|  |  |
| --- | --- |
| Группа М32051 | К работе допущен |
| Студент Пантелеев, Артамонов | Работа выполнена |
| Преподаватель Ефремова Е.А. | Отчет принят |

**Рабочий протокол и отчет по  
задаче от лектора №1**

**Задание**

A picture containing text, font, screenshot, document

Description automatically generated

**Решение**

Для создания прямоугольного импульса с длительностью из

плоских гармонических волн с несущей длиной волны , воспользуемся преобразованием Фурье. Преобразование Фурье представляет сигнал в виде непрерывной суммы

гармонических волн различных частот.

Спектральная ширина пакета связана с временной длительностью импульса через соотношение неопределенности Гейзенберга:

Теперь рассмотрим прохождение импульса через среду с заданной дисперсией фазовой скорости:

Для каждого значения b, нам необходимо определить характерное время

расплывания пакета. Оно связано с дисперсией групповой скорости и расстоянием, на котором происходит расплывание. Характерное время расплывания пакета можно определить, используя формулу для дисперсионного сдвига:

Так как , то

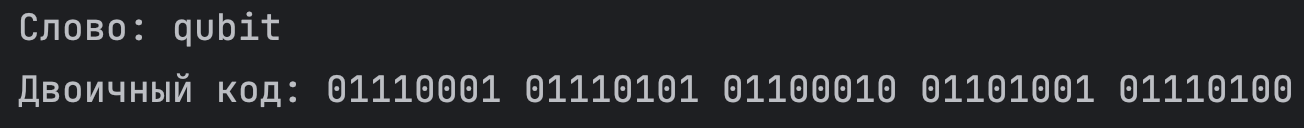
**Результат**

A black background with white text

Description automatically generated with low confidence

Для кодировки короткого сообщения с использованием импульсов будем

использовать метод кодирования с использованием амплитуды импульсов.





**Исходный код**

import matplotlib.pyplot as plt  
  
  
def to\_binary\_str(s):  
 return " ".join(f"{ord(i):08b}" for i in s)  
  
  
def calculate\_delta\_t\_dispersion(c, b, lambda\_0, L, delta\_lambda):  
 D = b \*\* 2 \* lambda\_0 / (c \*\* 2 + b \*\* 2 \* lambda\_0 \*\* 2) \* 0.5  
 delta\_t\_dispersion = (D \* L \* delta\_lambda) / lambda\_0 \*\* 2  
 return delta\_t\_dispersion  
  
  
c = 3e8  
L = 1 # предполагаемая длина среды в метрах  
lambda\_0 = 1.5e-6  
delta\_t = 1e-5  
delta\_lambda = 1 / delta\_t  
for b in [1, 10, 100]:  
 delta\_t\_dispersion = calculate\_delta\_t\_dispersion(c, b, lambda\_0, L, delta\_lambda)  
 print(f"Характерное время расплывания пакета для b = {b}: {delta\_t\_dispersion} с")  
  
word = "qubit"  
binary\_code\_str = to\_binary\_str(word)  
print(f"Слово: {word}")  
print(f"Двоичный код: {binary\_code\_str}")  
  
binary\_code = binary\_code\_str.replace(" ", "")  
  
time\_points = []  
current\_time = 0  
  
L = 1  
b = 1000  
delta\_t\_dispersion = calculate\_delta\_t\_dispersion(c, b, lambda\_0, L, delta\_lambda)  
bit\_time = delta\_t\_dispersion / len(binary\_code) # время для прохождения одного бита  
for bit in binary\_code:  
 time\_points.append(current\_time)  
 time\_points.append(current\_time + bit\_time)  
 current\_time += bit\_time  
  
values = []  
for bit in binary\_code:  
 if bit == "0":  
 values.extend([0, 0])  
 elif bit == "1":  
 values.extend([1, 1])  
  
plt.plot(time\_points, values)  
plt.title(f"Слово: {word}\nКод: {binary\_code}")  
plt.xlabel("Время")  
plt.ylabel("Амплитуда")  
plt.ylim(-0.5, 1.5)  
plt.show()

Исходный код опубликован на GitHub и может быть скачан и запущен:

<https://github.com/SmartOven/Python/blob/master/Physics/LecTask1/main.py>