Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Основы компьютерных сетей

ОТЧЕТ

по лабораторной работе №2

на тему

ПАКЕТНАЯ ПЕРЕДАЧА ДАННЫХ

Студент Т.Ю. Петрович

Преподаватель В.А. Марцинкевич

МИНСК 2024

**1 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

**1.1 Цель работы**

Изучить принципы пакетной передачи данных, разработать алгоритм бит/байт-стаффинга и написать программу для пакетной передачи данных через COM-порты.

**1.2 Исходные данные к работе**

Требования к наполнению программы:

1. Взять за основу программу, относящуюся к лабораторной работе №1.

2. Реализовать структуру пакета -- в данном случае кадра, показанную на рисунке:



Длина поля данных должна быть фиксированной и равной n + 1 байтам, где n -- номер по списку группы. В качестве флага использовать конкатенацию символа '$' и символа со значением 'a' + n В поле Source Address записывать номер передающего COM порта. Поля Destination Address и FCS предусмотреть, но передавать нулевыми.

3. Реализовать алгоритм байт-стаффинга.

Требования к интерфейсу программы:

1. Модифицировать окно состояния. Выводить структуру текущего кадра после приема (до де-бит/байт-стаффинга). Один кадр должен соответствовать одной строке. При этом выделять (любым образом) биты (байты), модифицированные в результате бит/байт-стаффинга.

**2 ОБЗОР ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ**

Для работы с COM-портами на языке программирования Python была выбрана библиотека pySerial.

Для представления и описания COM-порта используется класс serial, релизованный в библиотеки pySerial.

Для инициализации порта используется функция def update\_ports\_direction(\*args).

Для записи данных в порт используется функция send\_data().

Для чтения данных из порта используется функция read\_data()

Для запуска программы и открытие портов используется функция start\_program().

**3 ВЫПОЛНЕНИЕ РАБОТЫ**

**3.1 Обоснование выбранной системы программирования**

Для эмуляции COM-портов в данной работе была выбрана программа Eterlogic Virtual Serial Ports Emulator.

Эта программа предназначена для создания виртуальных COM-портов, которые можно использовать для тестирования и отладки программного обеспечения, взаимодействующего с серийными портами.

Основные особенности и преимущества Eterlogic Virtual Serial Ports Emulator включают:

- Создание виртуальных портов – программа позволяет создавать пары виртуальных COM-портов, которые могут обмениваться данными между собой так же, как физические порты.

- Поддержка различных приложений – эмулятор совместим с большинством программ, которые работают с COM-портами, что делает его полезным инструментом для разработки и тестирования.

Для взаимодействия с COM-портами в программе была использована библиотека pySerial.

Она предоставляет удобный интерфейс для работы с последовательными портами на языке Python, также она поддерживает такие функции, как открытие порта, отправка и приём данных, настройка параметров порта, таких как скорость передачи данных, количество стоп-битов и битов чётности.

Библиотека Tkinter использовалась для создания графического пользовательского интерфейса (GUI) в Python. Она предоставляет инструменты для создания оконных приложений с современным и адаптивным дизайном.

**3.2 Разработка программного обеспечения**

Для реализации пакетной передачи данных была написан класс, представляющая пакет:

class Package  
 flag  
 destianation\_addres   
 source\_address   
 data  
 fcs

Для реализации байт-стаффинга была написана функция func def byte\_stuffing(data). Схема алгоритма представлена на рисунке 3.1.

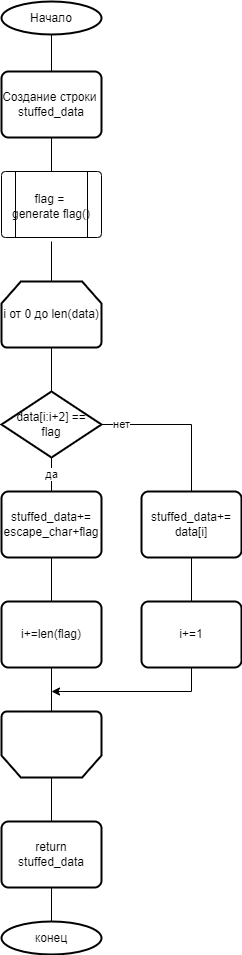


Рисунок 3.1 – Схема алгоритма байт-стаффинга

Для реализации де-байт-стаффинга и достования сегмента данных из пакета была написана функция func def bits\_to\_string(bytes\_array). Схема алгоритма представлена на рисунке 3.2.

Полный код программы представлен в ПРИЛОЖЕНИИ А.

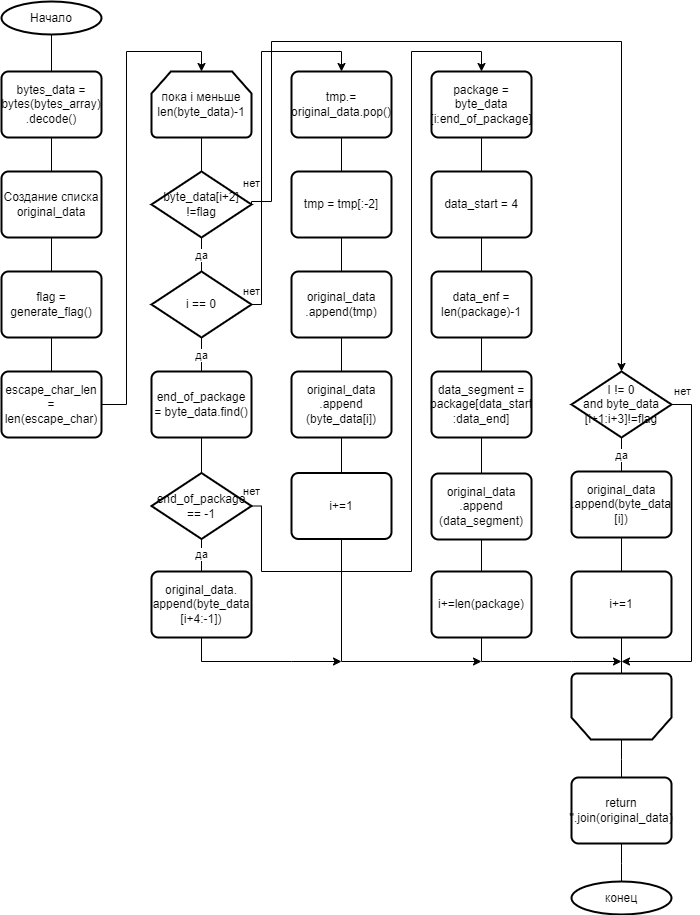


Рисунок 3.2 – Схема алгоритма де-байт-стаффинга

**4 РЕЗУЛЬТАТЫ РАБОТЫ**

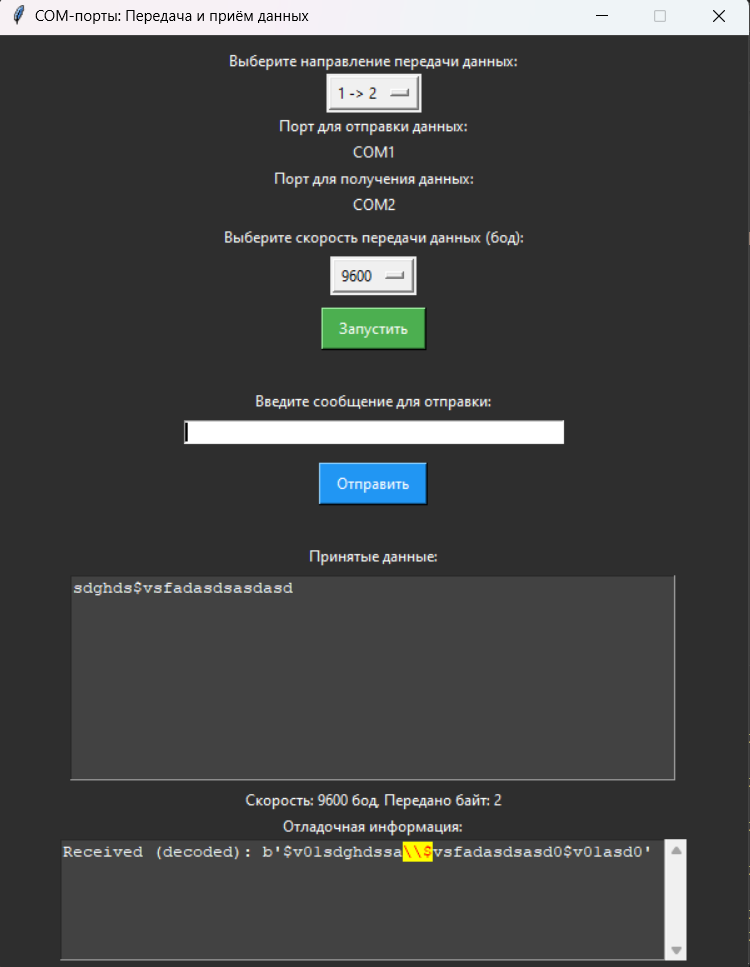
****

Рисунок 4.1 – Пример работы программы

**5 ВЫВОДЫ**

В ходе выполнения лабораторной работы были изучены принципы пакетной передачи данных.

Была реализована структура пакета данных. Также был разработан и внедрен в программу алгоритм байт-стаффинга и де-байт-стаффинга.

Для написания программы был использован язык программирования python с библиотеками pySerial и tkintert, программа эмуляции COM-портов Virtual Serial Ports Emulator.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

[1] pySerial - pySerial 3.4 documentation [Электронный ресурс]. Режим доступа: https://pyserial.readthedocs.io/en/latest/pyserial.html. – Дата доступа: 25.09.2024.

[2] tkinter – Python interface to Tcl/Tk Python 3.12.6 documenatation [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/tkinter.html. – Дата доступа: 25.09.2024.

[3] Вычислительные комплексы, системы и сети: лабораторный практикум для студентов специальности I-40 02 01 «Вычислительные машины, системы и сети» / И. И. Глецевич, Д. В. Ламовский, Д. А.Пынькин. – Минск : БГУИР, 2010. – 36 с. : ил.

# **ПРИЛОЖЕНИЕ А**

(обязательное)

Исходный текст программы

Содержимое файла main.py:

import serial

import threading

import time

from tkinter import Tk, Label, Button, Entry, StringVar, Text, Scrollbar, END, OptionMenu, messagebox, Frame

from packet\_handler import string\_by\_packages, bits\_to\_string

def update\_ports\_direction(\*args):

    selected\_direction = direction\_var.get()

    if selected\_direction == "1 -> 2":

        send\_var.set("COM1")

        receive\_var.set("COM2")

    elif selected\_direction == "5 <- 6":

        send\_var.set("COM6")

        receive\_var.set("COM5")

def send\_data():

    message = string\_by\_packages(entry\_message.get(), ser1.port[3])

    if message:

        try:

            sent\_message = []

            for char in message:

                ser1.write(char)

                sent\_message.append(char)

                time.sleep(0.1)

            update\_state(len(sent\_message))

            entry\_message.set("")

        except serial.SerialException as e:

            messagebox.showerror("Ошибка отправки", f"Ошибка при отправке данных: {e}")

def highlight\_debug\_info():

    start = "1.0"

    while True:

        start = text\_area.search(r'\\\\\$v', start, stopindex=END, regexp=True)

        if not start:

            break

        end = f"{start}+3c"

        text\_area.tag\_add("highlight", start, end)

        start = end

def read\_data():

    received\_data = b''

    while True:

        try:

            if ser2.in\_waiting > 0:

                received\_data += ser2.read(1)

            else:

                if received\_data:

                    try:

                        decoded\_char = bits\_to\_string(received\_data)

                        text\_output.insert(END, decoded\_char)

                        text\_output.insert(END, '\n')

                        text\_area.insert(END, f"Received (decoded): {received\_data}\n")

                        highlight\_debug\_info()

                        received\_data = b''

                    except UnicodeDecodeError:

                        hex\_data = received\_data.hex()

                        text\_output.insert(END, f"[не декодировано: 0x{hex\_data}]")

                        text\_area.insert(END, f"Received (hex): {hex\_data}\n")

                        highlight\_debug\_info()

                        received\_data = b''

                text\_output.see(END)

                text\_area.see(END)

            time.sleep(0.1)

        except serial.SerialException as e:

            messagebox.showerror("Ошибка приёма", f"Ошибка при чтении данных: {e}")

            break

def update\_state(bytes\_sent):

    total\_bytes = 0

    total\_bytes += bytes\_sent

    state\_label.config(text=f"Скорость: {ser1.baudrate} бод, Передано байт: {total\_bytes}")

def start\_program():

    global ser1, ser2

    send\_port = send\_var.get()

    receive\_port = receive\_var.get()

    baudrate = int(baudrate\_var.get())

    try:

        ser1 = serial.Serial(send\_port, baudrate=baudrate, timeout=1)

        ser2 = serial.Serial(receive\_port, baudrate=baudrate, timeout=1)

        threading.Thread(target=read\_data, daemon=True).start()

        update\_state(0)

    except serial.SerialException as e:

        messagebox.showerror("Ошибка порта", f"Не удалось открыть порт: {e}")

root = Tk()

root.title("COM-порты: Передача и приём данных")

root.geometry("600x750")

root.configure(bg="#2E2E2E")

root.resizable(False, False)

frame\_top = Frame(root, bg="#2E2E2E")

frame\_top.pack(pady=10)

frame\_middle = Frame(root, bg="#2E2E2E")

frame\_middle.pack(pady=10)

frame\_bottom = Frame(root, bg="#2E2E2E")

frame\_bottom.pack(pady=10)

direction\_var = StringVar(root)

direction\_var.set("1 -> 2")

direction\_var.trace("w", update\_ports\_direction)

Label(frame\_top, text="Выберите направление передачи данных:", bg="#2E2E2E", fg="white").pack()

direction\_menu = OptionMenu(frame\_top, direction\_var, "1 -> 2", "5 <- 6")

direction\_menu.pack()

send\_var = StringVar(root)

receive\_var = StringVar(root)

Label(frame\_top, text="Порт для отправки данных:", bg="#2E2E2E", fg="white").pack()

send\_label = Label(frame\_top, textvariable=send\_var, bg="#2E2E2E", fg="white")

send\_label.pack()

Label(frame\_top, text="Порт для получения данных:", bg="#2E2E2E", fg="white").pack()

receive\_label = Label(frame\_top, textvariable=receive\_var, bg="#2E2E2E", fg="white")

receive\_label.pack()

Label(frame\_top, text="Выберите скорость передачи данных (бод):", bg="#2E2E2E", fg="white").pack(pady=5)

baudrate\_var = StringVar(root)

baudrate\_var.set("9600")

baudrate\_menu = OptionMenu(frame\_top, baudrate\_var, "9600", "19200", "38400", "57600", "115200")

baudrate\_menu.pack()

start\_button = Button(frame\_top, text="Запустить", command=start\_program, bg="#4CAF50", fg="white", padx=10, pady=5)

start\_button.pack(pady=10)

Label(frame\_middle, text="Введите сообщение для отправки:", bg="#2E2E2E", fg="white").pack()

entry\_message = StringVar()

message\_entry = Entry(frame\_middle, textvariable=entry\_message, width=50)

message\_entry.pack(pady=5)

send\_button = Button(frame\_middle, text="Отправить", command=send\_data, bg="#2196F3", fg="white", padx=10, pady=5)

send\_button.pack(pady=10)

Label(frame\_bottom, text="Принятые данные:", bg="#2E2E2E", fg="white").pack()

text\_output = Text(frame\_bottom, height=10, width=60, bg="#424242", fg="white", insertbackground='white')

text\_output.pack(pady=5)

state\_label = Label(frame\_bottom, text="Скорость: неизвестно, Передано байт: 0", bg="#2E2E2E", fg="white")

state\_label.pack()

Label(frame\_bottom, text="Отладочная информация:", bg="#2E2E2E", fg="white").pack()

text\_area = Text(frame\_bottom, height=10, width=60, bg="#424242", fg="white", insertbackground='white')

scrollbar = Scrollbar(frame\_bottom)

scrollbar.pack(side="right", fill="y")

text\_area.pack(side="left", fill="both")

text\_area.config(yscrollcommand=scrollbar.set)

scrollbar.config(command=text\_area.yview)

text\_area.tag\_config("highlight", background="yellow", foreground="red")

root.mainloop()

if 'ser1' in globals() and ser1.is\_open:

    ser1.close()

if 'ser2' in globals() and ser2.is\_open:

    ser2.close()

Содержимое файла packet\_handler.py:

group\_number = 21

escape\_char = '\\'

class Package:

    def \_\_init\_\_(self, source\_address, data):

        self.flag = generate\_flag()

        self.destination\_address = 0

        self.source\_address = source\_address

        self.data = data

        self.fcs = 0

    def create\_string(self):

        flag\_bytes = self.flag.encode('utf-8')

        destination\_address\_bytes = str(self.destination\_address).encode('utf-8')

        source\_address\_bytes = str(self.source\_address).encode('utf-8')

        stuffed\_data\_bytes = self.data.encode('utf-8')

        fcs\_bytes = str(self.fcs).encode('utf-8')

        package\_bytes = (

            flag\_bytes +

            destination\_address\_bytes +

            source\_address\_bytes +

            stuffed\_data\_bytes +

            fcs\_bytes

        )

        return package\_bytes

def generate\_flag():

    global group\_number

    return f"${chr(ord('a') + group\_number)}"

def byte\_stuffing(data):

    global escape\_char

    stuffed\_data = ""

    flag = generate\_flag()

    i = 0

    while i < len(data):

        if data[i:i + len(flag)] == flag:

            stuffed\_data += escape\_char + flag

            i += len(flag)

            print(stuffed\_data)

        else:

            stuffed\_data += data[i]

            i += 1

    return stuffed\_data

def string\_by\_packages(s, source\_address):

    global group\_number

    packages = []

    s = byte\_stuffing(s)

    for i in range(0, len(s), group\_number + 1):

        data = s[i:i + group\_number + 1]

        package = Package(source\_address, data)

        packages.append(package.create\_string())

    return packages

def bits\_to\_string(bytes\_array):

    byte\_data = bytes(bytes\_array).decode('utf-8', errors='ignore')

    original\_data = []

    flag = generate\_flag()

    escape\_char\_len = len(escape\_char)

    i = 0

    while i < len(byte\_data)-1:

        if byte\_data[i:i + len(flag)] == flag:

            if i == 0 or byte\_data[i - escape\_char\_len:i] != escape\_char:

                end\_of\_package = byte\_data.find(flag, i + len(flag))

                if end\_of\_package == -1:

                    original\_data.append(byte\_data[i+4:-1])

                    break

                package = byte\_data[i:end\_of\_package]

                data\_start = 4

                data\_end = len(package) - 1

                data\_segment = package[data\_start:data\_end]

                original\_data.append(data\_segment)

                i += len(package)

            else:

                tmp = original\_data.pop()

                tmp = tmp[:-2]

                original\_data.append(tmp)

                original\_data.append(byte\_data[i])

                i += 1

        else:

            if(byte\_data[i+1:i+len(flag)+1] != flag and i != 0):

                original\_data.append(byte\_data[i])

            i += 1

    return ''.join(original\_data)