Санкт-Петербургский Государственный Электротехнический Университет

«ЛЭТИ»

кафедра физики

Задание №4 по дисциплине

«Физические основы информационных технологий» Название: Фильтрация звукового сигнала.

Фамилия И.О.: Токун Г.С. группа: 1303

Преподователь: Альтмарк А.М. Итоговый балл:

Крайник срок сдачи: 05.12.2023

Санкт-Петербург 2023

# Задание

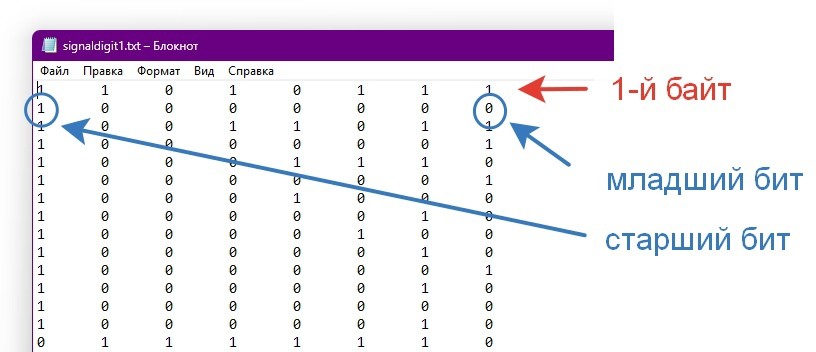
На входе приемника получен звуковой сигнал в двоичном коде (рис.1.). Необ- ходимо перевести двоичный код в десятичный и затем провести над аналоговым сигналом процедуру фильтрации от высокочастотных помех. Для фильтрации необходимо использовать пассивные фильтры (фильтры без дополнительного источника питания), которые могут в себя включать, резисторы, конденсаторы и катушки индуктивности.

Рис. 1: Структура данных в текстовом файле с сигналом

Исходные данные нужно взять в файле FOIT\_IDZ4.xlsx. В отчет нужно включить график сигнала во временной области и его спектр, схему фильтра и АЧХ его передаточной функции, спектр фильтрованного сигнала, а также гра- фик выходного сигнала во временной области. Файл IDZ4.txt должен содержать ответ на вопрос, который записан в звуком сигнале.

Помимо текстового файла IDZ4.txt в папке IDZ4 должен находиться Word- файл с отчетом, а также файл с кодом (Python, Mathcad, Mathematica). Для лучшего понимания отчетности смотрите папку “Пример организации яндекс- папки студентов”. Пример содержания файла IDZ4.txt: 25

# Выполнение работы

Код реализует процесс обработки входного аудиосигнала с использованием

RC-фильтра для устранения помех.

Для этого сначала считывается входной файл с цифровым представлением сигнала из файла ’signaldigit12.txt’. Затем создается аудиофайл ’input.wav’ на основе этого цифрового сигнала.

Далее происходит моделирование входного сигнала и определение его спектра с использованием быстрого преобразования Фурье (ДПФ). Частоты и амплиту- ды спектра входного сигнала отображаются на графиках.

Для применения RC-фильтра вычисляется передаточная функция *H*(*ω*) в

зависимости от угловой частоты *ω*. Эта функция затем применяется к спектру входного сигнала для получения спектра выходного сигнала. Затем выполняется обратное преобразование Фурье для получения выходного аудиосигнала.

На выходе формируются графики входного и выходного сигналов, ампли- туд спектра входного и выходного сигналов, а также амплитуды передаточной функции RC-фильтра. Файл ’output.wav’ представляет собой результат обра- ботки исходного аудиосигнала с использованием RC-фильтра.

# Приложение А

import matplotlib.pyplot as plt  
import numpy as np  
import wave  
import struct  
# считывание файла  
def read\_file(file\_path):  
 with open(file\_path, 'r') as f:  
 data = [int(line.replace("\t", ""), 2) for line in f.readlines()]  
 return data  
  
# стиль  
def set\_plot\_style(title, xlabel, ylabel):  
 plt.title(title)  
 plt.xlabel(xlabel)  
 plt.ylabel(ylabel)  
  
# отрисовка сигнала  
def draw\_signal(signal\_type, time, signal):  
 set\_plot\_style(f'{signal\_type}(t)', 't, c', f'{signal\_type}')  
 plt.plot(time, signal, color='blue')  
 plt.show()  
  
# отрисовка амплитуды  
def draw\_amplitude(amplitude\_type, frequency, amplitude):  
 set\_plot\_style(f'A\_{amplitude\_type}(f)', 'f, Гц', f'A\_{amplitude\_type}')  
 plt.plot(frequency, amplitude, color='green')  
 plt.show()  
  
def calculate\_h(omega):  
 C = 0.000000001  
 R0 = 125  
 Z\_C = 1 / (1j \* omega \* C) # 1j вместо j  
 R = [Z\_C]  
 r\_input = 1  
 n = 255  
 for i in range(1, n):  
 R.append((Z\_C \* (R0 + R[i - 1])) / (Z\_C + R0 + R[i - 1]))  
 for i in range(n):  
 a = 1 + ((R0 + R[i]) / Z\_C)  
 r\_input \*= a  
 r\_input \*= (R0 + (Z\_C \* (R0 + R[-1])) / (Z\_C + R0 + R[-1]))  
 r\_output = Z\_C  
 return r\_output / r\_input  
  
def draw\_afr(frequency, h\_values):  
 set\_plot\_style('|H(if)|', 'f, Гц', '|H|')  
 plt.plot(frequency, [abs(h\_i) for h\_i in h\_values], color='red')  
 plt.show()  
  
def create\_wav(signal, filename):  
 sample\_width = 2  
 sample\_rate = 44100  
 with wave.open(filename, mode="wb") as audio:  
 audio.setnchannels(1)  
 audio.setsampwidth(sample\_width)  
 audio.setframerate(sample\_rate)  
 for value in signal:  
 packed\_value = struct.pack('h', int(value))  
 audio.writeframes(packed\_value)  
  
def calculate\_stop\_i(N, df):  
 stop\_i = 0  
 frequency = [2 \* np.pi \* df \* i for i in range(N)] # Вынесено за пределы цикла  
 for i in range(N):  
 if frequency[i] > 20000 and stop\_i == 0:  
 stop\_i = i + 1  
 return stop\_i  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 j = complex(0, 1)  
 gain = 1000  
 file\_path = 'signaldigit12.txt'  
 signal\_input = read\_file(file\_path)  
 create\_wav(signal\_input, "input.wav")  
 N = len(signal\_input)  
 t = 3.25  
 dt = t / N  
 df = 1 / t  
 time = [dt \* i for i in range(N)]  
 spectre\_input = np.fft.fft(signal\_input)  
 spectre\_input\_module = [abs(number) for number in spectre\_input]  
 frequency = [df \* i for i in range(N)]  
 H = [calculate\_h(\_) for \_ in frequency[1:]]  
 spectre\_input\_sliced = spectre\_input[1:]  
 spectre\_output = [gain \* spectre\_input\_sliced[i] \* H[i] for i in range(N - 1)]  
 spectre\_output\_module = [abs(number) for number in spectre\_output]  
 signal\_output = np.fft.ifft(spectre\_output)  
 signal\_output\_real = [number.real for number in signal\_output]  
 stop\_i = calculate\_stop\_i(N, df)  
 draw\_signal('input', time, signal\_input)  
 draw\_amplitude('input', frequency[:stop\_i], spectre\_input\_module[:stop\_i])  
 draw\_amplitude('output', frequency[1:stop\_i], spectre\_output\_module[:stop\_i - 1])  
 draw\_signal('output', time[1:], signal\_output\_real)  
 draw\_afr(frequency[1:stop\_i], H[:stop\_i - 1])  
 create\_wav(signal\_output\_real, "output.wav")