ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра вычислительных машин, комплексов, систем и сетей

М.Е. Солозобов, Н.И. Романчева

ИНТЕРФЕЙСЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Учебно-методическое пособие
по выполнению лабораторных работ

для студентов
специальности 10.05.02
очной формы обучения

Москва
ИД Академии Жуковского
2020

УДК 004
ББК 6Ф7.3
С16

Рецензент:
Терентьев А.И. – канд. техн. наук

Солозобов М.Е.
С16 Интерфейсы вычислительных систем [Текст] : учебно-методическое пособие по выполнению лабораторных работ / М.Е. Солозобов, Н.И. Романчева. – М.: ИД Академии Жуковского, 2020. – 28 с.

Данное учебно-методическое пособие издается в соответствии с рабочей программой учебной дисциплины «Интерфейсы вычислительных систем» для студентов специальности 10.05.02 очной формы обучения.

Рассмотрено и одобрено на заседаниях кафедры 29.08.2020 г. и методического совета 29.08.2020 г.

УДК 004
ББК 6Ф7.3

В авторской редакции

Подписано в печать 10.12.2020 г.
Формат 60х84/16 Печ. л. 1,75 Усл. печ. л. 1,63
Заказ № 704/1008-УМП15 Тираж 90 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д. 20

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (495) 973-45-68
E-mail: zakaz@itsbook.ru

С О Д Е Р Ж А Н И Е

1. ОСНОВНЫЕ ТРЕБОВАНИЯ И ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Настоящее учебно-методическое пособие предназначено для студентов специальности 10.05.02 Информационная безопасность телекоммуникационных систем (специалитет), выполняющих лабораторные работы по дисциплине Интерфейсы вычислительных систем в соответствии с ФГОС3+. В данное учебно-методическое пособие включены материалы для выполнения лабораторных работ № 1-3.

Продолжительность каждой лабораторной работы - 4 часа.

Целью проведения лабораторных работ является закрепление основных теоретических положений, изложенных в лекциях по дисциплине Интерфейсы вычислительных систем.

В процессе выполнения лабораторных работ студенты должны получить практические навыки захвата передаваемых по шине USB данных, получения состава и параметров usb-устройств, навыков низкоуровневой работы с периферийными устройствами.

Лабораторная работа состоит из следующих этапов:

1) домашняя подготовка;

2) выполнение работы на компьютере в соответствии с заданием;

3) сдача выполненной работы преподавателю на персональном компьютере;

4) распечатка результатов работы на принтере;

5) оформление отчета;

6) защита лабораторной работы.

В процессе домашней подготовки студент: изучает лекционный материал, материал по теме лабораторной работы данного учебно-методического пособия и дополнительной литературы; знакомится с заданием на выполнение лабораторной работы;готовит проект отчета по выполнению лабораторной работы.

ВЫПОЛНЕНИЕ лабораторной работы проводится во время занятий в лаборатории кафедры ВМКСС МГТУГА в присутствии преподавателя в соответствии с расписанием, утвержденным проректором по УМР МГТУ ГА. В процессе выполнения лабораторной работы студент последовательно выполняет задание. По завершению работы - демонстрирует преподавателю результаты.

СДАЧА РАБОТЫ преподавателю на персональном компьютере заключается в демонстрации выполненной работы и выполнении непосредственно при преподавателе индивидуального дополнительного задания.

ОТЧЕТ по каждой лабораторной работе должен содержать: название работы; цель лабораторной работы; задание на выполнение лабораторной

работы; краткие комментарии по выполнению лабораторной работы; распечатки файлов результатов или диаграмм, подписанные преподавателем.

ЗАЩИТА лабораторной работы преподавателю проводится по контрольным вопросам и при наличии оформленного отчета (распечатки результатов выполнения лабораторной работы должны быть аккуратно приклеены). После защиты лабораторной работы преподавателем делается соответствующая запись на отчете студента.

РЕЙТИНГОВЫЙ КОНТРОЛЬ по лабораторным работам производится при их сдаче во время лабораторных занятий. Перед выполнением лабораторной работы производится экспресс-опрос для определения готовности студентов к выполнению работы (знания теоретического материала, целей работы и т.д.).

1) Допуск к ЛР

Допуск к выполнению ЛР происходит в виде устного опроса (при условии наличия у студента печатной версии титульного листа отчета по лабораторной работе). Студент, не прошедший устный опрос к выполнению лабораторной работы не допускается.

2) Выполнение и защита ЛР

Минимальная оценка выставляется за выполненную и защищенную лабораторную работу, а дополнительными баллами оценивается качество выполненной лабораторной работы и полнота знаний, показанная студентами при ее защите. Лабораторная работа считается сделанной, если она выполнена полностью в соответствии с заданием.

Сроки сдачи лабораторных работ

Номер лабораторной работы	Лабораторная работа	Срок сдачи*
1	Лабораторная работа №1	Лабораторная работа №1
2	Лабораторная работа №2	Лабораторная работа №2
3	Лабораторная работа №3	Лабораторная работа №3

**Срок сдачи лабораторной работы – в соответствии с расписанием проведения лабораторных работ.*

3) Отчет по ЛР

Отчет по лабораторной работе представляется в печатном виде в формате, предусмотренном шаблоном отчета по лабораторной работе.

4) Защита ЛР

Отчет не может быть принят и подлежит доработке в случае: небрежного выполнения, низкого качества представленного материала (неполное решение, не указаны единицы измерения), отсутствия необходимых разделов отчета, отсутствия необходимого графического материала, отсутствия выводов по работе; некорректной обработки результатов выполнения лабораторной работы, и т.п.

2. ЛАБОРАТОРНАЯ РАБОТА №1 ПЕРЕХВАТ ДАННЫХ НА ШИНЕ USB

2.1. Цель работы

Целью данной работы является получение практических навыков захвата передаваемых по шине USB данных.

2.2. Задание на выполнение работы

1. Изучить принципы работы: программ USBPcap, WireShark, WinHex - для просмотра записанных на диск файлов по кластерам.
2. Реализовать захват данных записи файла на USB диск с использованием USBPcap.
3. Используя WireShark просмотреть файл с командами записи .pcap.
4. Просмотреть записанных на диск файлов по кластерам, используя WinHex.
5. Отразить ход исследования и поиск файла, записанного на диск.
6. Сделать выводы.

2.3. Порядок выполнение работы

2.3.1. Реализация захват данных записи файла на USB диск

С помощью программы USBPcap реализовать захват записи данных на usb накопитель. USBPcap это инструмент USB Packet Capture с открытым исходным кодом для Windows, который может использоваться вместе с Wireshark для анализа трафика USB без использования виртуальной машины. В настоящее время прямой захват можно выполнить на основе «стандартного ввода» в cmd.exe, и Wireshark для захвата необработанного USB-трафика в Windows.

USBPcapDriver включает:

- корневой концентратор (USBPCAP_MAGIC_ROOTHUB);
- управление (USBPCAP_MAGIC_CONTROL);
- устройство (USBPCAP_MAGIC_DEVICE)

Поскольку USBPcap захватывает URB, передаваемые между объектом функционального устройства (FDO) и объектом физического устройства (PDO), есть некоторые элементы связи USB, которые заметны только в аппаратном USB-сниффере:

состояние шины (Suspended, Power ON, Power OFF, Reset, High Speed Detection Handshake);

Идентификатор пакета (PID);

Сплит транзакции (CSPLIT, SSPLIT);

Длительность состояния шины и времени, используемого для передачи пакета по кабелю;

Скорость передачи (низкая скорость, полная скорость, высокая скорость)

Передача управления USB-устройством будет отправлена на устройство после того, как устройству был назначен его адрес.

Иллюстрация процесса представлена на рис. 1.

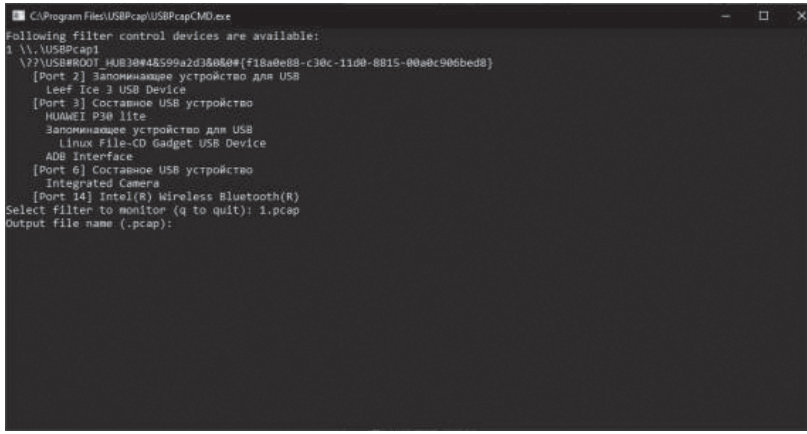

Рисунок 1. Реализация захвата данных записи файла на USB диск.

В результате в папке программы появился файл 1.pcap. Иллюстрация приведена на рис. 2.

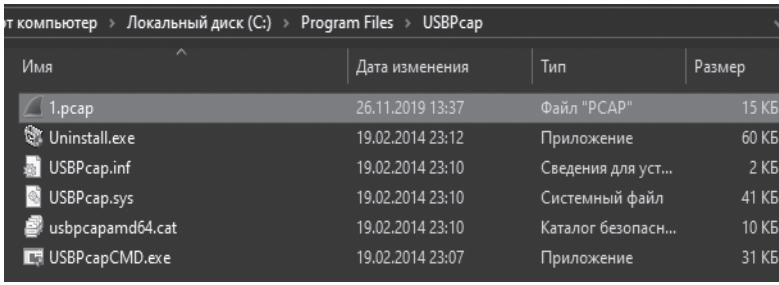

Рисунок 2. Иллюстрация к появлению в папке программы файла 1.pcap.

2.3.2. Ход исследования и поиск файла, записанного на диск

Открыть файл 1.pcap в программе Wireshark.

Найти команду, которая записала изменения в нашем текстовом файле. Иллюстрация процесса представлена на рис. 2 и рис. 3.

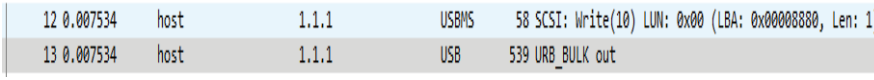

Рисунок 2. Поиск команды, записавшей изменения в нашем текстовом файле

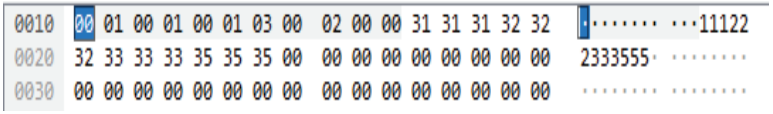

Рисунок 3. Поиск команды, записавшей изменения в файле

По адресу 0x00008880 из этой команды пытаемся найти запись на диске через WinHex, но по этому адресу нет нашей записи. Иллюстрация процесса представлена на рис. 4.

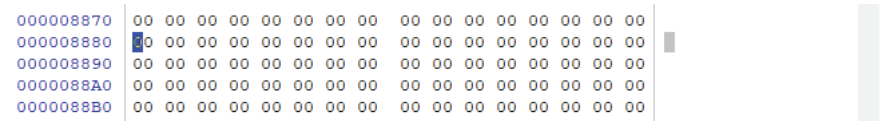

Рисунок 4. Поиск по адресу команды записи на диске.

Если в Wireshark посмотреть на самую первую запись на флэшке, то можно увидеть, что она находится не на нулевом адресе, а на 0x00000800. Иллюстрация процесса представлена на рис. 5.

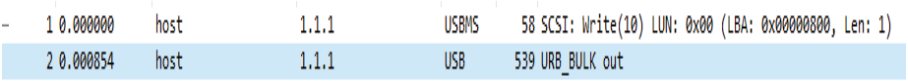

Рисунок 5. Просмотр первой записи на флэшке

Следовательно, в Wireshark все адреса идут со смещением на 800. Попробуем отнять от адреса текста из Wireshark 800, а потом умножить его на 200:

(8880 - 800)*200 = 01010000

Это адрес совпадает с адресом записанного текста в WinHex. Иллюстрация процесса представлена на рис. 6.

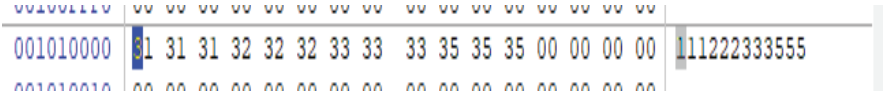

Рисунок 6. Поиск совпадающего адреса

Следующая командная запись в Wireshark – запись в таблице FAT. Иллюстрация процесса представлена на рис. 7.

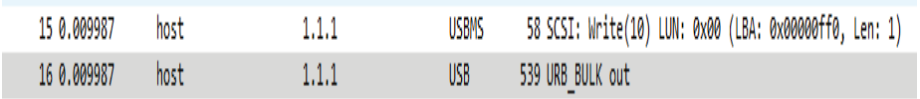

Рисунок 7. Запись в таблице FAT.

Берем адрес 00000ff0, отнимаем 800 и умножаем на 200:

(FF0 – 800) * 200 = FE000

В WinHex по этому адресу мы тоже можем увидеть таблицу FAT. Иллюстрация процесса представлена на рис. 8.

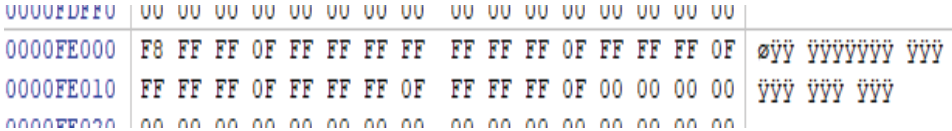

Рисунок 8. Вид таблицы FAT в программе WinHex

2.4. Вопросы к защите лабораторной работы

1) Каким образом происходит запись данных на USB носители?

2) Что представляет собой адрес LBA?

3) Как программа Wireshark интерпретирует адреса LBA?

4) Совпадает ли фактический адрес нахождения записей с адресом из команды записи WireShark?

5) Что понимается под термином «USB-сниффер»?

6) Что включает в себя USBPcapDriver?

7) Что такое FDO?

8) Что такое PDO?

9) Какие элементы связи есть между FDO и PDO?

2.5. Список рекомендуемой литературы

1) Таненбаум Э., Остин Т. Архитектура компьютера. - 6-е изд.- Спб.: Питер, 2018 – 816 с.

2) Ключев А.О., Ковязина Д.Р. Петров Е.В., Платунов А.Е. Интерфейсы периферийных устройств – Санкт-Петербург: СПбГУ ИТМО, 2010. – 294 с.

3. ЛАБОРАТОРНАЯ РАБОТА №2
ИЗУЧЕНИЕ МЕТОДОВ КОДИРОВАНИЯ ИНФОРМАЦИИ НА ФИЗИЧЕСКОМ УРОВНЕ

3.1. Цель работы

Целью данной работы является получение практических навыков для освоения способов физического и логического цифрового кодирования на физическом уровне при передаче данных.

3.2. Задание на выполнение работы

1. Ознакомиться с назначением, классификацией модемов.
2. Подключить модем с помощью стандартного полного кабеля к компьютеру, выполнить настройку параметров для общения с модемом.
3. Воссоздать связь модем + удаленный модем, воспользоваться АТ-командами для настройки синхронизации.
4. Осуществить передачу данных по линии связи.

Закодировать 32-х битную последовательность следующими кодами:

- NRZ;
- AMI;
- NRZI;
- 2B1Q;
- MLT-3;
- биполярным импульсным кодом;
- Манчестерским кодом;

Последовательность бит получить у преподавателя.

5. Выполнить скремблирование исходного кода и представить кодирование по AMI HDB3
6. Выполнить преобразование исходного кода по B8ZS и представить кодирование по AMI.
7. Выполнить преобразование исходного кода по HDB3 и представить кодирование по AMI.
8. Создать программу, написанную на языке программирования высокого уровня C#, C++, VBA, и т.п. позволяющую конвертировать последовательность 4-8 символов в физический или логический код в соответствии с вариантом задания.

Программа должна предоставлять визуально поля: ввода символов, битовые коды в промежуточных состояниях, а также поле вывода результирующей информации как виде двух битовой последовательности, так и в графическом виде.

9. Программный код и скомпилированную программу представить в отчете.
10. Все задания выполнить в виде временных диаграмм в письменном виде.

3.3. Основные теоретические сведения

При цифровом кодировании дискретной информации применяют потенциальные и импульсные коды.

Потенциальные коды для представления логических единиц и нулей используют только значение потенциала сигнала.

Импульсные коды представляют двоичные данные либо импульсами определенной полярности, либо частью импульса - перепадом потенциала определенного направления.

3.3.1. Требования к методам цифрового кодирования

Для передачи дискретной информации с использованием прямоугольных импульсов необходимо выбрать такой способ кодирования, который одновременно достигал бы следующих целей:

- имел при одной и той же битовой скорости наименьшую ширину спектра результирующего сигнала;
- обеспечивал синхронизацию между передатчиком и приемником;
- обладал способностью распознавать ошибки;
- обладал низкой стоимостью реализации.

3.3.2. Физическое кодирование. Потенциальные коды

1) Потенциальный код без возвращения к нулю (Non Return to Zero, NRZ)

Сигнал в линии имеет два уровня: нулю соответствует нижний уровень, единице - верхний. Переходы происходят на границе битового интервала.

Пример кодирования по методу NRZ представлен на рис. 9.

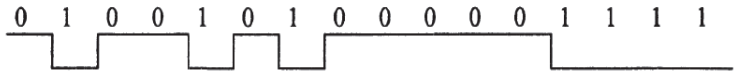

Рисунок 9. Кодирование по методу NRZ

2) Потенциальный код с инверсией при единице NRZI (Non Return to Zero with ones Inverted, NRZI)

При потенциальном кодировании с инверсией при единице (Non Return to Zero with ones Inverted, NRZI) используется только два уровня сигнала. При передаче нуля он передает потенциал, который был установлен в предыдущем такте (то есть не меняет его), а при передаче единицы потенциал инвертируется на противоположный. Он удобен в тех случаях, когда использование третьего уровня сигнала весьма нежелательно, например, в оптических кабелях, где устойчиво распознаются два состояния сигнала - свет и темнота, например, в стандарте Fast Ethernet (100Base-FX).

Код NRZI использует только два уровня сигнала и поэтому обладает хорошей помехоустойчивостью. Максимальную энергию имеют спектральные составляющие сигнала около частоты N/4 (Гц). Пример потенциального кода NRZI с инверсией при единице представлен на рис. 10.

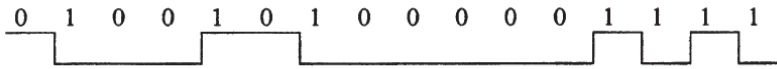

Рисунок 10. Потенциальный код NRZI с инверсией при единице

3) Метод биполярного кодирования с альтернативной инверсией (AMI)

При биполярном кодировании с альтернативной инверсией (Bipolar Alternate Mark Inversion, AMI) используются три уровня потенциала - отрицательный, нулевой и положительный. Для кодирования логического нуля используется нулевой потенциал. Логическая единица кодируется либо положительным потенциалом, либо отрицательным. При этом потенциал каждой новой единицы противоположен потенциалу предыдущей. Пример квазитроичного кодирования (AMI) представлен на рис. 11.

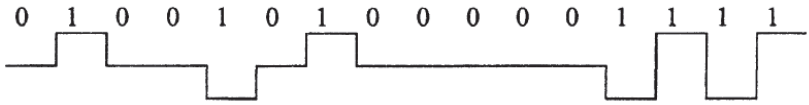

Рисунок 11. Квазитроичное кодирование (AMI)

4) Потенциальный код 2B1Q

Код 2B1Q, название которого отражает его суть — каждые два бита (2B) передаются за один такт (1) сигналом, имеющим четыре состояния (Q — Quadra).

Это потенциальный код с четырьмя уровнями сигнала для кодирования данных. Каждые два бита (2B) передаются за один такт сигналом, имеющим четыре состояния (1Q):

- Паре бит 00 соответствует потенциал -2,5 В.
- Паре бит 01 соответствует потенциал -0,833 В.
- Паре бит 11 соответствует потенциал +0,833 В.
- Паре бит 10 соответствует потенциал +2,5 В.

Таким образом, при данном методе кодирования кодируется не каждый бит информационного исходного сигнала, а два бита. На рис. 12 представлен пример потенциального кода 2B1Q.

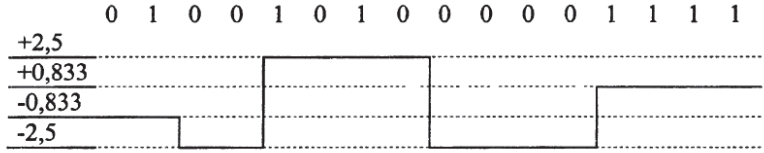

Рисунок 12. Потенциальный Код 2B1Q

5) Код MLT3 (Multi Level Transmission - 3)

Используются три уровня линейного сигнала: «-1», «0», «+1». Логической единице соответствует обязательный переход с одного уровня сигнала на другой. При передаче логического нуля изменение уровня линейного сигнала не происходит. Код MLT3, на первый взгляд, идентичен коду AMI. Разница состоит в том, что при кодировании «1» в AMI используются два уровня «-1» и «+1», в MLT3 три уровня кодировании «-1», «0», «+1». При кодировании «0» в AMI используются «0»- уровень, а при MLT3 также любой из трех «-1», «0», «+1», а именно при «0» в следующем такте уровень не меняется.

6) Биполярный импульсный код

В биполярном импульсном коде единица представлена импульсом одной полярности, а ноль – другой. Каждый импульс длится половину такта. Такой код обладает отличными самосинхронизирующими свойствами, но постоянная составляющая, может присутствовать, например, при передаче длинной последовательности единиц или нулей. На рис.13 представлен пример биполярного импульсного кода.

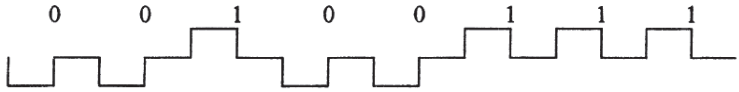

Рисунок 13. Биполярный Импульсный Код

7) Манчестерский код

В локальных сетях до недавнего времени самым распространенным методом кодирования был манчестерский код. Он применяется в технологиях Ethernet и Token Ring. В манчестерском коде для кодирования единиц и нулей используется перепад потенциала, то есть фронт импульса.

При манчестерском кодировании каждый такт делится на две части. Информация кодируется перепадами потенциала, происходящими в середине каждого такта.

Единица кодируется перепадом от низкого уровня сигнала к высокому, а ноль - обратным перепадом.

В начале каждого такта может происходить служебный перепад сигнала, если нужно представить несколько нулей или единиц подряд. На рис.14 представлен пример манчестерского кода.

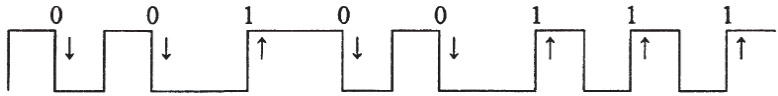

Рисунок14. Манчестерский код.

8) Избыточные коды

Избыточные коды основаны на разбиении исходной последовательности бит на части, которые часто называют символами. Затем каждый исходный символ заменяется на новый, который имеет большее количество бит, чем исходный.

9) Скремблирование

Скремблирование заключается в побитном вычислении результирующего кода на основании битов исходного кода и полученных в предыдущих тактах битов результирующего кода. Например, скремблер может реализовывать следующее соотношение:

$$B_i = A_i \, B_{i-3} \, B_{i-5},$$

где Bi- двоичная цифра результирующего кода, полученная на i-м такте работы скремблера,

Ai- двоичная цифра исходного кода, поступающая на i-м такте на вход скремблера,

Bi-3 и Bi-5 двоичные цифры результирующего кода, полученные на предыдущих тактах работы скремблера (соответственно на 3 и на 5 тактов ранее текущего такта) и объединенные операцией исключающего ИЛИ (сложение по модулю 2).

На рис. 15представлен пример скремблирования кода 1010 0000 0000 1101 и кодирования по коду AMI.

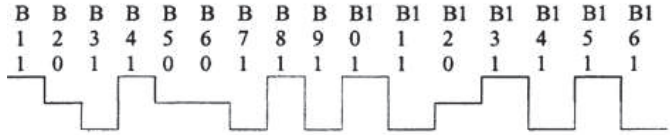

Рисунок 15. Пример скремблирования кода 1010 0000 0000 1101 и кодирования по коду AMI

10) Скремблирование кодами B8ZS и HDB3

Логическое кодирование B8ZS (Bipolar with 8-Zeros Substitution) и HDB3 (High-Density Bipolar 3-Zeros) используются для улучшения кода Bipolar AMI. Они основаны на искусственном искажении последовательности нулей запрещенными символами. На рис.16 представлен пример кодирования B8ZS.

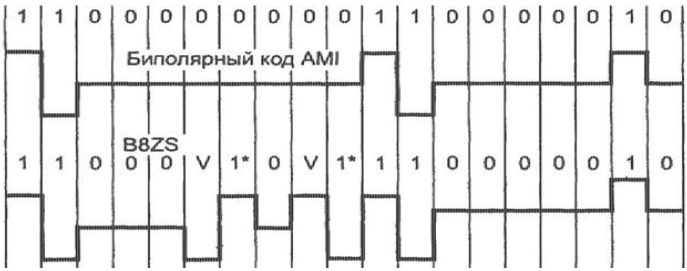

Рисунок 16. Пример кодирования B8ZS

Код B8ZS исправляет только последовательности, состоящие из 8 нулей. При нахождении такой последовательности в исходном коде, она заменяется на

последовательность 0-0-0-V-1*-0-V-1* , где V -сигнал единицы, запрещенной для данного такта полярности, то есть сигнал, не изменяющий полярность предыдущей единицы, 1 -дополнительная единица, вместо исходного нуля. Знак звездочки отмечает тот факт, что в исходном коде в этом такте была не единица, а ноль.

Код HDB3 исправляет любые четыре смежных нуля в исходной последовательности. Правила формирования кода HDB3 более сложные, чем: кода B8ZS. Каждые четыре нуля заменяются четырьмя сигналами, в которых имеется один сигнал V. Для: подавления постоянной составляющей полярность сигнала V чередуется при последовательных заменах. Кроме того, для замены: используются два образца четырех тактовых кодов. Если перед заменой исходный код содержал нечетное число единиц, задействуется последовательность 000V, а если число единиц было четным - последовательность 1*00V.

Улучшенные потенциальные коды обладают достаточно узкой полосой пропускания для любых последовательностей единиц и нулей, которые встречаются в передаваемых данных. Как и выше: 1 -дополнительная единица, вместо исходного нуля. Знак звездочки отмечает тот факт, что в исходном коде в этом такте была не единица, а ноль. На рис.17 представлен пример кодирования HDB3.

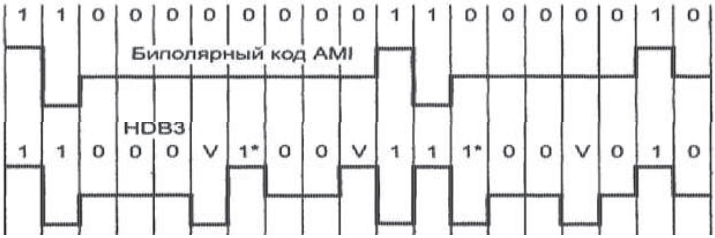

Рисунок 17. Пример кодирования HDB3

Программный код выполнения задания на языке программирования Python 3 представлен ниже.

Листинг программы

```
from tkinter import*

def paint(event):
    nul_count = 0
    one_count = 0
    up = True
    x = 10
    y = 50
    step_x = 20
    step_y = 20
```

```
canvas.delete('all')
b = name.get()
a = []
for i in b:
    lb.pack_forget()
    if i != '0' and i != '1':
        lb.configure(background = "light gray")
        lb.pack()
        return 0
    if i == '0':
        nul_count += 1
 #       a.append(i)
    if i == '1':
        nul_count = 0
        one_count += 1
  #      a.append(i)
    if nul_count == 4:
        if one_count % 2 != 0:
            a[len(a)-3] = '1'
            a.append('v')
            one_count += 2
            nul_count = 0
            continue
        if one_count % 2 == 0:
            a.append('v')
            one_count += 1
            nul_count = 0
            continue
    a.append(i)
print(a)
for i in range(len(a)):
    if a[i] == '0':
        canvas.create_line(x, y, x + step_x, y)
        canvas.create_text(x + 10,80, text = "0")
        x += step_x
    if a[i] == '1':
        if not up:
            canvas.create_line(x, y, x, y + step_y)
            y += step_y
            canvas.create_line(x, y, x + step_x, y)
            canvas.create_text(x + 10,80, text = "1")
            x += step_x
            canvas.create_line(x, y, x, y - step_y)
            y -= step_y
            up = True
            continue
        else:
            canvas.create_line(x, y, x, y - step_y)
            y -= step_y
```

```
                canvas.create_line(x, y, x + step_x, y)
                canvas.create_text(x + 10,80, text = "1")
                x += step_x
                canvas.create_line(x, y, x, y + step_y)
                y += step_y
                up = False
                continue
        if a[i] == 'v':
            if not up:
                canvas.create_line(x, y, x, y - step_y)
                y -= step_y
                canvas.create_line(x, y, x + step_x, y)
                canvas.create_text(x + 10,80, text = "v")
                x += step_x
                canvas.create_line(x, y, x, y + step_y)
                y += step_y
                continue
            else:
                canvas.create_line(x, y, x, y + step_y)
                y += step_y
                canvas.create_line(x, y, x + step_x, y)
                canvas.create_text(x + 10,80, text = "v")
                x += step_x
                canvas.create_line(x, y, x, y - step_y)
                y -= step_y
                continue
root = Tk()
root.geometry("600x150")
root.title("HDB3")
root.configure(background = "light gray")

name = StringVar()

lb = Label(text = "Введите корректные символы - 0 и 1!")
lb.pack_forget()

ent = Entry(root, textvariable = name, justify = RIGHT)
ent.pack()
ent.place(x = 50, y = 0)

canvas = Canvas(root, width = 600, height = 150)
canvas.pack()
canvas.place(x = 0, y = 50)

btn_1 = Button(root, text = "Обработать")
btn_1.configure(background = "light gray", width = 9)
btn_1.pack()
btn_1.place(x = 50, y = 22)
btn_1.bind("<Button-1>", paint)
```

```
root.mainloop()
```

На рис. 18 представлен пример работы программы при введенных данных – последовательности 1100000000110000010.

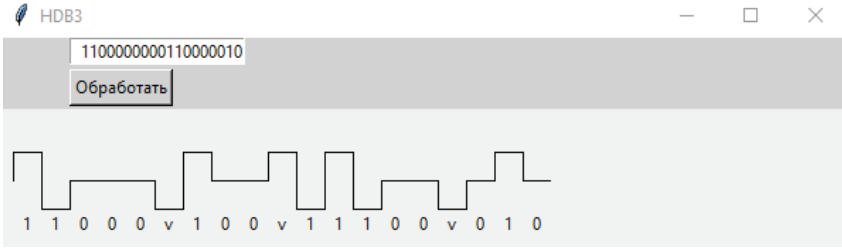

Рисунок 18. Преобразование кода по HDB3

3.4. Вопросы к защите лабораторной работы

1) Какие коды применяются при цифровом кодировании дискретной информации?

2) Какое значение используют потенциальные коды для представления логических единиц и нулей?

3) Сколько уровней уровня сигнала используется при потенциальном кодировании с инверсией?

4) Что такое импульсные коды?

5) На чем основаны избыточные коды?

6) Поясните метод биполярного кодирования.

7) Поясните процедуру скремблирования.

8) Для каких целей используется логическое кодирование B8ZS и HDB3?

9) Какой метод кодирования используется в технологиях Ethernet и Token Ring?

3.5. Список рекомендуемой литературы

1) Таненбаум Э., Остин Т. Архитектура компьютера. - 6-е изд.- Спб.: Питер, 2018 – 816 с.

2) Ключев А.О., Ковязина Д.Р. Петров Е.В., Платунов А.Е. Интерфейсы периферийных устройств – Санкт-Петербург: СПбГУ ИТМО, 2010. – 294 с.

4. ЛАБОРАТОРНАЯ РАБОТА №3
НИЗКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

4.1. Цель работы

Целью данной работы является получение навыков низкоуровневого программирования периферийных устройств с использованием библиотеки libusb.

4.2. Задание на выполнение работы

1. Реализовать программу, получающую список всех подключенных к машине USB устройств с использованием libusb.
2. Для каждого найденного устройства напечатать его класс, идентификатор производителя и идентификатор изделия, основываясь на программе, приведенную в листинге 1.
3. Изучить состав и характеристики, обнаруженных с помощью реализованной программы, USB устройств.
4. Дополнить программу, реализованную в п.2 функцией печати серийного номера USB устройства. Для написания функции рекомендуется использовать функции:

libusb_open, libusb_close, libusb_get_string_descriptor_ascii для печати поля iSerialNumber дескриптора устройства.

5. Составить отчет, содержащий полный компилируемый листинг реализованной программы и команды для ее компиляции.

4.3. Порядок выполнения работы

1) *Реализация программы, получающей список всех подключенных к машине USB устройств.*

Реализация программы, получающей список всех подключенных к машине USB устройств с использованием libusb, представлена ниже.

Листинг программы 1.

```
#include "libusb.h"
#include <stdio.h>
#include <clocale>
#include <string>
#include <iostream>
using namespace std;
void printdev(libusb_device* dev,int j);
int main() {
    setlocale(LC_ALL, "Russian");
    libusb_device** devs;
    libusb_context* ctx = NULL;
```

```
        int r;
        ssize_t cnt;
        ssize_t i;
        r = libusb_init(&ctx);
        if (r < 0) {
                fprintf(stderr,
                        "Ошибка: инициализация не выполнена, код: %d.\n", r);
                return 1;
        }
        libusb_set_debug(ctx, 3);
        cnt = libusb_get_device_list(ctx, &devs);
        if (cnt < 0) {
                fprintf(stderr,
                        "Ошибка: список USB устройств не получен.\n", r);
                return 1;
        }
        printf("найдено устройств: %d\n", cnt);
        printf("================================"
                "================================\n");
        printf("* количество возможных конфигураций\n");
        printf("|     * класс устройства\n");
        printf("|     |     * идентификатор производителя\n");
        printf("|     |     |     * идентификатор устройства\n");
        printf("|     |     |     |     * количество интерфейсов\n");
        printf("|     |     |     |     |     * количество альтернативных настроек\n");
        printf("|     |     |     |     |     |     * класс устройства\n");
        printf("|     |     |     |     |     |     |     * номер интерфейса\n");
        printf("|     |     |     |     |     |     |     |     * количество конечных точек\n");
        printf("|     |     |     |     |     |     |     |     |     * тип дескриптора\n");
        printf("|     |     |     |     |     |     |     |     |     |     * адрес конечной точки\n");
        printf("+-----+------+-----+-----+-----+-----+-----+-----+-----+"
                "-----+-----+----------------------\n");
        for (i = 0; i < cnt; i++) {
                printdev(devs[i],i);
        }
        printf("================================"
                "================================\n");
        libusb_free_device_list(devs, 1);
        libusb_exit(ctx);
        return 0;
}
void printdev(libusb_device* dev,int j) {
        libusb_device_descriptor desc;
```

```
        libusb_config_descriptor* config;
        libusb_device_handle* hand;
        const libusb_interface* inter;
        const libusb_interface_descriptor* interdesc;
        const libusb_endpoint_descriptor* epdesc;
        int h = libusb_get_device_descriptor(dev, &desc);
        if (h < 0) {
                fprintf(stderr,
                        "Ошибка: дескриптор устройства не получен, код: %d.\n", h);
                return;
        }

        int r= libusb_get_config_descriptor(dev, 0, &config);
        if (r < 0) {
                fprintf(stderr,
                        "Ошибка: конфигурация дескриптора устройства не
получена, код: %d.\n", r);
                return;
        }
        printf("| %.2d | %.2d | %.4d|%.4d| %.3d |    |    |    |    |\n",
                (int)desc.bNumConfigurations,
                (int)desc.bDeviceClass,
                desc.idVendor,
                desc.idProduct,
                (int)config->bNumInterfaces
        );
        for (int i = 0; i < (int)config->bNumInterfaces; i++) {
                inter = &config->interface[i];
                printf("|    |    |    |    |    |"
                        "%.2d | %.2d |    |    |    |\n",
                        inter->num_altsetting,
                        (int)desc.bDeviceClass
                );
                for (int j = 0; j < inter->num_altsetting; j++) {
                        interdesc = &inter->altsetting[j];
                        printf("|    |    |    |    |    |    |    |"
                                " %.2d | %.2d |    |\n",
                                (int)interdesc->bInterfaceNumber,
                                (int)interdesc->bNumEndpoints
                        );
                        for (int k = 0; k < (int)interdesc->bNumEndpoints; k++) {
                                epdesc = &interdesc->endpoint[k];
                                printf(
```

```
                                        "|   |   |   |   |   |   |   |   |"
                                        "  %.2d|  %.9d\n",
                                        (int)epdesc->bDescriptorType,
                                        (int)(int)epdesc->bEndpointAddress
                                );
                        }
                }
        }
```

Результат выполнения программы, получающей список всех подключенных к машине USB устройств с использованием libusb, представлен на рис. 19.

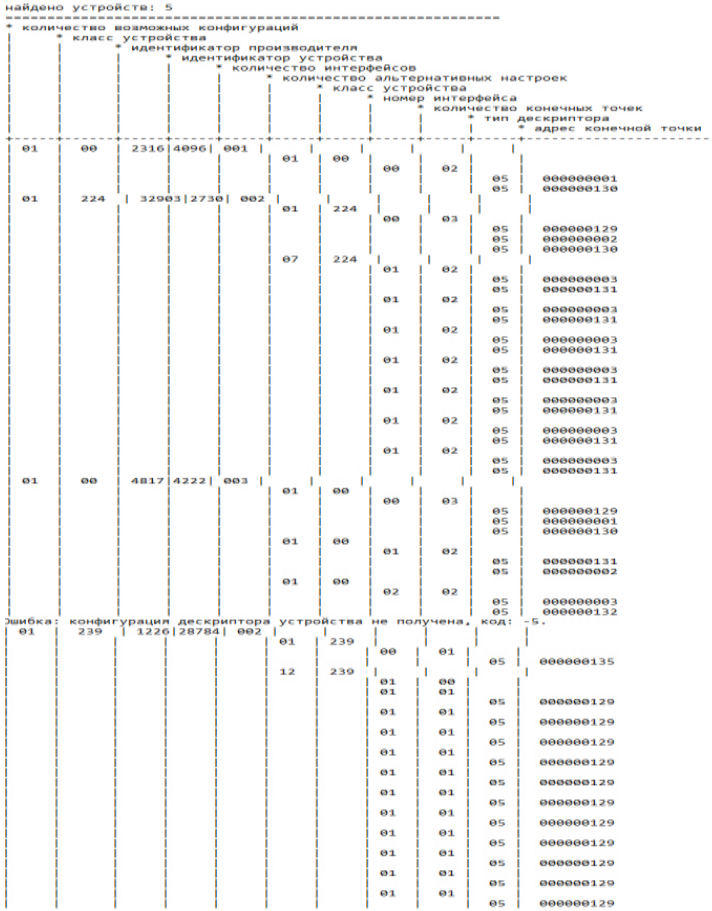

Рисунок 19. Результат выполнения программы

2) Получение информации об обнаруженных устройствах

Для каждого найденного устройства напечатать его класс, идентификатор производителя и идентификатор изделия. Данные о каждом обнаруженном устройстве представлены в табл. 1.

Таблица 1. Данные о найденных с использованием libusb устройствах

Устройство	Кол-во возможных конфигураций	Класс устройства	Идентификатор производителя	Идентификатор устройства	Кол-во интерфейсов
Флеш-накопитель	1	00 – Не задано	2316 – Silicon Motion, Inc. Taiwan	4096	1
Беспроводной контроллер	1	224(E0h) – Wireless Controller	32903 - Intel	2730	2
Телефон	1	00 – Не задано	4817 - Huawei Technologies Co., Ltd.	4222	3
Что – то из внутреннего содержимого ноутбука	1	239(EFh) - Miscellaneous	1226 - Lite-On Technology Corp.	28784	2

3. Дополнение реализованной программы функцией печати серийного номера USB устройства

Для написания функции были использованы функции libusb_open, libusb_close, libusb_get_string_descriptor_ascii для печати поля iSerialNumber дескриптора устройства.

Данные функции выполняют следующие задачи:

- libusb_open() – открытие устройства и получение его дескриптора.
- libusb_close() – закрытие дескриптора устройства.
- libusb_get_string_descriptor_ascii() – извлекает строковой дескриптор устройства в виде строки в ASCII.

В результате работы к первоначальному виду кода необходимо добавить следующий фрагмент:

```
unsigned char* y = {};
r = libusb_open(dev, &hand);
if (r < 0) {
        fprintf(stderr,
                "Ошибка:устройство не открыто, код: %d.\n", r);
        return;
}
```

```
r=libusb_get_string_descriptor_ascii(hand, h, y, 100);
if (r < 0) {
        fprintf(stderr,
                "Ошибка:дескриптор не получен, код: %d.\n", r);
        return;
}
std::cout << y << endl;
libusb_close(hand);
```

Результат выполнения дополненной программы представлен на рис. 20.

```
найдено устройств: 5
============================================================
* количество возможных конфигураций
|     * класс устройства
|     |        * идентификатор производителя
|     |        |      * идентификатор устройства
|     |        |      |     * количество интерфейсов
|     |        |      |     |     * количество альтернативных настроек
|     |        |      |     |     |     * класс устройства
|     |        |      |     |     |     |     * номер интерфейса
|     |        |      |     |     |     |     |     * количество конечных точек
|     |        |      |     |     |     |     |     |     * тип дескриптора
|     |        |      |     |     |     |     |     |     |    * адрес конечной точки
+-----+--------+------+-----+-----+-----+-----+-----+-----+----+---------------------
| 01  |   00   | 2316|4096| 001 |     |     |     |     |    |
|     |        |      |     |     | 01  | 00  |     |     |    |
|     |        |      |     |     |     |     | 00  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000001
|     |        |      |     |     |     |     |     |     | 05 |   000000130
Ошибка:устройство не открыто, код: -12.
| 01  |  224   | 32903|2730| 002 |     |     |     |     |    |
|     |        |      |     |     | 01  | 224 |     |     |    |
|     |        |      |     |     |     |     | 00  | 03  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     |     |     | 05 |   000000002
|     |        |      |     |     |     |     |     |     | 05 |   000000130
|     |        |      |     |     | 07  | 224 |     |     |    |
|     |        |      |     |     |     |     | 01  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000003
|     |        |      |     |     |     |     |     |     | 05 |   000000131
|     |        |      |     |     |     |     | 01  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000003
|     |        |      |     |     |     |     |     |     | 05 |   000000131
|     |        |      |     |     |     |     | 01  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000003
|     |        |      |     |     |     |     |     |     | 05 |   000000131
|     |        |      |     |     |     |     | 01  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000003
|     |        |      |     |     |     |     |     |     | 05 |   000000131
|     |        |      |     |     |     |     | 01  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000003
|     |        |      |     |     |     |     |     |     | 05 |   000000131
|     |        |      |     |     |     |     | 01  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000003
|     |        |      |     |     |     |     |     |     | 05 |   000000131
|     |        |      |     |     |     |     | 01  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000003
|     |        |      |     |     |     |     |     |     | 05 |   000000131
Ошибка:устройство не открыто, код: -12.
| 01  |   00   | 4817|4222| 003 |     |     |     |     |    |
|     |        |      |     |     | 01  | 00  |     |     |    |
|     |        |      |     |     |     |     | 00  | 03  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     |     |     | 05 |   000000001
|     |        |      |     |     |     |     |     |     | 05 |   000000130
|     |        |      |     |     | 01  | 00  |     |     |    |
|     |        |      |     |     |     |     | 01  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000131
|     |        |      |     |     |     |     |     |     | 05 |   000000002
|     |        |      |     |     | 01  | 00  |     |     |    |
|     |        |      |     |     |     |     | 02  | 02  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000003
|     |        |      |     |     |     |     |     |     | 05 |   000000132
Ошибка:дескриптор не получен, код: -2.
Ошибка: конфигурация дескриптора устройства не получена, код: -5.
| 01  |  239   | 1226|28784| 002 |    |     |     |     |    |
|     |        |      |     |     | 01  | 239 |     |     |    |
|     |        |      |     |     |     |     | 00  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000135
|     |        |      |     |     | 12  | 239 |     |     |    |
|     |        |      |     |     |     |     | 01  | 00  |    |
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
|     |        |      |     |     |     |     | 01  | 01  |    |
|     |        |      |     |     |     |     |     |     | 05 |   000000129
Ошибка:устройство не открыто, код: -5.
============================================================
```

Рисунок 20. Результат выполнения программы, дополненной функцией печати серийного номера USB устройства

Листинг программы, дополненной функцией печати серийного номера USB устройства представлен ниже:

Листинг программы 2.

```
#include "libusb.h"
#include <stdio.h>
#include <clocale>
#include <string>
#include <iostream>
using namespace std;

void printdev(libusb_device* dev,int j);
int main() {
	setlocale(LC_ALL, "Russian");
	libusb_device** devs;
	libusb_context* ctx = NULL;
	int r;
	ssize_t cnt;
	ssize_t i;
	r = libusb_init(&ctx);
	if (r < 0) {
		fprintf(stderr,
			"Ошибка: инициализация не выполнена, код: %d.\n", r);
		return 1;
	}
	libusb_set_debug(ctx, 3);
	cnt = libusb_get_device_list(ctx, &devs);
	if (cnt < 0) {
		fprintf(stderr,
			"Ошибка: список USB устройств не получен.\n", r);
		return 1;
	}
	printf("найдено устройств: %d\n", cnt);
	printf("================================"
		"================================\n");
	printf("* количество возможных конфигураций\n");
	printf("|     * класс устройства\n");
	printf("|     |     * идентификатор производителя\n");
	printf("|     |     |     * идентификатор устройства\n");
	printf("|     |     |     |     * количество интерфейсов\n");
	printf("|     |     |     |     |     * количество альтернативных настроек\n");
	printf("|     |     |     |     |     |     * класс устройства\n");
	printf("|     |     |     |     |     |     |     * номер интерфейса\n");
	printf("|     |     |     |     |     |     |     |     * количество конечных точек\n");
```

```
        printf("|    |    |    |    |    |    |    |    * тип дескриптора\n");
        printf("|    |    |    |    |    |    |    |    |    * адрес конечной точки\n");
        printf("+-----+------+-----+-----+-----+-----+-----+-----+"
                "-----+-----+----------------------\n");
        for (i = 0; i < cnt; i++) {
                printdev(devs[i],i);
        }
        printf("================================"
                "================================\n");
        libusb_free_device_list(devs, 1);
        libusb_exit(ctx);
        return 0;
}

void printdev(libusb_device* dev,int j) {
        libusb_device_descriptor desc;
        libusb_config_descriptor* config;
        libusb_device_handle* hand;
        const libusb_interface* inter;
        const libusb_interface_descriptor* interdesc;
        const libusb_endpoint_descriptor* epdesc;
        int h = libusb_get_device_descriptor(dev, &desc);
        if (h < 0) {
                fprintf(stderr,
                        "Ошибка: дескриптор устройства не получен, код: %d.\n", h);
                return;
        }

        int r= libusb_get_config_descriptor(dev, 0, &config);
        if (r < 0) {
                fprintf(stderr,
                        "Ошибка: конфигурация дескриптора устройства не
получена, код: %d.\n", r);
                return;
        }
        printf("| %.2d | %.2d | %.4d|%.4d| %.3d|    |    |    |    |\n",
                (int)desc.bNumConfigurations,
                (int)desc.bDeviceClass,
                desc.idVendor,
                desc.idProduct,
                (int)config->bNumInterfaces
        );
        for (int i = 0; i < (int)config->bNumInterfaces; i++) {
```

```
            inter = &config->interface[i];
            printf("|    |    |    |    |    |"
                    "%.2d | %.2d |    |    |    |\n",
                    inter->num_altsetting,
                    (int)desc.bDeviceClass
            );
            for (int j = 0; j < inter->num_altsetting; j++) {
                    interdesc = &inter->altsetting[j];
                    printf("|    |    |    |    |    |    |"
                            " %.2d |  %.2d |    |\n",
                            (int)interdesc->bInterfaceNumber,
                            (int)interdesc->bNumEndpoints
                    );
                    for (int k = 0; k < (int)interdesc->bNumEndpoints; k++) {
                            epdesc = &interdesc->endpoint[k];
                            printf(
                                    "|    |    |    |    |    |    |    |    |"
                                    "  %.2d |  %.9d\n",
                                    (int)epdesc->bDescriptorType,
                                    (int)(int)epdesc->bEndpointAddress
                            );
                    }
            }
    }


    unsigned char* y = {};
    r = libusb_open(dev, &hand);
    if (r < 0) {
            fprintf(stderr,
                    "Ошибка:устройство не открыто, код: %d.\n", r);
            return;
    }
    r=libusb_get_string_descriptor_ascii(hand, h, y, 100);
    if (r < 0) {
            fprintf(stderr,
                    "Ошибка:дескриптор не получен, код: %d.\n", r);
            return;
    }
    std::cout << y << endl;
    libusb_close(hand);
}
```

Перечень возникающих ошибок и расшифровка кодов ошибок libusb: 2 – неверный параметр, 5 – объект не найден, 12 – операция не поддерживается или не реализована на данной платформе.

4.4. Вопросы к защите лабораторной работы

1. Что такое libusb?
2. Какие задачи решает libusb?
3. На какие группы можно разбить функции libusb?
4. Как с помощью библиотеки libusb получить состав и конфигурацию USB-устройств?
5. Поясните коды ошибок libusb.

4.5. Список рекомендуемой литературы

1) Таненбаум Э., Остин Т. Архитектура компьютера. - 6-е изд.- Спб.: Питер, 2018 – 816 с.

2) Внешние устройства Руководство пользователя Компания Hewlett-Packard Development (Hewlett-Packard Development Company, L.P.), 2006.