Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**
**(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)**

БГТУ.СМК-Ф-4.2-К5-01

Факультет О Естественнонаучный
шифр Наименование

Кафедра О7 Информационные системы и программная инженерия
шифр Наименование

Дисциплина Системное программное обеспечение

КУРСОВАЯ РАБОТА
на тему

# Разработка программного обеспечения, обеспечивающего удаленное управление периферийным устройством по локальной сети

Выполнил студент группы И407Б
Альков В.С..
Фамилия И.О.

**РУКОВОДИТЕЛЬ**
Седелкин В.А.
Фамилия И.О. Подпись

Оценка

«_____» 2022 г.

Санкт-Петербург
2022г.

**СОДЕРЖАНИЕ**

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете применяются следующие сокращений и обозначения.

**IP** — Internet Protocol ( IP, досл. «межсетевой протокол») — маршрутизируемый протокол сетевого уровня стека TCP/IP. Именно IP стал тем протоколом, который объединил отдельные компьютерные сети во всемирную сеть Интернет.

**Modbus** — открытый коммуникационный протокол, основанный на архитектуре ведущий — ведомый ( англ. master-slave; в стандарте Modbus используются термины client-server). Широко применяется в промышленности для организации связи между электронными устройствами.

**TCP** — Transmission Control Protocol (TCP, протокол управления передачей) — один из основных протоколов передачи данных интернета.

# ВВЕДЕНИЕ

Темой данной курсовой работы является разработка программного обеспечения, которое обеспечивает удаленное управление периферийным устройством по локальной сети. В современном мире существует необходимость создания приложений для быстрой и максимально эффективной передачи данных по сети, так как на сегодняшний день огромное количество компьютеров и различных устройств связаны либо через локальные сети, либо по глобальной сети интернет. Высокоуровневые языки программирования C и C++ предоставляют возможности для создания приложений, взаимодействующих по сети и использующих различные сетевые протоколы передачи данных.

Сети состоят из компьютеров-хостов, взаимодействующих друг с другом. Они соединены между собой каналами связи (Wi-Fi, Ethernet и т.д.). Также в соединении важную роль играют маршрутизаторы: они передают пакеты данных между различными элементами сети на основе правил и таблиц маршрутизации.

Все это взаимодействие не может происходить в свободной или каждый раз разной форме: это неудобно и неэффективно. Поэтому для этого применяются протоколы - некие соглашения в каком виде и как будут передаваться пакеты информации.

В наше время существует огромное количество протоколов, такой как TCP/IP, позволяющий надежно передавать данные в “межсетевом пространстве”, IP – протокол, первый объединивший отдельные компьютеры в единую сеть и так далее.

В данной курсовой работе применяется протокол Modbus TCP, широко используемый в промышленности для организации связи между электронными устройствами.

Таким образом, целью данной курсовой работы является разработка клиент-серверного приложения для удаленного управления периферийным устройством и получения информации с его составляющих.

Для выполнения поставленной цели, необходимо выделить ряд задач:

1) Разработать серверную часть приложения, которая должна:

    а. Принимать данные и управляющие команды от программы – клиента по локальной сети через подмножество протокола Modbus TCP.

    б. Обеспечивать управление периферийным устройством USB-HID, осуществляя запись и чтение данных в соответствии с принятыми от программы – клиента командами.

2) Разработать клиентскую часть приложения, которая должна:

    а. Осуществлять взаимодействие с пользователем.

    б. Подключаться к серверу в локальной сети и передавать ему на выполнение команды по протоколу Modbus TCP.

3) Разработать интерпретатор для протокола Modbus TCP

## 1. Постановка задачи

Задача курсовой работы заключается в разработке клиент-серверного приложения для удаленного взаимодействия с периферийным устройством по локальной сети, использующего протокол передачи данных Modbus TCP.

Разработка приложения будет происходить на высокоуровневом языке программирования C. Использование библиотеки HIDAPI позволяет приложению взаимодействовать с устройствами USB и Bluetooth HID-класса в Windows, Linux, FreeBSD и MacOS X. Лицензия GNU Lesser General Public License version 3 разрешает копировать и распространять дословные копии этого лицензионного документа, не внося изменения.

На вход программы будут попадать текстовые данные, введенные пользователем. Результаты запросов (сообщения, список возможных команд и т.д.) должны выводится в понятном и удобном для чтения формате для последующего использования пользователем.

## 2. Описание модели Modbus

Модель протокола Modbus TCP представлена в виде двух структур: структуры modbus с объявлением полей для передачи данных в соответствии с протоколом и использованием методов API Modbus TCP и структуры package, где объявляется экземпляр структуры modbus. Структура package предназначена для формирования пакетов, в дальнейшем принятия и отправки пакетов.

Композиция моделей представлена на рисунке 1.

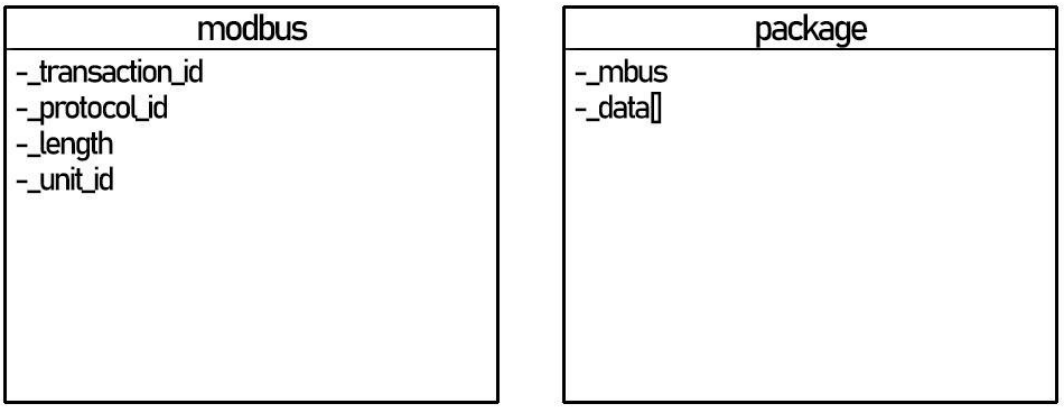


Рисунок 1 – Диаграмма структур протокола Modbus TCP

**Структура «modbus»**

Структура modbus  предназначена для объявления данных протокола Modbus TCP.

**Структура «package»**

Структура package содержит в себе экземпляр структуры modbus и массив, предназначенный для отправления и принятия данных согласно протоколу Modbus TCP.

## 3. Серверная часть ПО

### Описание работы серверной части

Серверная часть представляет собой приложение, которое получает доступ к переферийному устройству, после успешного подключения к которому настраивает сокет и начинает прослушивание клиентских запросов. В случае ошибки подключения (получения доступа к МК) программа завершается, выводя на экран сообщение для пользователя “Unable to open device”. После получения запроса, программа соответствующим образом его обрабатывает и отправляет ответ клиенту.

### Описание функций серверной части

В серверной части программного обеспечения используются следующие функции и структуры:

- Struct Color – структра, содержит поля типа short int red, green, blue для удобного выделения памяти при получении пакета;
- OnPing() – функция типа void, принимающий пакет от клиентской части. Используется для проверки работоспособности сервера;
- OnFill() – функция типа void, принимающий указатель на массив buf (команды для работы с МК), указатель на устройство handle и пакет данных pack. Получает из пакета данных цвет заполнения экрана, целиком заполняет экран соответствующим цветом и в случае успешного заполнения добавляет в пакет информации, возвращаемый клиенту сообщение “Screen is filled!”, в случае неудачной отправки информации на устройство добавит в пакет сообщение “Error!”;
- onColor() – функция типа void, принимающий указатель на массив buf (команды для работы с МК), указатель на устройство handle и пакет данных pack. Формирует команду изменения яркости светодиода,

получает данные об интенсивности каналов R G B из пакета и задает на устойстве соответсвующую яркость каждого светодиода. В случае успешной отправки информации на устройство о яркости светодиодов, метод добавит в пакет сообщение “Color is defined!”, в случае неудачи сообщение “Error!”;

– resistorState() – функция типа void, принимающий указатель на массив buf (команды для работы с МК), указатель на устройство handle и пакет данных pack. Отправляет запрос на получение данных с устройства, в случае успеха добавляет в пакет соответствующее значение напряжения резистора, в случае неудачи добавляет в пакет сообщение “Error!”.

## 4. Клиентская часть ПО

### Описание работы клиентской части

Клиентская часть представляет собой приложение, которое настраивает сокет и дает возможность отправлять запросы на сервер согласно протоколу Modbus TCP. Программа ожидает команд от пользователя до момента выхода пользователя из программы.

### Описание функций клиентской части

В клиентской части программного обеспечения используются следующие функции и структуры:

- Struct Color – структура, содержит поля типа short int red, green, blue для удобного выделения памяти при отправке пакета;
- getPackage() – функция типа struct package*, принимающая поле unit_id. Используется для формирования пакета в соответствии с протоколом Modbus TCP;
- ping() – функция типа void, принимающая сокет. Формируют пакет команды ping для проверки работы сервера и отправляет этот пакет. В случае получения ответа на экран выводится соответствующее сообщение, отправленное сервером. Иначе на экран выводится “timeout”;
- readColors() – функция типа struct color*. Создает экзепляр структуры color и записывает в соответствующие поля экземпляра интенсивность светодиодов red, green, blue;
- fillScreen() – функция типа void, принимаюащая сокет и значение цвета для заполнения экрана (0 или 1). Формирует пакет для функции заливки экрана, в случае успешной заливки на экране выведется соответсвующее сообщение от сервера, иначе сообщение “timeout”;
- sendColors() – функция типа void, принимающая сокет. Вызывает

функцию readColors() для ввода значений интенсивности светодиодов. Формирует пакет для функции изменения интенсивности светодиодов R, G, B. В случае успешного изменения на экран выведется соответствующее сообщение от сервера, иначе “timeout”;

- getResistorState() – функция типа void, принимающая сокет. Формирует пакет для отправки запроса на получение текущего состояния резистора. В случае успешной отправки на экран выведется соответствующее значение резистора, отправленное сервером;
- getSocket() – функция, возвращающая Socket. Используется для создания и настройки сокета.

## Результаты работы программы

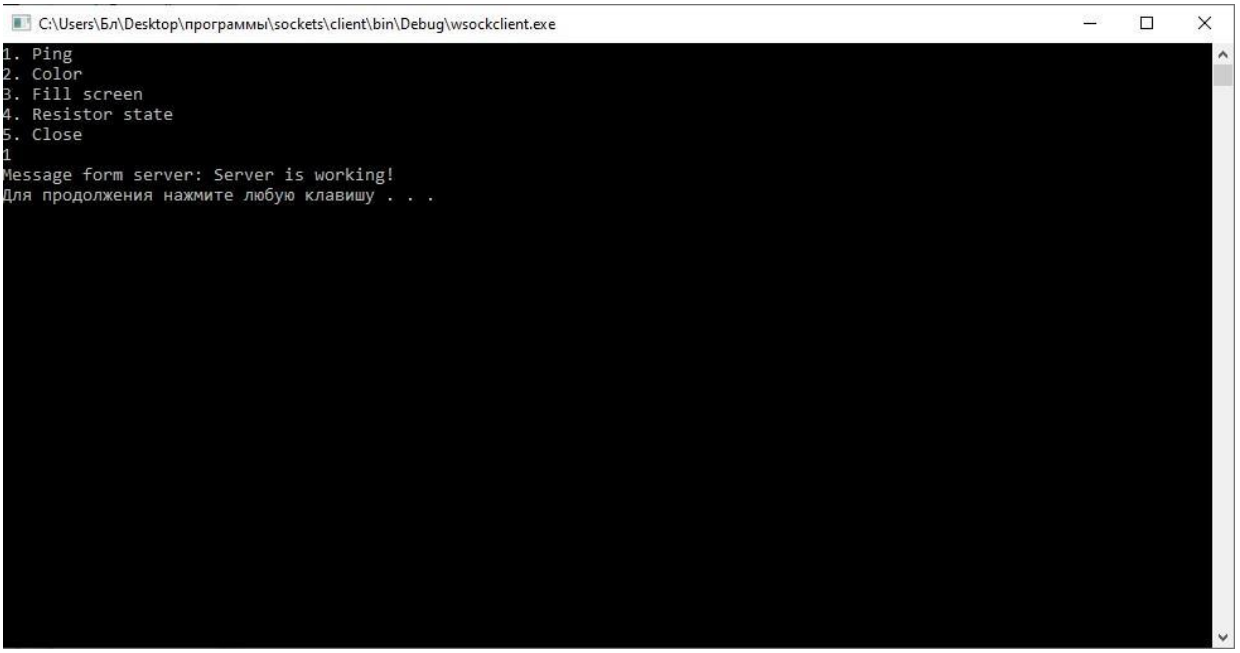


Рисунок 2 – Главное меню клиента и вызов функции Ping с последующим получением ответа от сервера

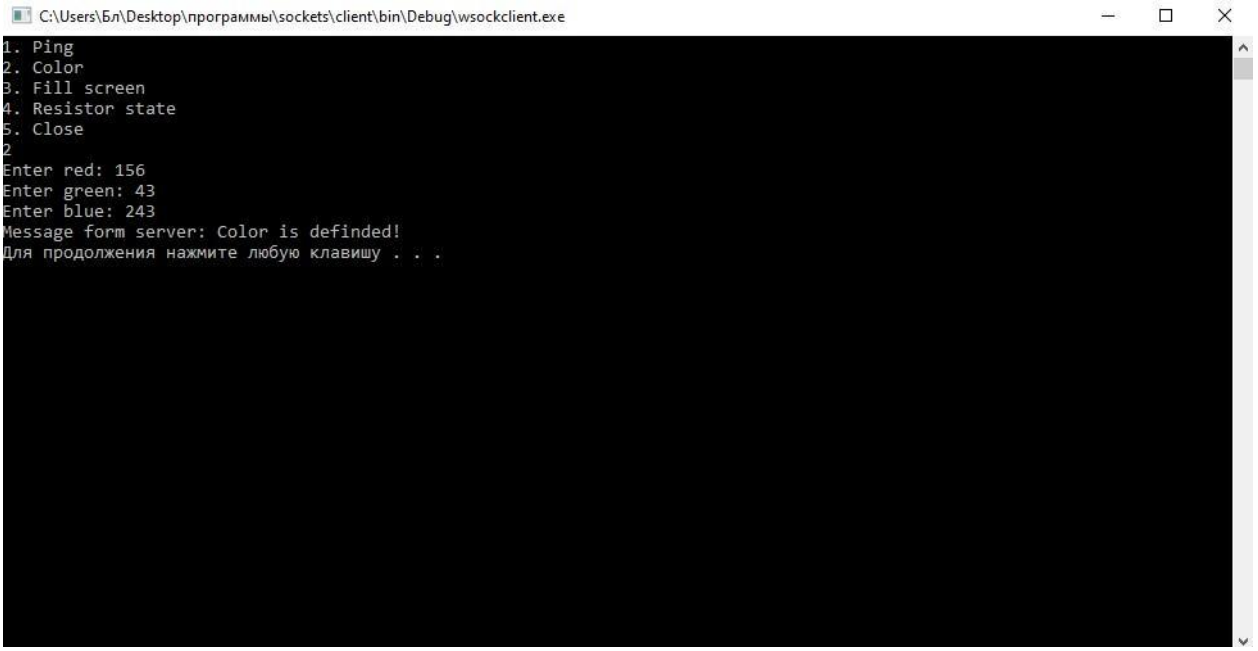


Рисунок 3 – Вызов функции Color с вводом значений интенсивности светодиодов и последующим получением ответа от сервера

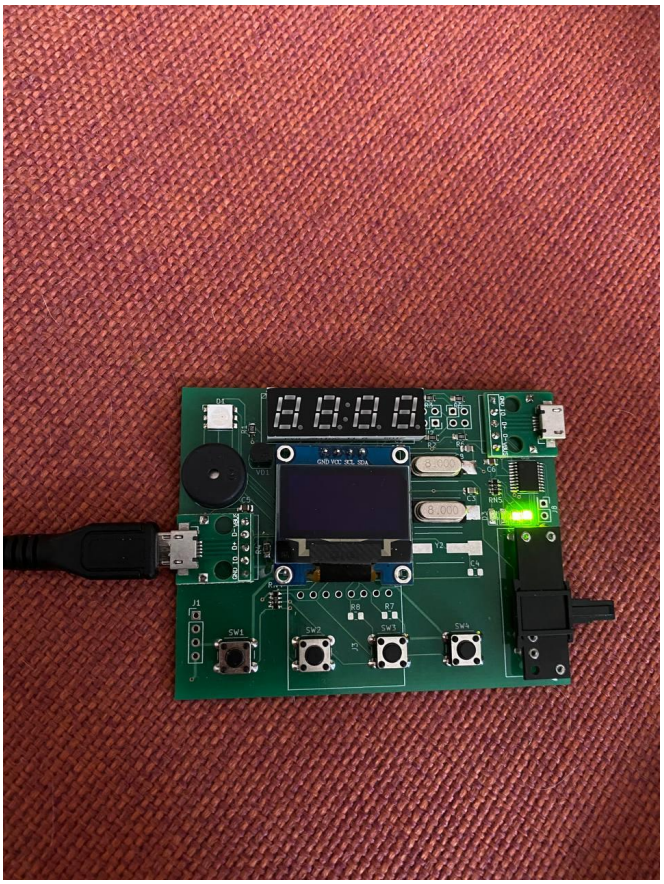

Рисунок 4 – Изначальное состояние устройства

Рисунок 5 – Состояние устройства после вызова пользователем функции Color с указанием интенсивностей светодиодов

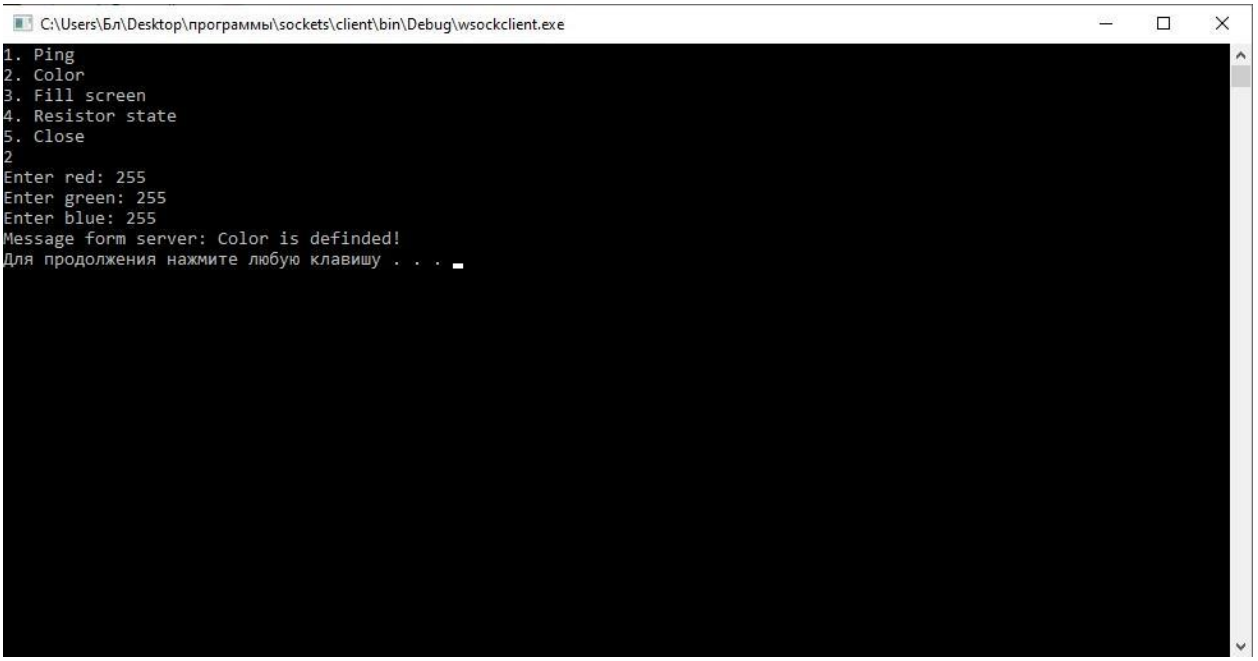


Рисунок 6 – Еще один вызов функции Color  с другими значениями интенсивности

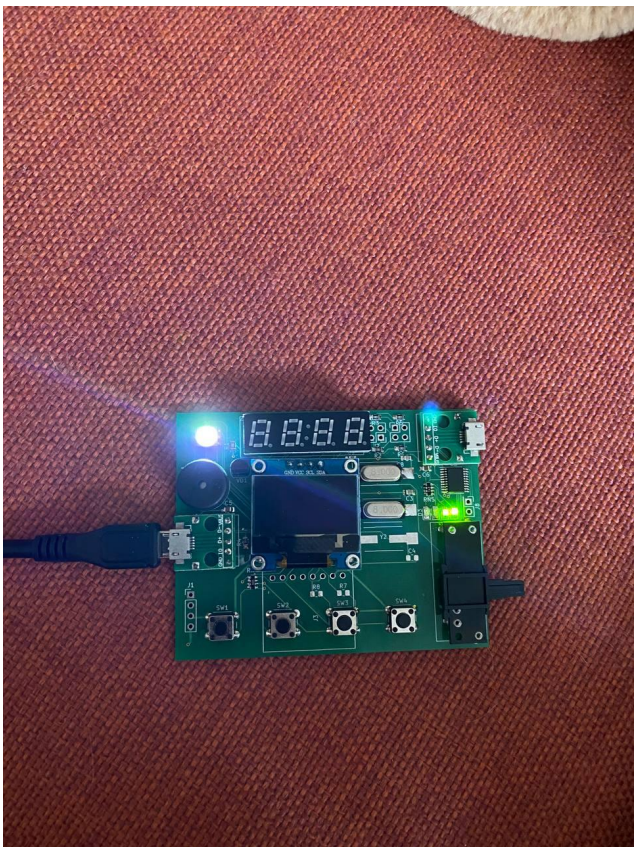

Рисунок 7 – Состояние устройства после второго вызова функции Color (светодиод теперь белый)


Рисунок 8 – Вызов функции закрашивания экрана

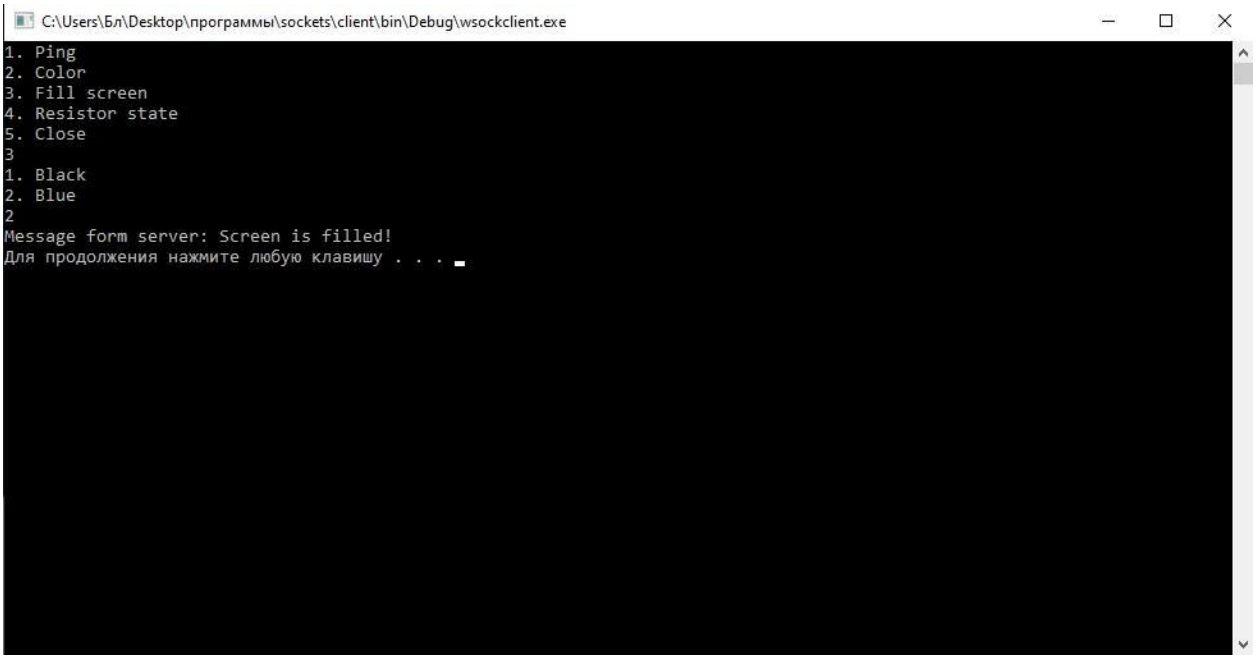


Рисунок 9 – Выбор цвета и получение сообщения от сервера об успешном закрашивании

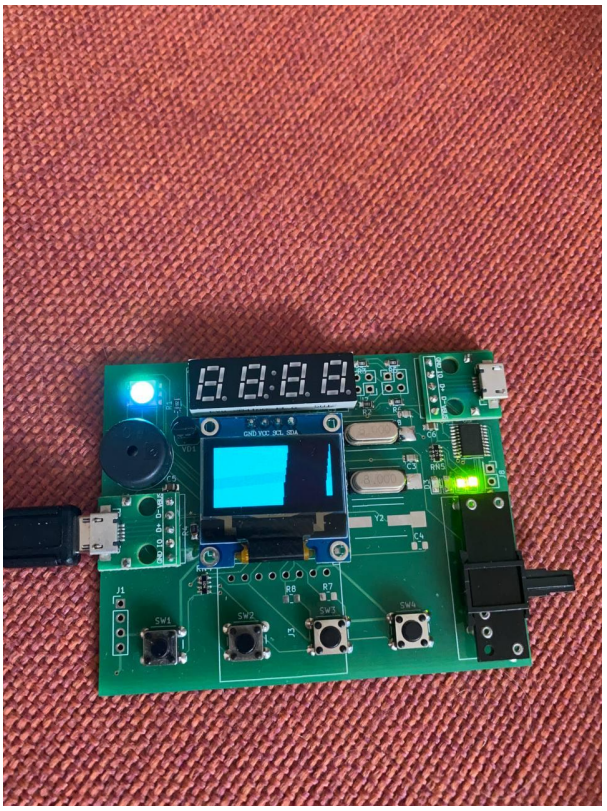

Рисунок 10 – Процесс закрашивания экрана

Рисунок 11 – Результат закрашивания экрана

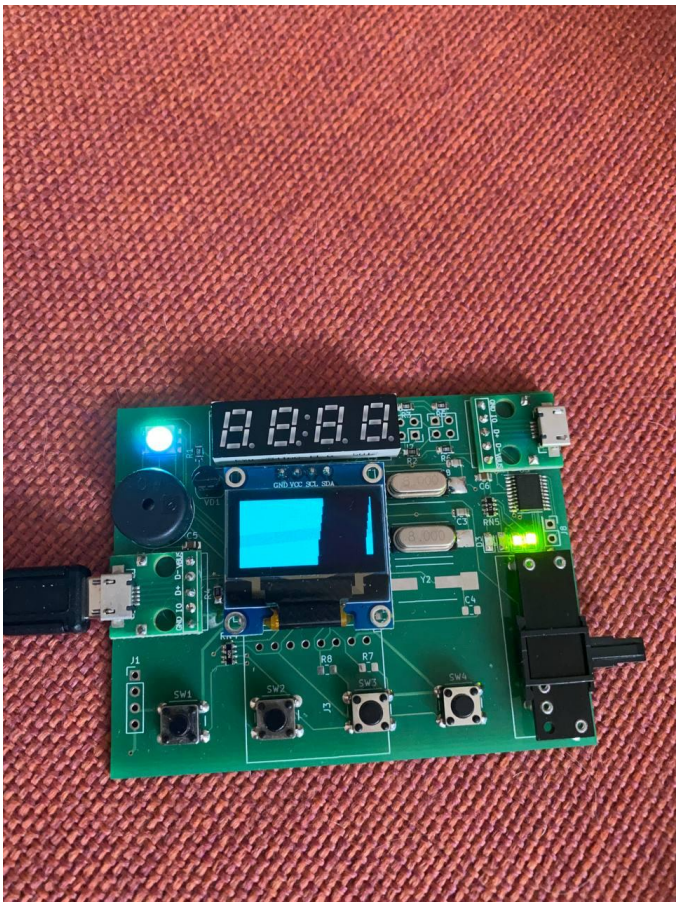

Рисунок 12 – Процесс повторного закрашивания экрана, теперь в черный цвет

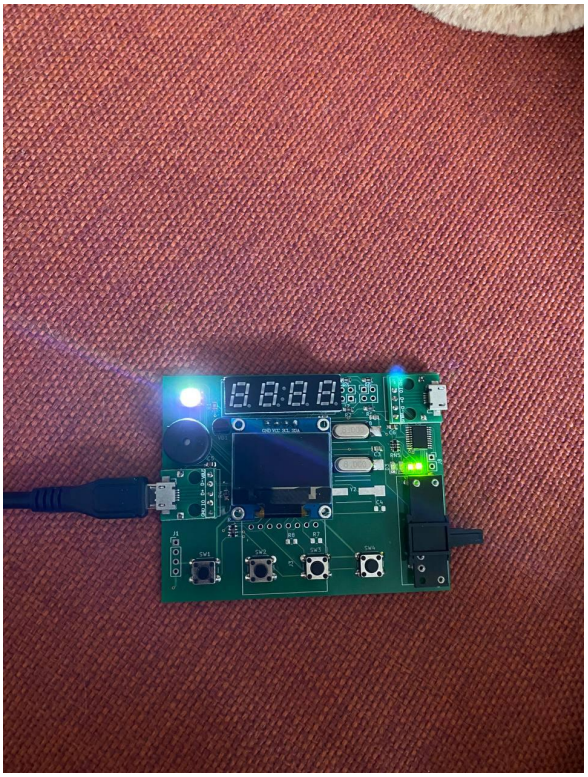

Рисунок 13 – Результат

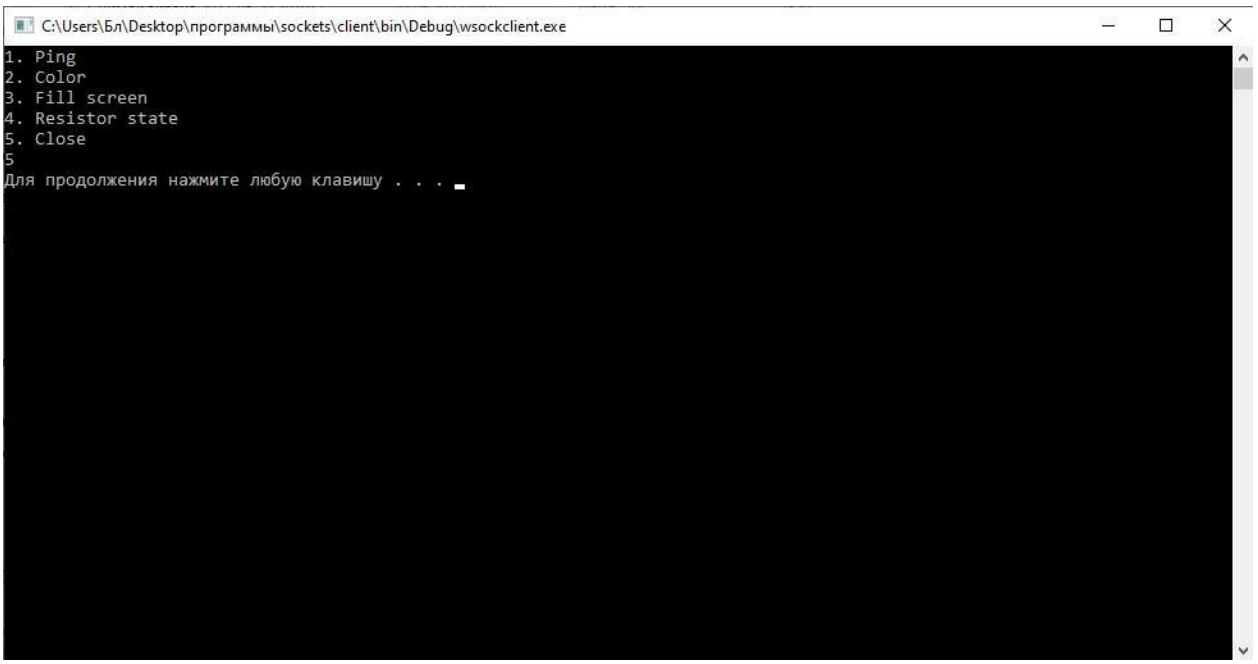


Рисунок 14 – Вызов функции для получения от сервера текущего значения напряжения резистора

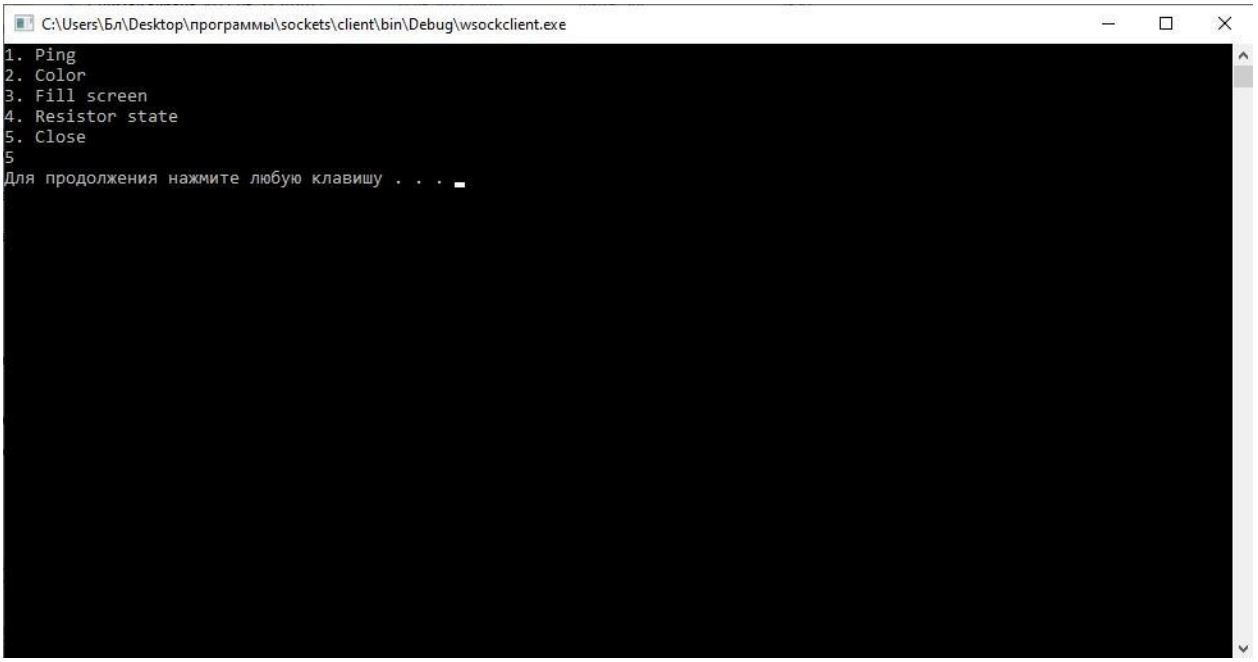


Рисунок 15 – Закрытие программы

# ЗАКЛЮЧЕНИЕ

В ходе курсовой работы было разработано клиент-серверное приложение для удаленного взаимодействия с периферийным устройством по локальной сети, использующее протокол Modbus TCP. В программе удалось реализовать все запланированные методы и решения. Были приобретены новые навыки работы с интернет протоколами и управления периферийными устройствами, усовершенствованы навыки владения структурами и функциями.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Таненбаум Э., Уэзеролл Д. Т18 Компьютерные сети. 5-е изд. — СПб.: Питер, 2012. — 960 с.: ил..
2. Хабр [Электронный ресурс] UML для самых маленьких. Диаграммы классов: https://habr.com/ru/post/511798/ (Дата обращения 11.05.2022)
3. ipc2u [Электронный ресурс] подробное описание протокола Modbus TCP с примерами команд: https://ipc2u.ru/articles/prostye-resheniya/modbus-tcp/ (Дата обращения 12.05.2022)
4. Metanit [Электронный ресурс] введение в сети и протоколы: https://metanit.com/sharp/net/1.1.php (Дата обращения 09.05.2022)
5. Хабр [Электронный ресурс] Сети для самых маленьких: https://habr.com/ru/post/134892/ (Дата обращения 10.05.2022)