UNIVERSIDADE DO VALE DE ITAJAÍ - UNIVALI

Campus de Itajaí

Ciência da Computação

Redes de Computadores II

Alunos:

Rogério Franchini Borges Júnior

Victor Trindade De Carvalho

Professor(a): Felipe Viel

Camada de Enlace

Itajaí

2023

Descrição do projeto a ser desenvolvido

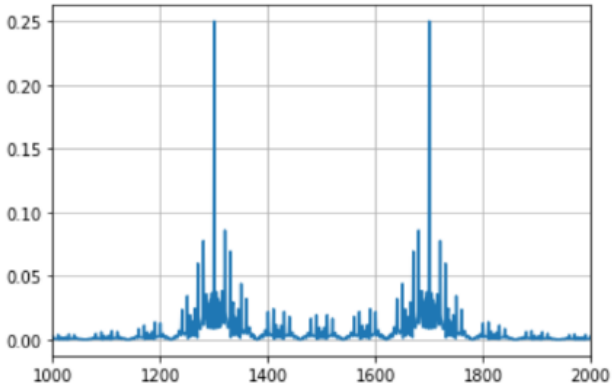
Projeto 1

A comunicação de dados entre equipamentos é uma das grandes impulsionadoras das tecnologias atuais, permite que haja troca de informação por meio de transferência (ou irradiação) de energia elétrica/eletromagnética guiada ou não guiada. Diante do exposto e linkando essa questão com os conceitos abordados em redes de computadores II, camada física e computação, neste trabalho você(s) irá(ão) implementar um algoritmo para entrar em conformidade com o padrão ITU V.23 (link e link).

O padrão V.23 é um padrão criado para moduladores/demoduladores que usam a modulação digital FSK (ou BFSK).

Sendo assim, você irá:

- Ler o sinal modulado. O sinal será criado usando conceitos vistos em aula e com as dicas fornecidas.
- Detectar as frequências usadas e qual modo está sendo usado;



- Baseado nas frequências detectadas, você terá que extrair a informação (1's e 0's) contidas no sinal.
- Transformar a sequência de 1's e 0's em valores ASCII.
    - A mensagem usada é "hello comp"
- E, baseado nisso, gerar uma resposta (pode ser um conjunto de caracteres como "ack" ou algo de escolha).

Dado o contexto do trabalho acima, segue abaixo a demonstração de como foi desenvolvido e após os resultados, análises e discussões.

- **Desenvolvimento da implementação**
  0b1101000011001010110110001101100011011110010000001100011011011101101110000

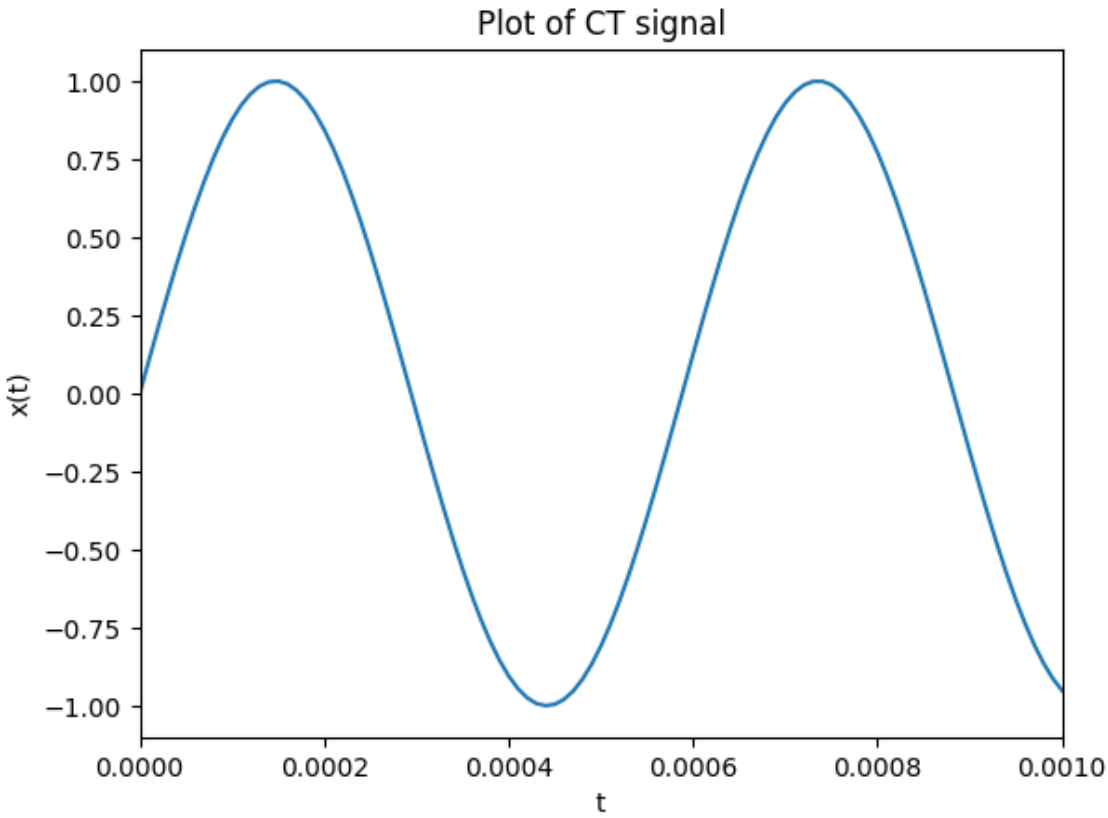


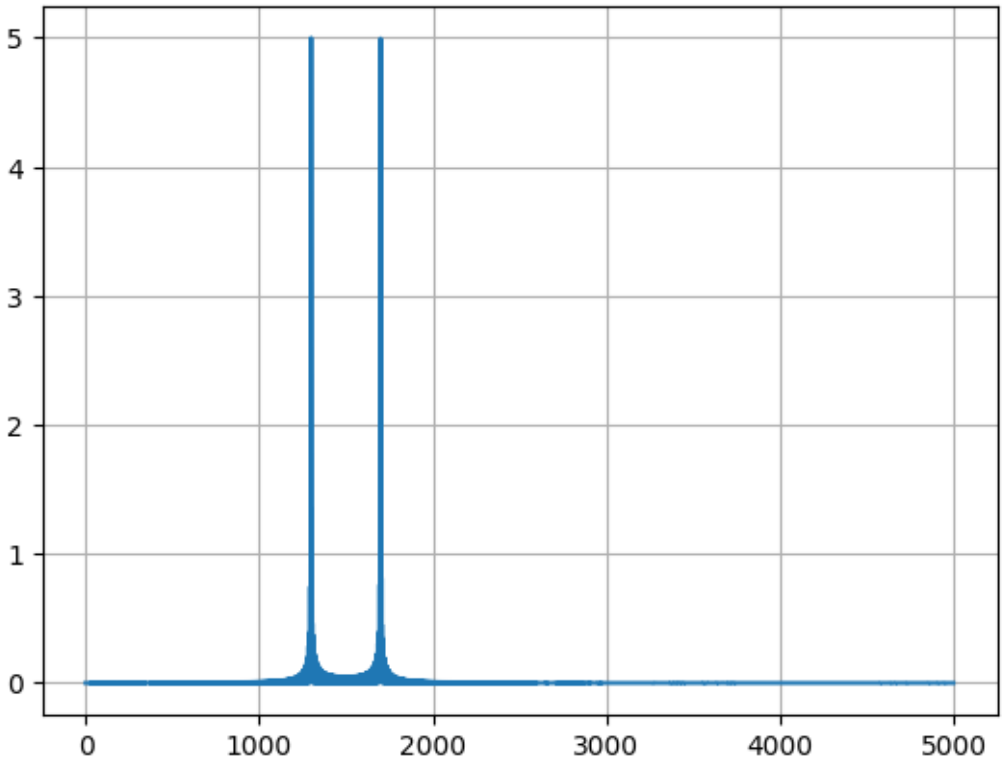
Grafico de modulação

Aqui tem-se feito a modulação do sinal com a mensagem codificada com a função bin que transforma nossa mensagem que se transformou em inteiro, para um binário e a matriz de amostras. Após isso é dividida o vetor com a frequência de cada elemento do vetor e gerado o sinal modulado com a frequência de bits usando a lib *np.hstack*.

```
#construindo mensagem
ascii_message = 'hello comp'
message = bin(int.from_bytes(ascii_message.encode(), 'big'))
print(message)
len_message = len(ascii_message)

#extraindo o b da mensagem
bin_message = '0' + message[2:]

#cria matriz de amostras do tamanho da mensagem
freq_bin_message = np.zeros(len(bin_message))
len(freq_bin_message)

#preenche vetor freq_bin_message com a frequencia de cada elemento
for i in range(len(bin_message)):
    if bin_message[i] == '1':
        freq_bin_message[i] = fc1
    else:
        freq_bin_message[i] = fc2

#criando sinal
signal = np.zeros(0)
t = np.zeros(0)

#dica: usar hstack do numpy, a função cont_sin() passando como parâmentro a frequência que representa o bit na mensagem
#      e incrementar em 0.1 o valor de T para gerar o deslocamento de tempo de bit em 0,1 segundos
for freq in freq_bin_message:
    time, signal_bit = cont_sin(T, Fs, freq)
    T += 0.1
    t = np.hstack([t, time])
    signal = np.hstack([signal, signal_bit])

plt.plot(t, signal)
plt.xlabel('t')
plt.ylabel('x(t)')
plt.title(r'Plot of CT signal')
plt.xlim([0, 0.001])
plt.show()
```

```
N = 80000
T = 1 / 10000

yf = fft(signal)
xf = fftfreq(N, T)[:N // 2]
#import matplotlib.pyplot as plt
plt.plot(xf, 2.0 / N * np.abs(yf[0:N // 2]))
plt.grid()
plt.show()
```

Após isso é usado um período e tamanho definido no código e achado as frequências e amplitudes do sinal com a lib *fftfreq* e depois é exibido o gráfico de modulação abaixo.



Frequências e amplitudes do sinal

0b01101000011001010110110001101100011011110010000001100011
011011110110110101110000
b'hello comp

```
string_demodulada = '0b'  #usar para converter string_demodulada após a conversão de frequências em 0 ou 1
samples_bit = 10000

for bit_position in range(len(freq_bin_message)):
    #capturar uma quantidade de valores do sinal (vetor) dentro do tempo de bit adequado
    signal_result = signal[(samples_bit * bit_position):(samples_bit *
                                                         (bit_position + 1))]


    T = 1 / 100000  # calcular o período de amostragem (1/F sendo F 10000 Hz)
    N = signal_result.size  # pegar a quantidade de amostras do bit

    #aplicar FFT para saber qual é a frequência
    f = fftfreq(len(signal_result), T)
    frequencias = f[:N // 2]
    amplitudes = np.abs(fft(signal_result))[:N // 2] * 1 / N

    #parte onde é verificado se o bit (com uma quantidade de amostras que são usadas para detectar se é 0 ou 1) analisa se é 0 ou 1

    #print(frequencias[np.argmax(amplitudes)])
    if frequencias[np.argmax(amplitudes)] == fc2:
        string_demodulada += '0'
    else:
        string_demodulada += '1'

# Converter Mensagem
print(string_demodulada)
n = int(string_demodulada, 2)
binascii.unhexlify('%x' % n)

print(binascii.unhexlify('%x' % n))
```

Essa mensagem é resultado da demodulação do sinal baseado na amostragem do vetor de bits definido pelo período (1/100000) e também aplicando fftreq para achar frequência e amplitude do sinal. Feito isso, é plotado o gráfico e montado a mensagem demodulada, se o resultado der certo, deverá exibir a mensagem “hello comp”.

- **Análise e discussão sobre os resultados**

Este trabalho teve como escopo, definir as constantes do padrão ITU v.23, como a taxa de baud rate, as frequências das portadoras, a taxa de amostragem, entre outras. Gerar um sinal baseado no padrão ITU v.23, modular o sinal com FSK, aplicar FFT no sinal modulado para obter a frequência e amplitude, demodular o sinal com FSK, obtendo a frequência portadora obtendo sinal original, aplicar fft no sinal demodulado para ver o sinal original no domínio da frequência.

Com os gráficos e a implementação completa, acredita-se que foram analisados e desenvolvidos todos os pontos especificados no trabalho