МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра компьютерной инженерии и моделирования

СИСТЕМА УПРАВЛЕНИЯ ЭЛЕКТРИЧЕСКОЙ НАГРУЗКОЙ С ПОМОЩЬЮ СЕТИ ИНТЕРНЕТ

Курсовая работа
по дисциплине «Программирование»
студентки 1 курса группы ИВТ-б-о-192(1)
Фамилия Имя Отчество

направление подготовки 09.03.01 «Информатика и вычислительная техника»

Научный руководитель _____
(оценка)
старший преподаватель кафедры
компьютерной инженерии и моделирования _____ Чабанов В.В.
(подпись, дата)

Симферополь, 2020

РЕФЕРАТ

Разработка проекта «Система управления электрической нагрузкой с помощью сети интернет». 29 с., 33 ил., 4 ист.

Объект разработки – устройство управляющее электрической нагрузкой, макет, демонстрирующий работоспособность устройства, прошивку для микроконтроллера ESP8266, web-сайт и desktop приложение для управление устройством через Wi-Fi интерфейс.

Цель работы – Создание системы управления электрической нагрузкой с возможностью управления через сеть интернет. Проектирование 3д модели устройства в САПР “Компас-3Д”, печать спроектированных деталей на 3д принтере. Написание стабильной прошивки для микроконтроллера ESP8266 в свободной интегрированной среде разработки Arduino IDE. Создание веб-сайта с интерфейсом управления нагрузкой в виде двигателя постоянного тока, а также создание desktop-приложения в кроссплатформенном фреймворке Qt, позволяющем разрабатывать программное обеспечение на языке программирования C++.

ПРОГРАММИРОВАНИЕ, C++, Arduino IDE, ESP8266, IoT, ws2812b,

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Задачей курсовой работы является создание полноценного программного и аппаратного продукта, который может использоваться в коммерческих целях. Данный проект включает в себя разработку системы управления, состоящую из:

- создания и печати 3д модели;
- пайки;
- написания десктоп приложения;
- использования технологии WebSocket;
- Wi-Fi интерфейс;

В ходе выполнение курсовой работы, были получены не только навыки в программировании, но ещё и базовые инженерные навыки, такие как: пайка, прошивка микроконтроллера, разработка и создание 3д моделей.

ГЛАВА 1
АНАЛИЗ СРЕДСТВ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

1.1 Arduino IDE

Компилирование кода было произведено в среде разработки Arduino IDE. Интегрированная среда разработки Arduino - это кроссплатформенное приложение, написанное на языке программирования Java. Он используется для написания и загрузки программ на Arduino-совместимые платы, а также, с помощью ядер сторонних производителей, на платы разработки других производителей.

1.2 Microsoft Visual Studio

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

1.3 Qt Framework

Windows Forms — интерфейс программирования приложений(API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, C++, так и на VB.Net, J# и др.

1.4 Компас-3Д

Компас-3Д — мощная и универсальная система трёхмерного проектирования, ставшая стандартом для тысяч предприятий, благодаря простоте освоения и широким возможностям твердотельного, поверхностного и прямого моделирования.

Ключевой особенностью продукта является обеспечение сквозного процесса проектирования от реализации идеи в 3D до подготовки полного комплекта документации. В основе КОМПАС-3D лежат собственное математическое ядро и параметрические технологии, разработанные специалистами АСКОН. Продукт содержит инструменты для коллективного проектирования изделий и объектов строительного проектирования любой степени сложности и позволяет подготовить полноценную электронную модель изделия, здания и сооружения.

1.5 Cura

Слайсер Cura — это бесплатный проект компании Ultimaker. Программа совместима с огромным количеством 3D принтеров, слайсер может работать с файлами STL, 3MF и OBJ и в случае необходимости исправлять ошибки в 3d моделях. Слайсер отображает траекторию движения головки принтера, время печати и массу материала, которое будет затрачено при печати из выбранного материала. Нужно понимать одну очень важную особенность, слайсер CURA бесплатный и подходит для начинающих, и для продвинутых 3d печатников.

ГЛАВА 2
АППАРАТНАЯ ЧАСТЬ

2.1 Выбор микроконтроллера.

На рынке предоставлен большой выбор микроконтроллеров, однако выбор пал на ESP8266. По сравнению с другими микроконтроллерами, данный имеет следующие преимущества:

1. Наличие модуля Wi-Fi.
2. Встроенный стек протоколов TCP/IP.
3. Встроенный 10-битный Аналого-Цифровой преобразователь.
4. Шины: UART/HSPI/I2C.
5. Цена в 3$.

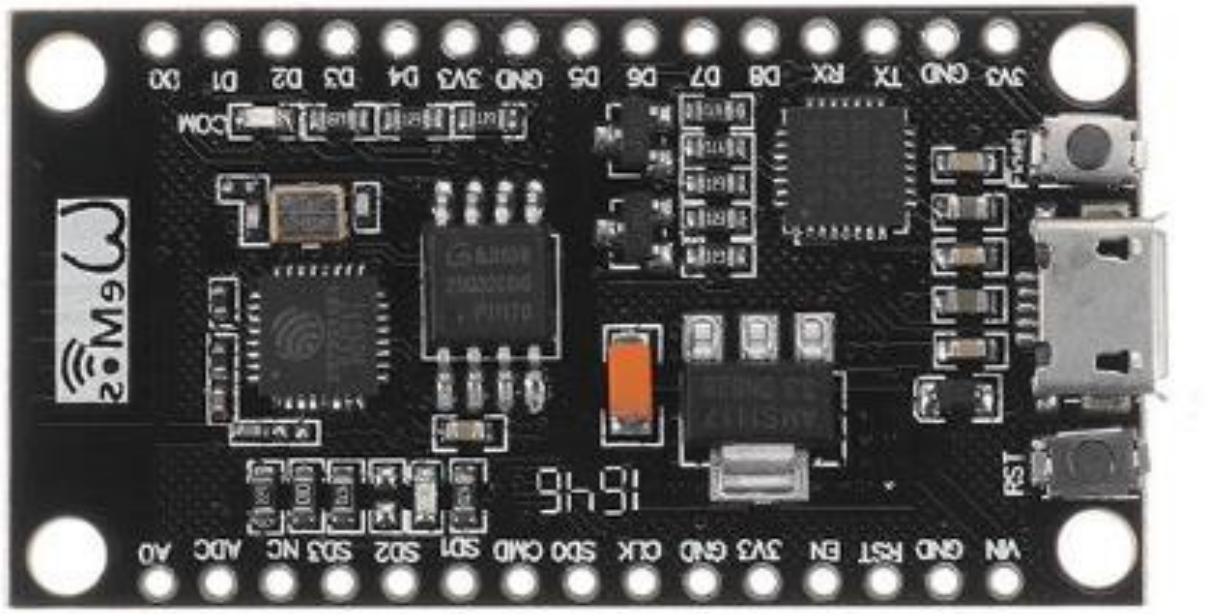

Рисунок 2.1.1 Микроконтроллер ESP8266

2.2 Выбор светодиодной ленты.

Была выбрана “умная” светодиодная адресная лента WS2812b. Светодиоды серии WS2812b можно отнести уже ко второму поколению полноцветных (RGB) светодиодов с индивидуальной (пиксельной) адресацией. В одном корпусе светодиода типоразмера 5050 собраны светодиоды RGB и

контроллер. Каждый кристалл светодиода может отображать 256 уровней яркости и более 16 млн. цветов. В отличие от предшествующей модели WS2811, имеющей 6 выводов, светодиод WS2812B имеет всего 4 вывода. Каждый из кристаллов (красный, синий, зеленый) при максимальной яркости потребляет ток порядка 20мА. Соответственно, максимальное энергопотребление светодиода составляет примерно 60мА для белого цвета.

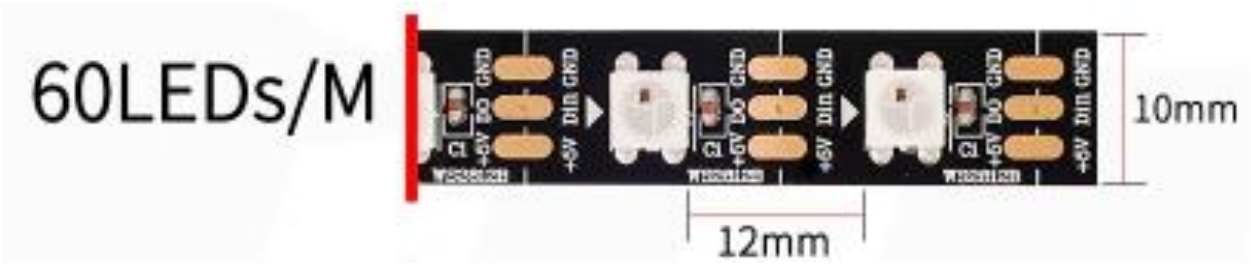

Рисунок 2.2.1 Светодиодная ленты WS2812B

Лента имеет 3 контактные площадки: GND – земля, 5V- питание, DiN – сигнальный контакт.

2.4 Выбор двигателя постоянного тока.

Для обеспечения симуляции работы радиального двигателя, был выбрал стандартный мотор-редуктор, часто используемый в различных проектах.

Рисунок 2.2.2 Мотор-редуктор

Характеристики движителя:

- Коэффициент передачи: 48:1
- Рабочее напряжение: 3-12 В (рекомендованное 6-8 В)
- Максимальный крутящий момент: 800 г*см (при 3 В)
- Скорость без дополнительной нагрузки: 170 об/мин (при 3 В)
- Ток нагрузки: 70 мА (максимум - 250 мА) (при 3 В)

2.3 Создание 3д модели устройства

Создание 3д модели производилось в отечественной САПР “КОМПАС-3Д”. КОМПАС-3D — система трехмерного проектирования, ставшая стандартом для тысяч предприятий, благодаря сочетанию простоты освоения и легкости работы с мощными функциональными возможностями твердотельного и поверхностного моделирования.

Ключевой особенностью продукта является использование собственного математического ядра C3D и параметрических технологий, разработанных специалистами АСКОН.

КОМПАС-3D обеспечивает поддержку наиболее распространенных форматов 3D-моделей (STEP, ACIS, IGES, DWG, DXF), что позволяет организовывать эффективный обмен данными со смежными организациями и заказчиками, использующими любые CAD / CAM / CAE-системы в работе.

Устройство состоит из 2 основных частей, корпуса и модели радиального двигателя.

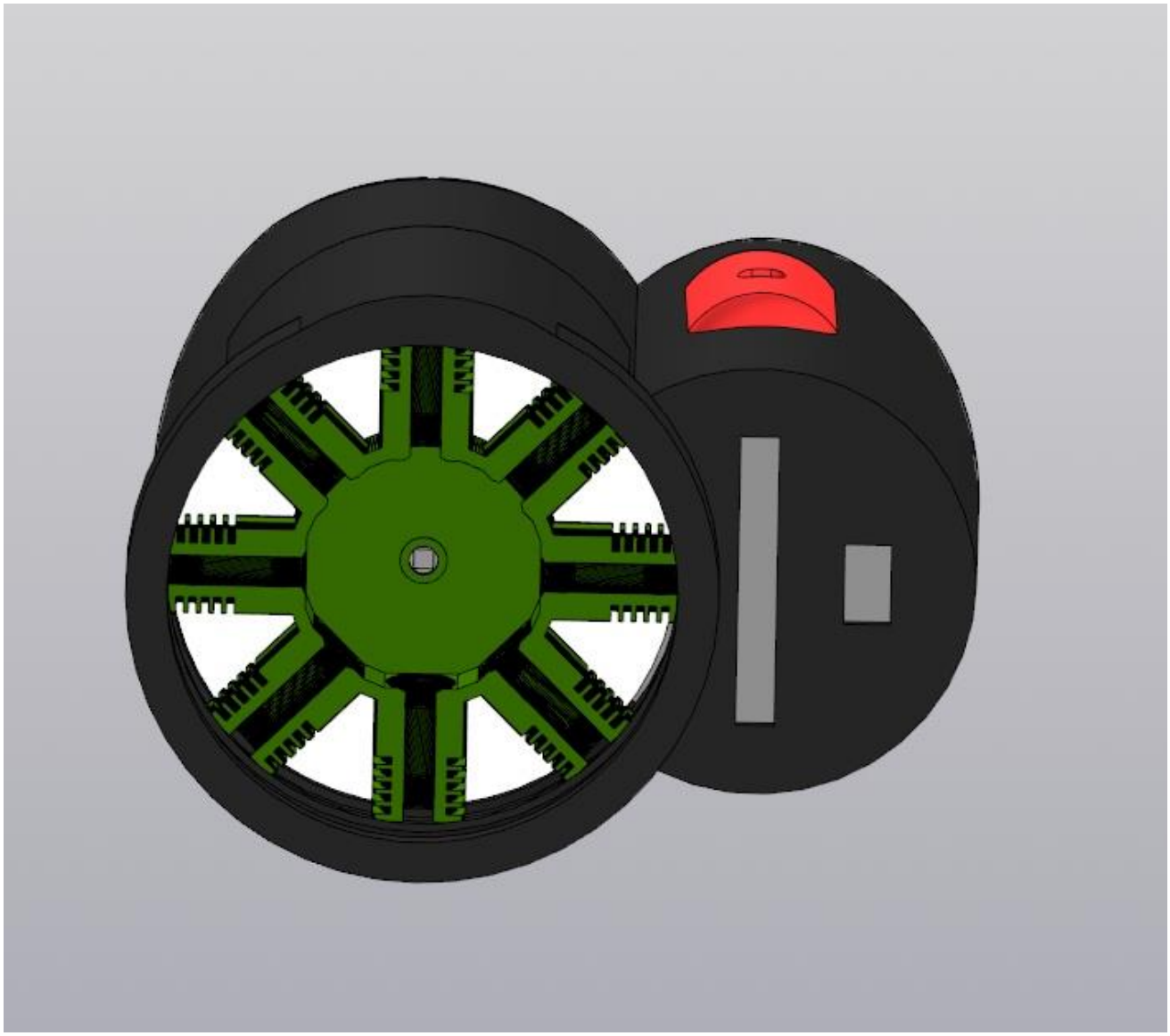

Рисунок 2.3.1 3Д модель устройства

2.4 Печать 3д модели

Для печати моделей, необходимо провести предварительную подготовку:

1. Перевод деталей в формат STL.
2. Импортировать файл формата STL в слайсер Cura. Слайсер Cura — это бесплатный проект компании Ultimaker. Программа совместима с огромным количеством 3D принтеров, слайсер может работать с файлами STL, 3MF и OBJ и в случае необходимости исправлять ошибки в 3d моделях. Слайсер отображает траекторию движения головки принтера,

время печати и массу материала, которое будет затрачено при печати из выбранного материала.

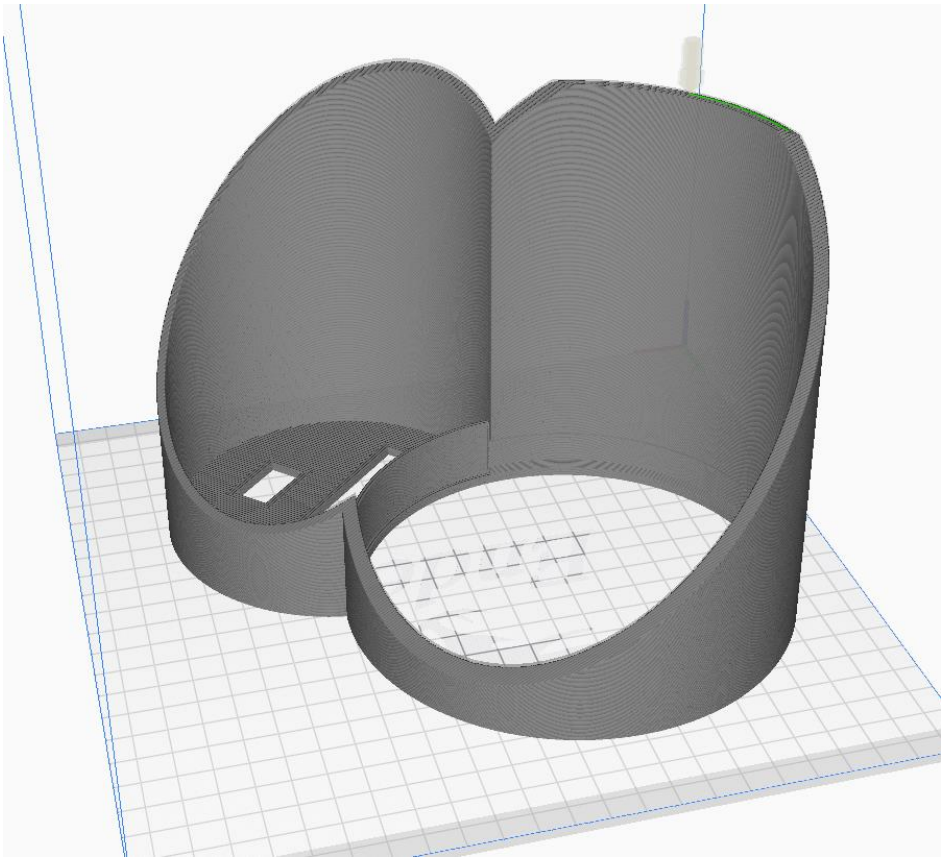

Рисунок 2.4.1 Модель подготовленная к печати

3. Сохранить “нарезанный” файл в формате Gcode на флеш-накопитель.

4. Вставить флеш-накопить в принтер и отправить в печать.

2.5 Сборка устройства.

После обработки деталей, напечатанных на 3д принтере, мотор-редуктор был соединён к модели радиального двигателя, после чего был установлен микроконтроллер в посадочное место. Двигатель подключён к микроконтроллеру через полевой транзистор, позволяющий управлять двигателем с помощью ESP8266. Также была установлена светодиодная лента и подключена к МК через токоограничивающий резистор.

Для управления светодиодной лентой или двигателем также возможно использовать переменный резистор. За подачу питания на двигатель отвечает переключатель, позволяющий физически отключить питание от мотора.

Рисунок 2.5.1 Собранное устройство

ГЛАВА 3
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРОШИВКИ И ПРОГРАММ

3.1 Постановка задачи программной реализации.

1. Написание стабильной прошивки сервера на микроконтроллере ESP8266.
2. Создание приложения desktop – приложения.
3. Создание веб-интерфейса.

3.2 Описание работы сервера.

При первом включении устройства, сервер открывает точку доступа по Wi-Fi интерфейсу. Пользователь подключается к точке доступа, далее пользователю предлагается выбрать его домашнюю точку доступа Wi-Fi, после чего устройство перезагружается и подключается к домашней сети, далее устройство будет автоматически подключаться к данной сети. На данном этапе сервер находится в одной сети с предполагаемыми клиентами. Любой клиент в данной сети может управлять сервером. Для того чтобы избежать ситуации, когда в сети находится клиент, который не должен управлять сервером, существует простая http – авторизация. Обмен данными между клиентом и сервером происходит по WebSocket протоколу.

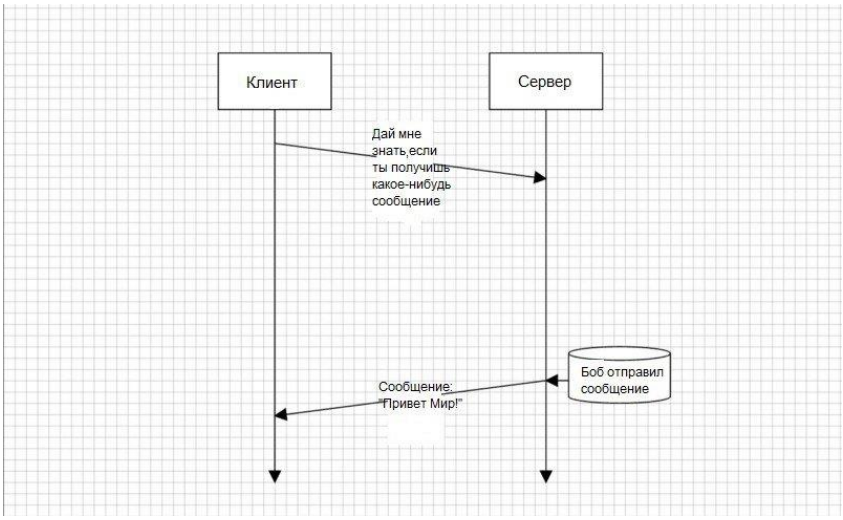

Рисунок 3.2.1 Иллюстрация работы протокола

Веб-сокеты (Web Sockets) — это передовая технология, которая позволяет создавать интерактивное соединение между клиентом (браузером) и сервером для обмена сообщениями в режиме реального времени. Веб-сокеты, в отличие от HTTP, позволяют работать с двунаправленным потоком данных, что делает эту технологию совершенно уникальной.

3.3 Написание стабильной прошивки сервера на микроконтроллере ESP8266.

3.3.1 Описание протокола общения с клиентами.

Для того, чтобы сделать сервер универсальным, то есть с возможностью управления различными клиентами на различных платформах, необходимо определить “протокол общения” понятный серверу.

```
if (payload[0] == '#') {
    // we get mode data
    speed = String((char *)&payload[1]).toInt();

    String json = "#";

    json += speed;
    // DBG_OUTPUT_PORT.println(json);
    webSocket.broadcastTXT(json);


}
if (payload[0] == '*') {
    // we get mode data
    step = String((char *)&payload[1]).toInt();

    String json = "*";

    json += step;
    // DBG_OUTPUT_PORT.println(json);
    webSocket.broadcastTXT(json);
```

Рисунок 3.3.1.1 Обработка данных, получаемых сервером.

В данном случае разработан следующий протокол обмена данных: Клиент передаёт серверу сообщение, в котором первый символ определяет какую переменную на сервере необходимо изменить, а последующее число – то, на сколько необходимо изменить. Зная данный протокол, любой программист может создать свой клиент под данный сервер.

3.3.2 Описание программного кода сервера.

```
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <WiFiClient.h>
#include <FS.h>
#include <WiFiManager.h>
#include <Ticker.h>
#include <WebSockets.h>
#include <WebSocketsServer.h>
#include "FastLED.h"
```

Рисунок 3.3.2.1 Подключение библиотек

Библиотеки <ESP8266WiFi.h> и <WiFiClient> необходимы для функционирования встроенного WiFi – интерфейса. <ESPAsyncTVP.h> позволяет открыть асинхронное соединение для передачи данных, в случае синхронного соединения, сервер ожидает итерации на который происходит обработка, а асинхронное соединение вызывает прерывание и мгновенную обработку данных.

<FS.h> обеспечивает сервер удобной файловой системой, позволяющей загружать и редактировать файл веб-интерфейса через браузер.

Система подключения сервера к необходимой сети реализована с помощью библиотеки <WifiManager.h>.

Протокол WebSocket реализован в библиотеках <WebSockets.h> и <WebSocketsServer.h>.

Для управления адресной светодиодной лентой используется библиотека “FastLED.h”.

После запуска микроконтроллера, единоразово вызывается функция void setup(), после чего программный код циклично выполняется в функции void loop().

```
//motor led
pinMode(D8, OUTPUT);
pinMode(D7, OUTPUT);
FastLED.addLeds<WS2811, PIN, GRB>(leds, NUM_LEDS).setCorrection(TypicalLEDStrip);
pinMode(2, OUTPUT);
FastLED.setBrightness(10);
// motor led
```

Рисунок 3.3.2.2 Назначение выходов микроконтроллера

В функции void setup(), происходит назначение выходов МК, а также настройка параметров адресной светодиодной ленты.

```
if (!wifiManager.autoConnect()) {
    ESP.reset();
    delay(10000);
}
```

Рисунок 3.3.2.3 Попытка подключения к точке доступа

Данное условие необходимо для перезагрузки устройства в случае, если сервер не смог подключиться к сохранённой точки доступа, после конфигурации.

```
const char* www_username = "admin";
const char* www_password = "esp8266";
```

Рисунок 3.3.2.4 Переменные хранящие данные логина и пароля.

В функции void loop () происходит зацикленное выполнение запросов.

```
void loop()
{
digitalWrite(D8, button);
FastLED.setBrightness(brightness);
rainbow();
}
```

Рисунок 3.3.2.5 Функция void loop().

В данном случае digitalWrite() отвечает за включение или выключения двигателя, setBrightness() меняет яркость светодиодной ленты. А также вызывается функция rainbow().

```
void rainbow()
{
    for (int i = 0; i < NUM_LEDS; i++) {

        leds[i] = CHSV(counter + i * step, 255, 255);
    }
    counter++;
    FastLED.show();
    checkForRequests();
    delay(speed);
}
```

Рисунок 3.3.2.6 Функция rainbow().

Данная функция создает так называемый эффект радуги на светодиодной ленты с изменяемыми параметрами speed и step. Изменять данные параметры будут клиенты, тем самым изменяя свечение светодиодной ленты. Функция checkForRequests() вызывает опрос протокола WebSocket и http – сервера.

```
void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t lenght) {
    switch (type) {
    case WStype_DISCONNECTED:
        // DBG_OUTPUT_PORT.printf("WS: [%u] Disconnected!\n", num);
        break;

    case WStype_CONNECTED: {
```

Рисунок 3.3.2.7 Функция WebSocketEvent

В случае если поступили данные по протоколу WebSocket, вызывается функция обработчик webSocketEvent(), которая принимает и обрабатывает данные.

```
json += brightness;
webSocket.sendTXT(num, json);
json = "#";
json += speed;
webSocket.sendTXT(num, json);
json = "*";
json += step;
webSocket.sendTXT(num, json);

if (button) {

    json = "?1";

}
else json = "?0";
webSocket.sendTXT(num, json);
```

Рисунок 3.3.2.8 Отправка клиентам текущее состояние сервера.

Сначала функция webSocketEvent() проверяет, произошло ли соединение, после чего отправляет клиенту текущее состояние переменных сервера, для того, чтобы синхронизировать клиент и сервер.

```
if (payload[0] == '#') {
    speed = String((char *)&payload[1]).toInt();
    String json = "#";
    json += speed;
    webSocket.broadcastTXT(json);
}
```

Рисунок 3.3.2.9 Обработка принятых данных.

После получения данных от клиента данных в формате заранее установленного “протокола общения”, сервер обрабатывает данные и в зависимости от того какие пришли данные, меняет значения переменных. Также сервер моментально отправляет на все клиенты запрос, сообщающий о том, что произошло изменение переменной.

3.4 Создание desktop – приложения.

Desktop – приложение было создано в Qt Framework, написано на языке C++. Приложение является клиентом и отправляет данные на сервер.

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ws = new QWebSocket();
    ui->setupUi(this);
    connect(ws, &QWebSocket::connected, this, &MainWindow::onConnected);
    connect(ws, &QWebSocket::disconnected, this, &MainWindow::onDisconnected);
    connect(ws, &QWebSocket::textMessageReceived, this, &MainWindow::onReceive);
    ws->open(QUrl("ws://192.168.0.228:81"));
}
```

Рисунок 3.4.1 Конфигурация WebSocket

При запуске происходит соединение событий с обработчиками событий, а также подключение WebSocket клиента к серверу по статическому ip – адресу с портом 81.

```
void MainWindow::onDisconnected()
{
    ui->step->setEnabled(false);
    ui->speed->setEnabled(false);
    ui->brightness->setEnabled(false);
    ui->endis->setEnabled(false);
    ui->progress->setVisible(true);
    ui->connection->setVisible(true);
}

void MainWindow::onConnected()
{
    ui->step->setEnabled(true);
    ui->speed->setEnabled(true);
    ui->brightness->setEnabled(true);
    ui->endis->setEnabled(true);
    ui->progress->setVisible(false);
    ui->connection->setVisible(false);
}
```

Рисунок 3.4.2 Блокировка и разблокировка интерфейса

До того, как произошло соединение, элементы интерфейса, отвечающие за изменение параметров сервера, не активны. После того как произошло успешное подключение, скрываются элементы, демонстрирующие, что соединение все ещё не установлено, а элементы управления становятся активными.

```
void MainWindow::onReceive(const QString &mes)
{
    QString message = mes;
    if(message == "?0")
    {
        ui->endis->setText("Включить");
    }
    else if(message == "?1")
    {
        ui->endis->setText("Выключить");
    }
    else if(message.contains('*'))
    {
        message = message.replace('*', "");
        ui->step->setValue(message.toInt());
    }
    else if(message.contains('#'))
    {
        message = message.replace('#', "");
        ui->speed->setValue(message.toInt());
    }
    else if(message.contains('@'))
    {
        message = message.replace('@', "");
        ui->brightness->setValue(message.toInt());
    }
}
```

Рисунок 3.4.3 Обработчик ответа сервера

При отправке данных с сервера, срабатывает обработчик, который “парсит” информацию, полученную с сервера, а также изменяет данные слайдеров и кнопки в соответствии полученными данными.

```
void MainWindow::on_brightness_valueChanged(int value)
{
    ws->sendTextMessage("@" + QString::number(ui->brightness->value()));
}

void MainWindow::on_step_valueChanged(int value)
{
    ws->sendTextMessage("*" + QString::number(ui->step->value()));
}

void MainWindow::on_speed_valueChanged(int value)
{
    ws->sendTextMessage("#" + QString::number(ui->speed->value()));
}

void MainWindow::on_endis_clicked()
{
    ws->sendTextMessage("?");
}
```

Рисунок 3.4.4 Отправка данных на сервер

Когда пользователь изменяет значение слайдера или нажимает на кнопку “Включить”, срабатывает обработчик, который отправляет на сервер запрос, согласно “протоколу общения”.

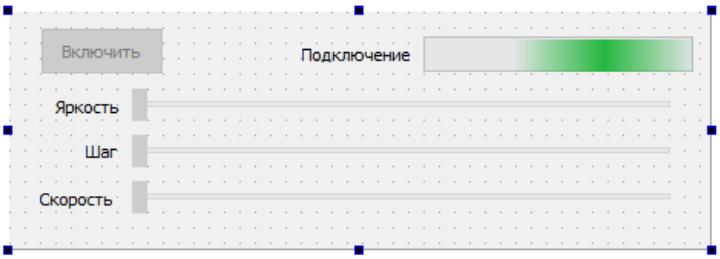

Рисунок 3.4.5 Графический интерфейс в Qt

3.5 Создание веб-интерфейса.

Был создан веб-интерфейс, написанный на языке разметки html вставками скриптов написанных на JavaScript. Интерфейс позволяет управлять сервером, с любого устройства поддерживающего браузер. После того как клиент вводит в адресную строку браузера адрес сервера, браузер запрашивает файл страницы с сервера, а тот в свою очередь, высылает файл веб-интерфейса.

```
function init() {
    console.log("Connection websockets to:", ws_url);
    connection = new WebSocket(ws_url, ['arduino']);

    // When the connection is open, send some data to the server
    connection.onopen = function () {
        //connection.send('Ping'); // Send the message 'Ping' to the server
        console.log('WebSocket Open');
    };
```

Рисунок 3.5.1 Конфигурация WebSocket протокола

В файле веб-интерфейса происходит конфигурация и соединение с сервером по WebScoket протоколу.

```
$("#pane2").on("change", ".update_brightness", function() {
    var brightness = $("#rng_brightness").val();

    wsSendCommand("@" + brightness);
});

    $("#pane2").on("change", ".update_speed", function() {
    var brightness = $("#rng_speed").val();

    wsSendCommand("#" + brightness);
});
    $("#pane2").on("change", ".update_step", function() {
    var brightness = $("#rng_step").val();

    wsSendCommand("*" + brightness);
});

    $("#pane2").on("change", ".update_switch", function() {

      wsSendCommand("?");


});
```

Рисунок 3.5.2 Отправка данных по WebSocket протоколу.

При измении слайдера или кнопки, вызывается функция, отправляющая данные по WebSocket протоколу на сервер.

```
console.log('WebSocket from server: ' + e.data);
if(e.data == '?1') {
  document.getElementById('switch1').checked = true;
} else if (e.data == '?0') {
  document.getElementById('switch1').checked = false;
}
else if(e.data[0] == '@'){

  document.getElementById('rng_brightness').value = e.data.substring(1);
}
    else if(e.data[0] == '#'){

  document.getElementById('rng_speed').value = e.data.substring(1);
}

    else if(e.data[0] == '*'){

  document.getElementById('rng_step').value = e.data.substring(1);
}
```

Рисунок 3.5.3 Реализация обратной связи с сервером.

При получении данных от сервера, обновляется состояние интерфейса, так называемая форма обратной связи.

Рисунок 3.5.4 Веб-интерфейс.

ГЛАВА 4
ТЕСТИРОВАНИЕ

4.1 Проверка работоспособности.

Так как при пайке и сборке, происходит нагрев компонентов, существует вероятность выхода из строя одного из компонентов. Для проверки работоспособности компонентов, в ESP8266 была загружена тестовая прошивка и проверена работоспособность устройства.

Список клиентов DHCP

ID	Имя клиента	MAC-адрес	Назначенный IP-адрес	Срок действия адреса
1	ESP-2E88E2	CC-50-E3-2E-88-E2	192.168.0.106	Постоянный
2	android-896f2fcd6656c12a	40-D3-AE-36-DA-0E	192.168.0.105	01:58:24
3	GM8GO-30e9c3c83bd874ca	7C-B9-60-1A-87-6A	192.168.0.100	01:58:26
4	DESKTOP-RE4BT5U	3C-91-80-D4-83-BF	192.168.0.102	01:58:30
5	Redmi4X-Redmi	50-8F-4C-DC-79-A3	192.168.0.101	01:58:58

Рисунок 4.1.1 Привязка MAC- адреса

При запуске, устройство создало точку доступа Wi-Fi, к которой был подключен смартфон и выбрана домашняя Wi-Fi сеть, после чего устройство перезагрузилось и подключилось к домашней сети.

Необходимо зайти в настройки роутера и привязать MAC- адрес ESP8266 к ip – адресу 192.168.0.106

Устройство готово к работе.

4.2 Проверка работоспособности Desktop-приложения.

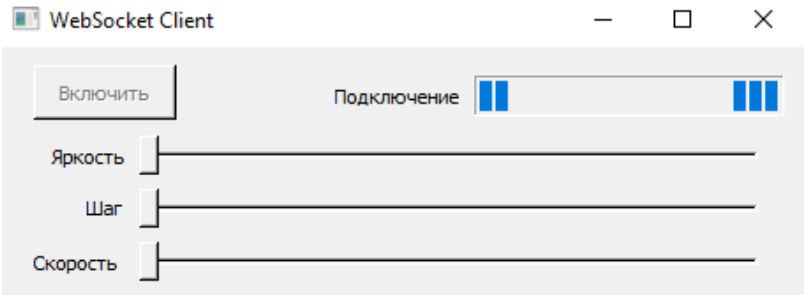

Рисунок 4.2.1 Программа устанавливает соединение.

Запускаем программу и ждём, когда клиент установит соединение с сервером.

Рисунок 4.2.2 Программа готова к работе.

После того, как соединение было установлено, активные элементы были разблокированы и готовы к работе.

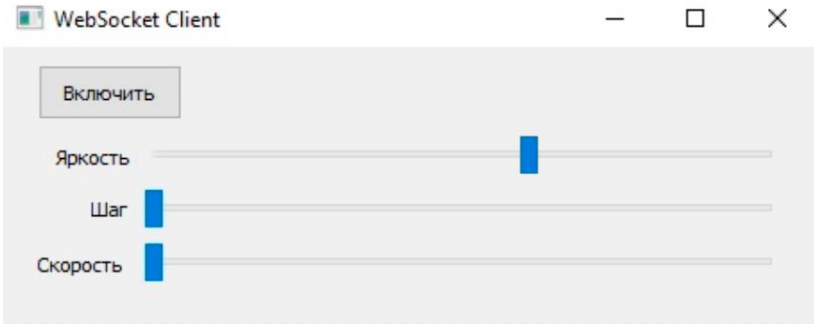

Рисунок 4.2.3 Программа работает.

При изменении ползунка яркости на сервер была отправлена информация соответствующая информация. Сервер изменил яркость на соответствующее слайдеру значение.

4.3 Проверка работоспособности веб-интерфейса.

Для проверки работоспособности веб-интерфейса был запущен браузер Google Chrome, а также само устройство. В адресную строку был введён адрес сервера, а именно: 192.168.0.106.

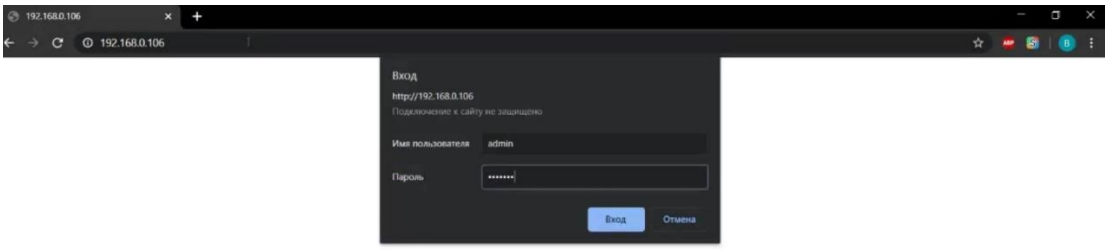

Рисунок 4.3.1 Окно авторизации.

После ввода адреса сервера, выводится окно авторизации, вводим логин и пароль.

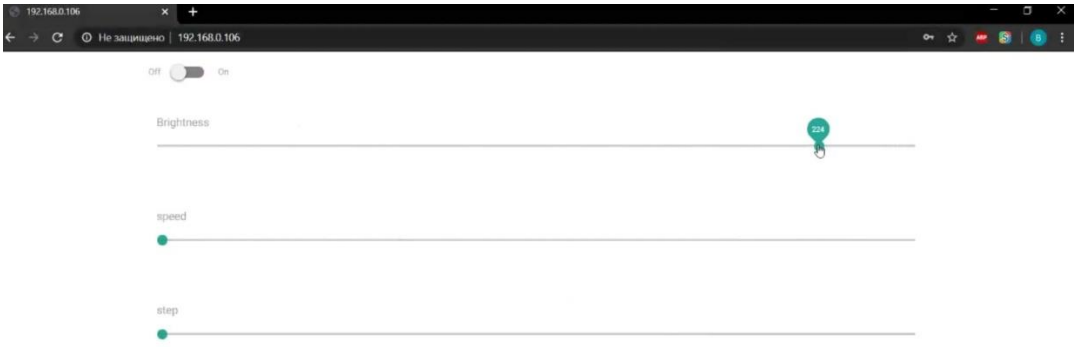

Рисунок 4.3.2 Веб-интерфейс.

Клиент успешно отправляет данные, а сервер принимает.

4.4 Выводы после тестирования.

В ходе тестирования были проверена работоспособность как аппаратной части, так и программного кода. Была проверена работа desktop – приложения, а

также веб-интерфейса. Авторизация работает, сервер реагирует и отправляет своё состояние клиентам.

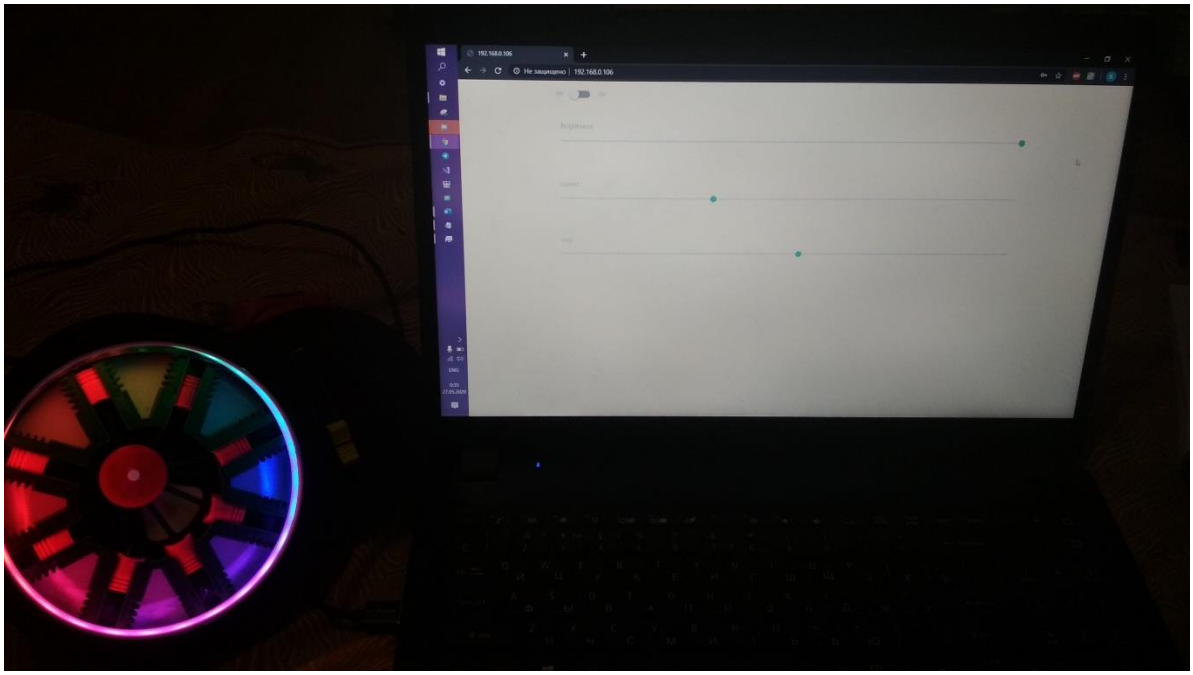

Рисунок 4.4.1 Устройство работает.

ГЛАВА 5
ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА

5.1 Перспективы технического развития

Изначально проект подразумевал демонстрацию возможностей микроконтроллера ESP8266 и клиент-серверной системы, созданной в данной курсовой работе. В перспективе данная связка имеет массу применений например: IoT.

Интернет вещей (англ. internet of things, IoT) — концепция вычислительной сети физических предметов “вещей”, оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключающее из части действий и операций необходимость участия человека.

В данным момент IoT – девайсы, буквально, захватывают мир. Ещё в 2009 году устройств в сети стало больше, чем населения Земли, причём с каждым годом это значение увеличивается в геометрической прогрессий. “Умные” вещи используются повсюду: холодильники, микроволновки, гаражи, часы, очки, духовка и прочие умные девайсы. Также набирает популярность система “Умный дом”, позволяющая дистанционно управлять всеми умными вещами в доме.

5.2 Перспективы коммерциализации

В дальнейшем планируется использовать систему, созданную в рамках данной курсовой работы, для создания: умной лампы, умных ворот, умного освещения и прочих систем, которые возможно автоматизировать и сделать “умными”. Глобальный рынок технологий Интернета вещей (IoT), который состоит из программного обеспечения, сервисов, услуг подключения и устройств, в 2018 году достиг 201 млрд долларов. Согласно прогнозам аналитической компании Global Data, к 2023 году его объем составит 318 млрд долларов, при совокупном годовом приросте (CAGR) в 20%.

На рынке лидируют решения для коммунальных и производственных предприятий. Аналитики оценивают их возможности в 58% в этом году и 55% рынка в 2023 году. Энергетике и транспорту эксперты отводят 15% рынка в течение прогнозируемого периода. Наибольшим потенциалом прибыли на рынке обладают программное обеспечение и сервисы, такие как проектирование, установка, обслуживание и управление проектами, а также IoT-платформы, услуги по разработке приложений и ПО.

С точки зрения внедрения IoT на производстве, предприятия только начинают использовать технолонию для более эффективного контроля над расходами или повышения производительности. Развертывания IoT становятся обширней, технология набирает привлекательность для компаний. Технологию меняют дополненная и виртуальная реальность, машинное обучение и искусственный интеллект, что в итоге позволит предприятиям не только оптимизировать с помощью IoT текущие продукты и процессы, но и генерировать новые потоки доходов посредством разработки собственных новых продуктов и услуг.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы был создан полноценный программный и аппаратный продукт, который, в перспективе может использоваться в коммерческих целях. Данный проект включает в себя разработку и системы управления, включающую в себя: создание и печать 3д модели, сборку, пайку, написание приложений на Windows, создание веб-интерфейса, использование технологии WebSocket и Wi-Fi интерфейс. Были получены базовые инженерные навыки такие как: 3д моделирование, 3д печать, схемотехника, работа с клиент-серверной системой, программирование микроконтроллеров, создание IoT девайса, создание Web-интерфейса с JavaScript вставками, программирование на языке C++ в Qt Framework.

ЛИТЕРАТУРА

1. “Internet of Things with ESP8266” Марко Шварц 2016 г
2. “ESP8266 Home Automation Projects: Leverage the Power of this Tiny WiFi Chip” Каталин Батрини 2017 г
3. “ESP8266 NodeMCU Using Arduino IDE (Internet of Things): Getting Start with ESP8266 (iot Hands on Projects)” Джакобе Кале 2018 г
4. Википедия – свободная энциклопедия. Arduino IDE [Электронный ресурс https://ru.wikipedia.org/wiki/Arduino]