit_rate/time_step)
```

```
        :(i+1)*int(1/bit_rate/time_step)] =
        np.cos(2 * np.pi * (fc + delta_f)
        * time[i*int(1/bit_rate/time_step):(i+1)
        *int(1/bit_rate/time_step)])

#Plot the modulated signal
plt.plot(time, modulated_signal)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Modulated Signal (without interference)')
plt.show()

#Add interference
interference = np.sin(2 * np.pi * 2* delta_f * time)
modulated_signal_with_interference = modulated_signal + interference

#Plot the signal with interference
plt.plot(time, modulated_signal_with_interference)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Modulated Signal (with interference)')
plt.show()

#Perform coherent demodulation
demodulated_signal = np.zeros_like(modulated_signal_with_interference)

for i in range(len(binary_message)):
    if binary_message[i] == '0':
        demodulated_signal[i*int(1/bit_rate/time_step)
        :(i+1)*int(1/bit_rate/time_step)] =
        modulated_signal_with_interference
        [i*int(1/bit_rate/time_step):(i+1)*int(1/bit_rate/time_step)]
        * np.cos(2 * np.pi * (fc - delta_f)
        * time[i*int(1/bit_rate/time_step):(i+1)*int(1/bit_rate/time_step)])
    else:
        demodulated_signal[i*int(1/bit_rate/time_step)
        :(i+1)*int(1/bit_rate/time_step)] =
        modulated_signal_with_interference
        [i*int(1/bit_rate/time_step):
```

```
        (i+1)*int(1/bit_rate/time_step)]
        * np.cos(2 * np.pi * (fc + delta_f)
        * time[i*int(1/bit_rate/time_step):(i+1)*int(1/bit_rate/time_step)])

#Perform low-pass filtering
from scipy import signal

#Define the cutoff frequency for the low-pass filter
cutoff_freq = bit_rate / 2

#Create the low-pass filter
b, a = signal.butter(5, cutoff_freq, fs=1/time_step)

#Apply the low-pass filter to the demodulated signal
filtered_signal = signal.lfilter(b, a, demodulated_signal)

#Calculate the received message
received_message = ''
for i in range(len(binary_message)):
    if filtered_signal[i*int(1/bit_rate/time_step)] > 0:
        received_message += '1'
    else:
        received_message += '0'

#Calculate the error bit rate
error_count = sum(bit1 != bit2 for bit1,
bit2 in zip(binary_message, received_message))
error_bit_rate = error_count / len(binary_message)

#Print the error bit rate
print('Error bit rate:', error_bit_rate)

#Plot the demodulated and filtered signal
plt.plot(time, filtered_signal)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Demodulated and Filtered Signal')
plt.show()
```

7.2 QAM

```
import numpy as np
import matplotlib.pyplot as plt

#Define the carrier frequency and symbol period
fc = 1000  #carrier frequency (Hz)
Tsym = 0.01 #symbol period (seconds)

#Define the constellation points
constellation_4QAM = [1+1j, -1+1j, -1-1j, 1-1j]
constellation_16QAM = [3+3j, 1+3j, -1+3j, -3+3j, 3+1j, 1+1j, -1+1j, -3+1j,
3-1j, 1-1j, -1-1j, -3-1j, 3-3j, 1-3j, -1-3j, -3-3j]

#Define the data to be transmitted
data_4QAM = [0, 1, 2, 3, 0, 1, 2, 3]
data_16QAM = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

#Choose 4-QAM or 16-QAM
constellation = constellation_4QAM
data = data_4QAM

#Time vector
Fs = len(data)/Tsym
t = np.linspace(0, len(data)*Tsym, int(Fs*len(data)))

#Create the modulated signal
sig = np.zeros(len(t), dtype=np.complex64)
for i, sym in enumerate(data):
    sig[i*int(Fs*Tsym):(i+1)*int(Fs*Tsym)] =
    constellation[sym]*np.exp(2j*np.pi*fc*t[i*int(Fs*Tsym)
    :(i+1)*int(Fs*Tsym)])

#Add noise
SNRdB =13 #signal to noise ratio (in dB)
signal_power = np.sum(np.abs(sig)**2)/len(sig)
noise_power = signal_power/(10**(SNRdB/10))
noise = np.sqrt(noise_power/2)*(np.random.randn(len(sig))
+1j*np.random.randn(len(sig)))
```

```
noisy_sig = sig + noise

#Plot the modulated signal and noisy signal
fig, axs = plt.subplots(2)
fig.suptitle('QAM Modulated Signal with Noise')
axs[0].plot(t, sig.real, label='I')
axs[0].plot(t, sig.imag, label='Q')
axs[0].set_xlabel('Time(seconds)')
axs[0].set_ylabel('Amplitude')
axs[0].set_title('Modulated Signal')
axs[0].legend()
axs[1].plot(t, noisy_sig.real, label='I')
axs[1].plot(t, noisy_sig.imag, label='Q')
axs[1].set_xlabel('Time(seconds)')
axs[1].set_ylabel('Amplitude')
axs[1].set_title('Noisy Signal')
axs[1].legend()
plt.show()

#Perform QAM demodulation
demodulated_data = []
for i in range(len(data)):
    symbol = constellation[data[i]]
    segment = noisy_sig[i*int(Fs*Tsym):(i+1)*int(Fs*Tsym)]

    #Perform correlation
    correlation = np.conj(symbol) * segment

    #Find the index of the closest constellation point
    closest_index = np.argmin(np.abs(correlation - symbol))

    #Store the demodulated data
    demodulated_data.append(closest_index)

#Calculate the error bit rate
error_count = sum(bit1 != bit2 for bit1,
bit2 in zip(data, demodulated_data))
error_bit_rate = error_count / len(data)
```

```
#Print the error bit rate
print('Error bit rate:', error_bit_rate)

#Plot the demodulated data
plt.stem(demodulated_data, linefmt='C3-',
markerfmt='C3o', basefmt='C0-')
plt.xlabel('Symbol Index')
plt.ylabel('Demodulated Data')
plt.title('Demodulated Data')
plt.show()
```

8 Bibliografia

1. *Modulacje analogowe i cyfrowe w środowisku Mathcad i Vissim* Krystyna Maria Noga, Akademia Morska w Gdyni

2. *Digital Communications Design for the Real World* Andy Bateman.

3. *Wprowadzenie do cyfrowego przetwarzania sygnałów* Richard G. Lyons

4. Python Software Foundation. *Python Language Reference, version 3.9.*

Spis rysunków