Утверждена распоряжением декана ФЭУ № 1 от 21.01.2020 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СКБ ФЭУ «Промышленная робототехника»

СОГЛАСОВАНО

Декан ФЭУ

_____ А.С. Гудим
(подпись)

«19» 03 2020 г.

Заведующий кафедрой 17

_____ Н.Н. Любушкина
(подпись)

«19» 03 2020 г.

УТВЕРЖДАЮ

Начальник отдела ОПРО

_____ Ю.С. Иванов
(подпись)

«19» 03 2020 г.

Настольная лампа переменного тока с управлением по Wi-Fi. Комплект документации на управляющую программу для микроконтроллеров ESP8266 и ATTINY85.

Руководитель проекта _____ С.И. Сухоруков
(подпись, дата)

Ответственный исполнитель _____ А.Р. Овсянников
(подпись, дата)

Комсомольск-на-Амуре 2020

Карточка проекта

Название	Настольная лампа переменного тока с управлением по Wi-Fi.
Тип проекта	Инициативный
Исполнители	А.Р. Овсянников –8МРб-1
Срок реализации	Март 2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ

на разработку

Выдано студентам: А.Р. Овсянников –8МРб-1

Название проекта: Настольная лампа переменного тока с управлением по Wi-Fi.

Назначение: Изучение программирования микроконтроллеров ATTINY85 и ESP8266

Область использования: Промышленная автоматизация

Функциональное описание управляющей программы:

После запуска кода, устройство инициализируется в указанной пользователем сети Wi-Fi, в случаи ее отсутствия создает собственную точку Wi-Fi, после предоставляет доступ пользователю к web-интерфейсу для управления яркостью лампы переменного напряжения.

Оборудование, для которого разрабатывается программа:

ATTINY85, ESP8266

Требования: Управляющая программа должна обеспечивать плавное регулирование яркостью лампы переменного тока через web-интерфейс или при помощи физического энкодера на корпусе устройства.

План работ:

Наименование работ	Срок
Разработка принципиальной схемы устройства.	Март 2020
Разработка печатной платы устройства.	Март 2020
Разработка программной части устройства.	Март 2020
Разработка UI.	Март 2020
Разработка управляющей программы для обработки корпуса лампы по плоскому контуру и объемной модели.	Март 2020

Комментарии:

_____

_____

_____

Перечень графического материала:

1. Принципиальная схема
2. Чертёж печатной платы

Руководитель проекта _____ С.И. Сухоруков

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ПАСПОРТ

Управляющей программы для микроконтроллеров ESP8266 и ATTINY85.

Настольная лампа переменного тока с управлением по Wi-Fi.

Руководитель проекта _____ С.И. Сухоруков

(подпись, дата)

Ответственный исполнитель _____ А.Р. Овсянников

(подпись, дата)

Комсомольск-на-Амуре 2020

Содержание

1 Общие положения 7
1.1 Наименование программы 7
1.2 Наименования документов, на основании которых ведется проектирование системы 7
1.3 Перечень организаций, участвующих в разработке системы 7
1.4 Сведения об использованных при проектировании 8
нормативно-технических документах 8
2 Описание программы 9
2.1 Общие сведения 9
2.2 Функциональное назначение программы 9
2.3 Описание логической структуры 16
2.4 Используемые технические средства 16
2.5 Вызов и загрузка 16
3 Руководство оператора 17
3.1 Назначение программы; 17
3.2 Условия выполнения программы; 17
3.3 Выполнение программы; 17
ПРИЛОЖЕНИЕ А 18
ПРИЛОЖЕНИЕ Б 24
ПРИЛОЖЕНИЕ В 25

1 Общие положения

Настоящий паспорт является документом, предназначенным для ознакомления с основной структурой, особенностями и правилами эксплуатации управляющей программы «Настольная лампа переменного тока с управлением по Wi-Fi.» (далее «программа»).

Паспорт входит в комплект поставки программы. Перед запуском программы внимательно изучите правила ее эксплуатации.

1.1 Наименование программы

Полное наименование программы – «Настольная лампа переменного тока с управлением по Wi-Fi.».

1.2 Наименования документов, на основании которых ведется проектирование системы

Создание программы «Настольная лампа переменного тока с управлением по Wi-Fi.» осуществляется на основании требований и положений следующих документов:

- задание на разработку.

1.3 Перечень организаций, участвующих в разработке системы

Заказчиком создания программы «Настольная лампа переменного тока с управлением по Wi-Fi.» является Федеральное государственное бюджетное образовательное учреждение высшего образования «Комсомольский-на-Амуре государственный университет» (далее заказчик), находящийся по адресу: 681013, Хабаровский край, г. Комсомольск-на-Амуре, Ленина пр-кт., д. 27.

Исполнителем работ по созданию программы «Настольная лампа переменного тока с управлением по Wi-Fi.» является Конструктор студенческого конструкторского бюро электротехнического факультета (далее СКБ ФЭУ), студент группы 8МРб-1, А.Р. Овсянников.

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

ГОСТ 19.001-77. Единая система программной документации (ЕСПД). Общие положения.

ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения

ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.

ГОСТ 19.102-77. ЕСПД. Стадии разработки.

ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.

ГОСТ 19.402-78. ЕСПД. Описание программы.

ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.

ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению.

2 Описание программы

2.1 Общие сведения

Управляющая программа обеспечивает развертку веб сервера в локальной сети Wi-Fi с web страницей управления устройством, также обработкой физического энкодера.

2.2 Функциональное назначение программы

Программа применяется для изменения яркости АС лампы.

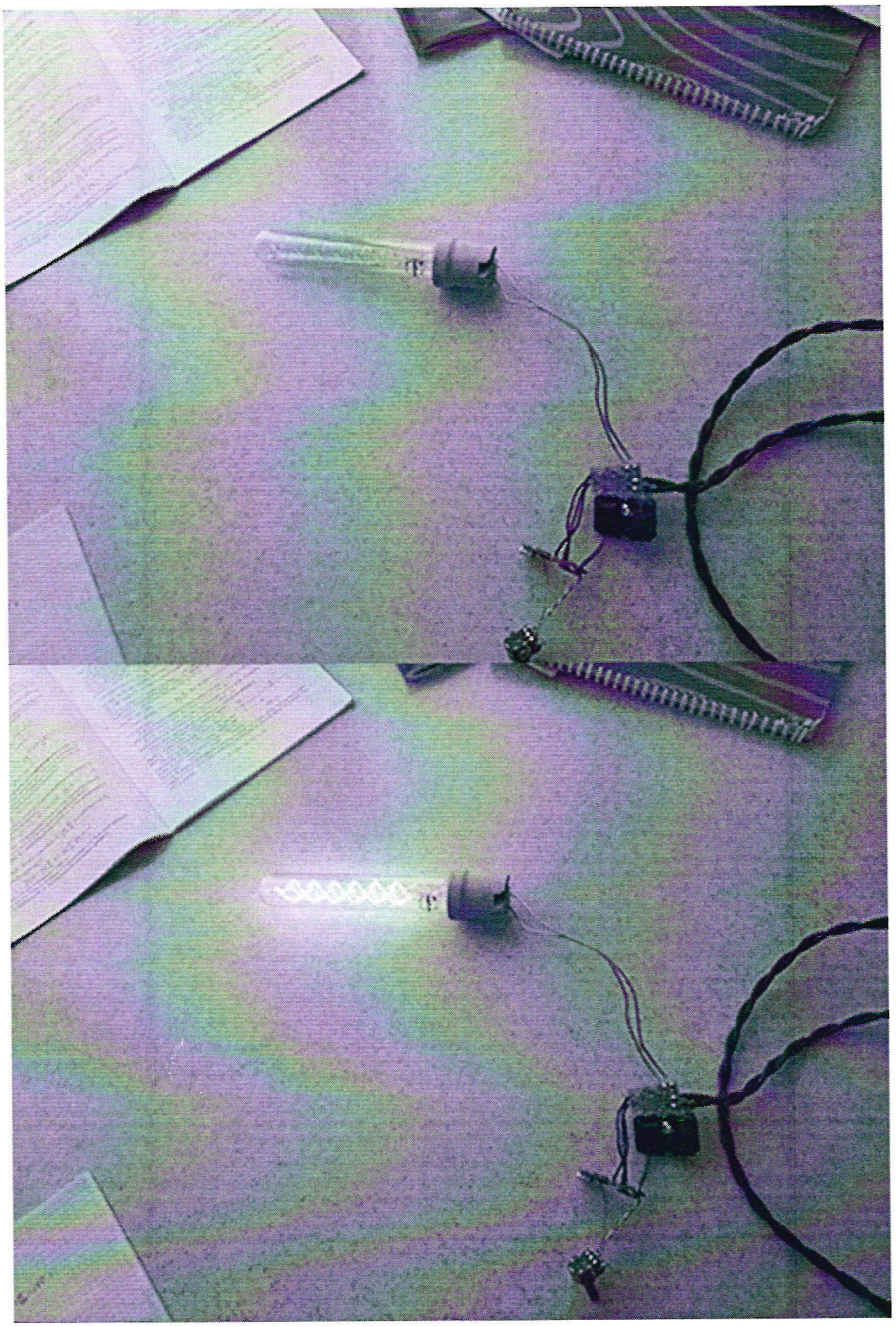


Рисунок 2.3 – Демонстрация работы

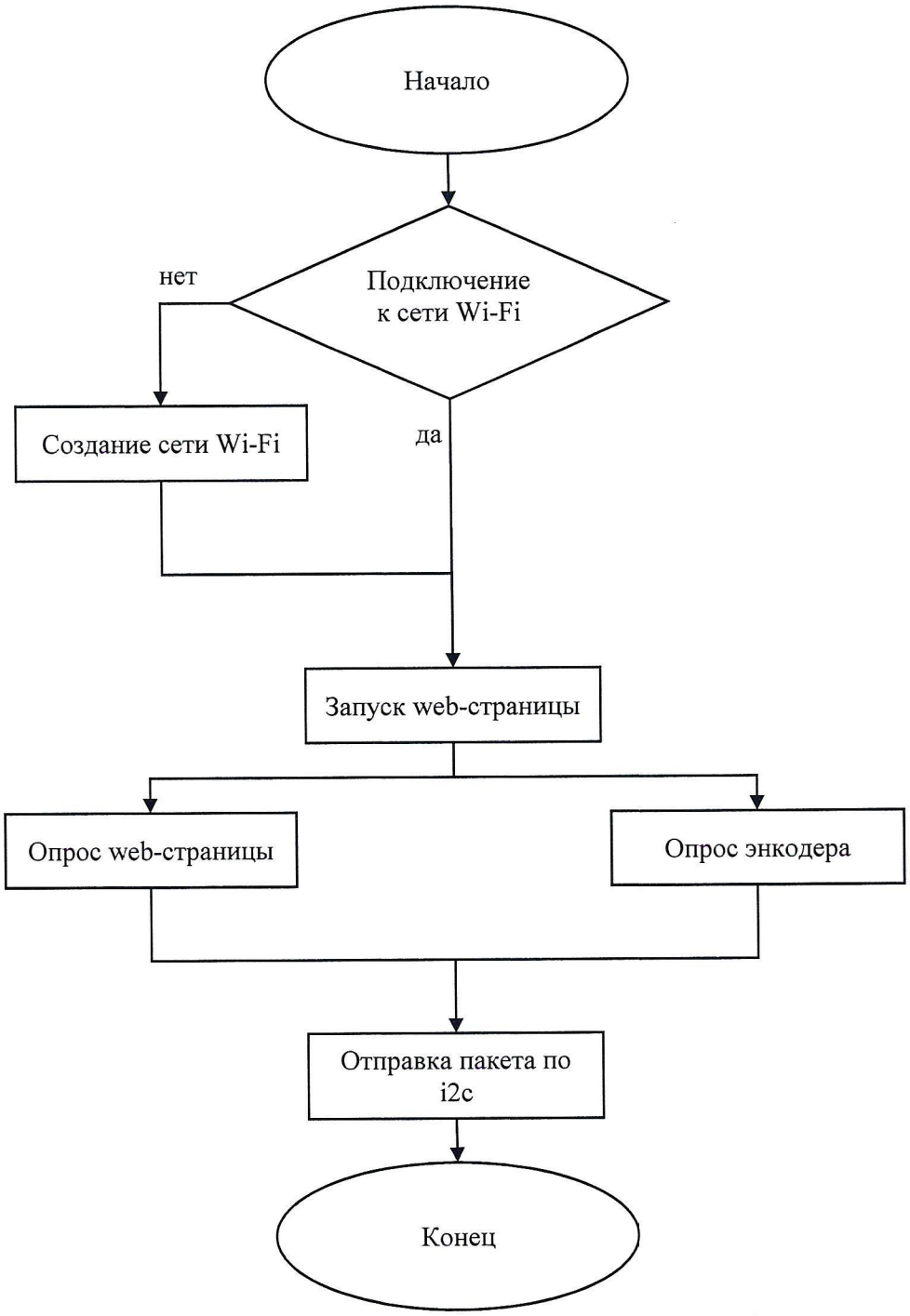


Рисунок 2.2 – Блок-схема алгоритма

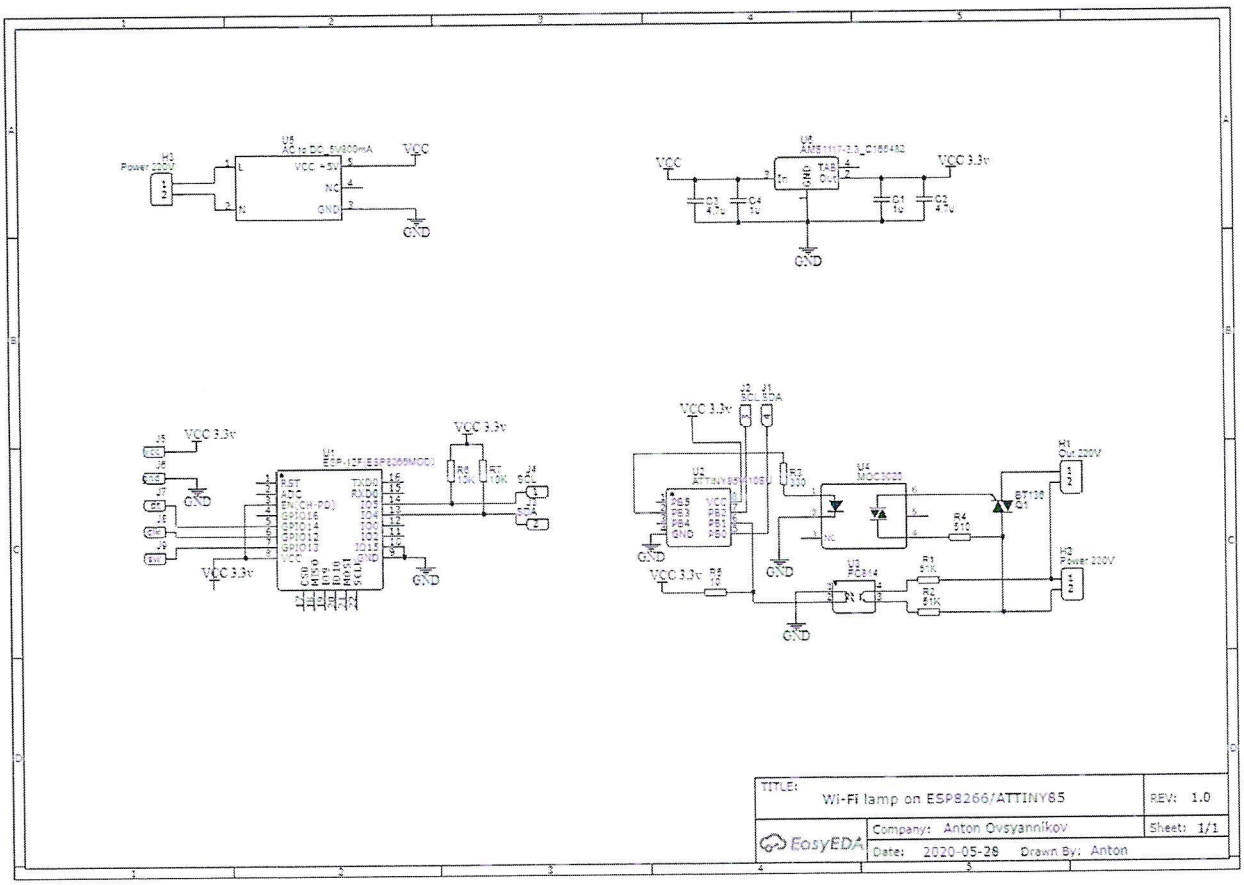


Рисунок 2.3 – Принципиальная схема устройства.

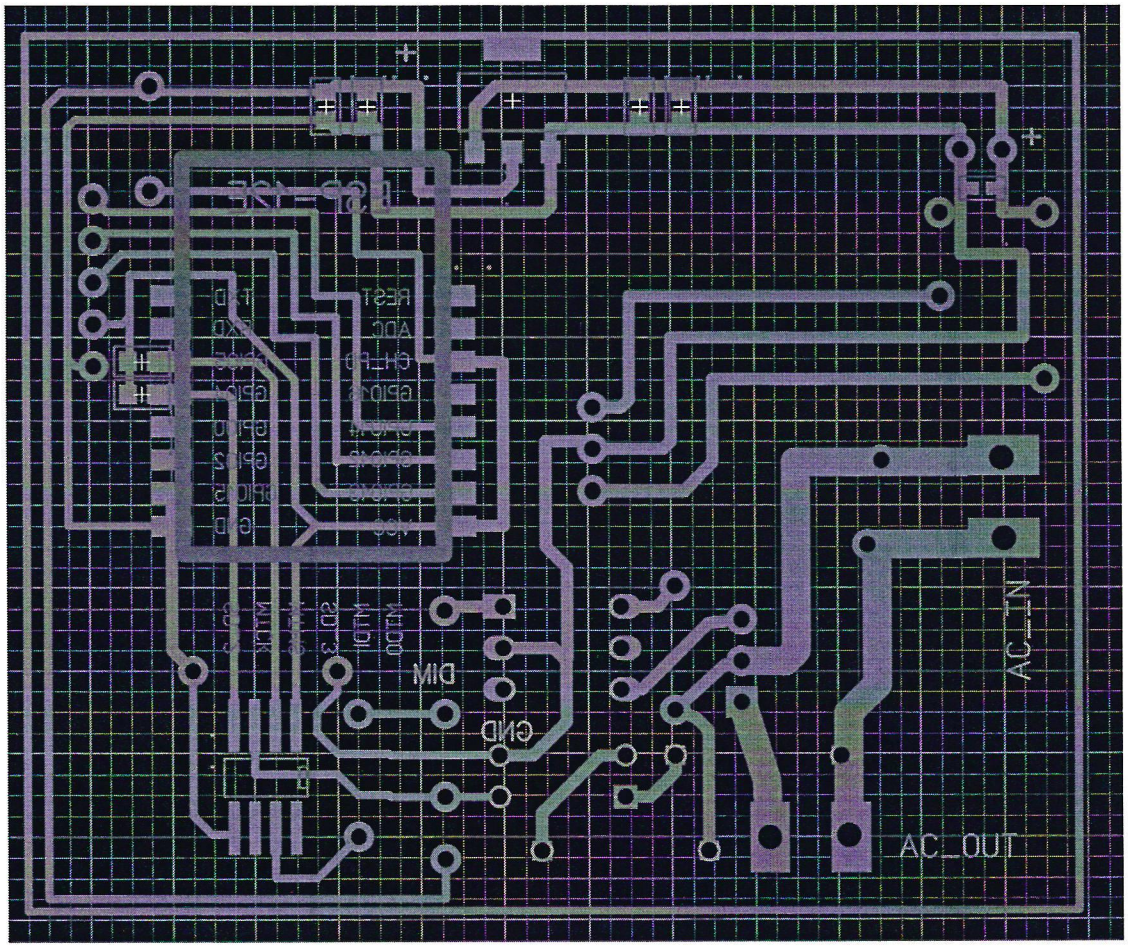


Рисунок 2.4 – Чертёж печатной платы устройства.

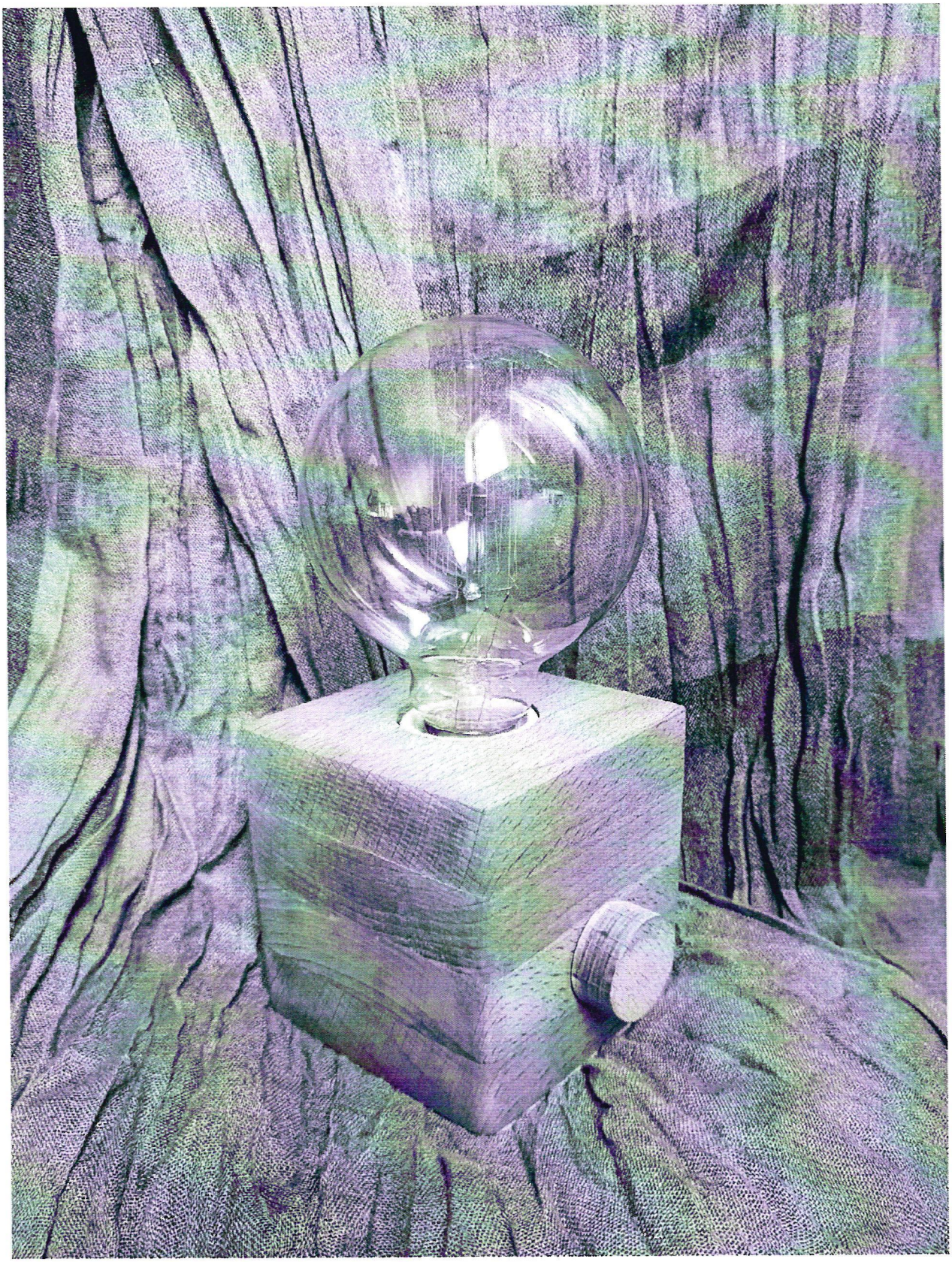


Рисунок 2.5 – Конечный результат устройства.

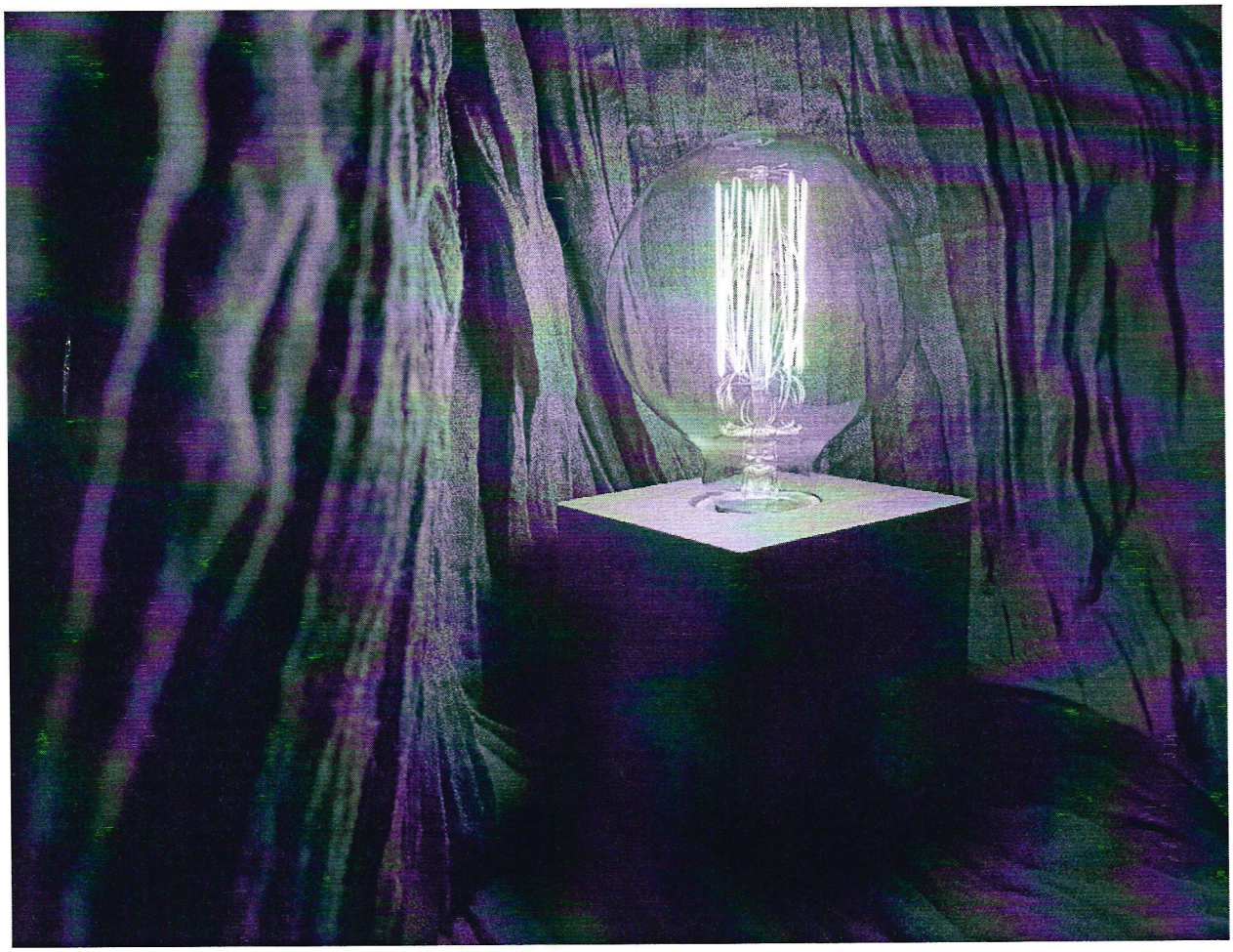


Рисунок 2.6 – Конечный результат устройства.

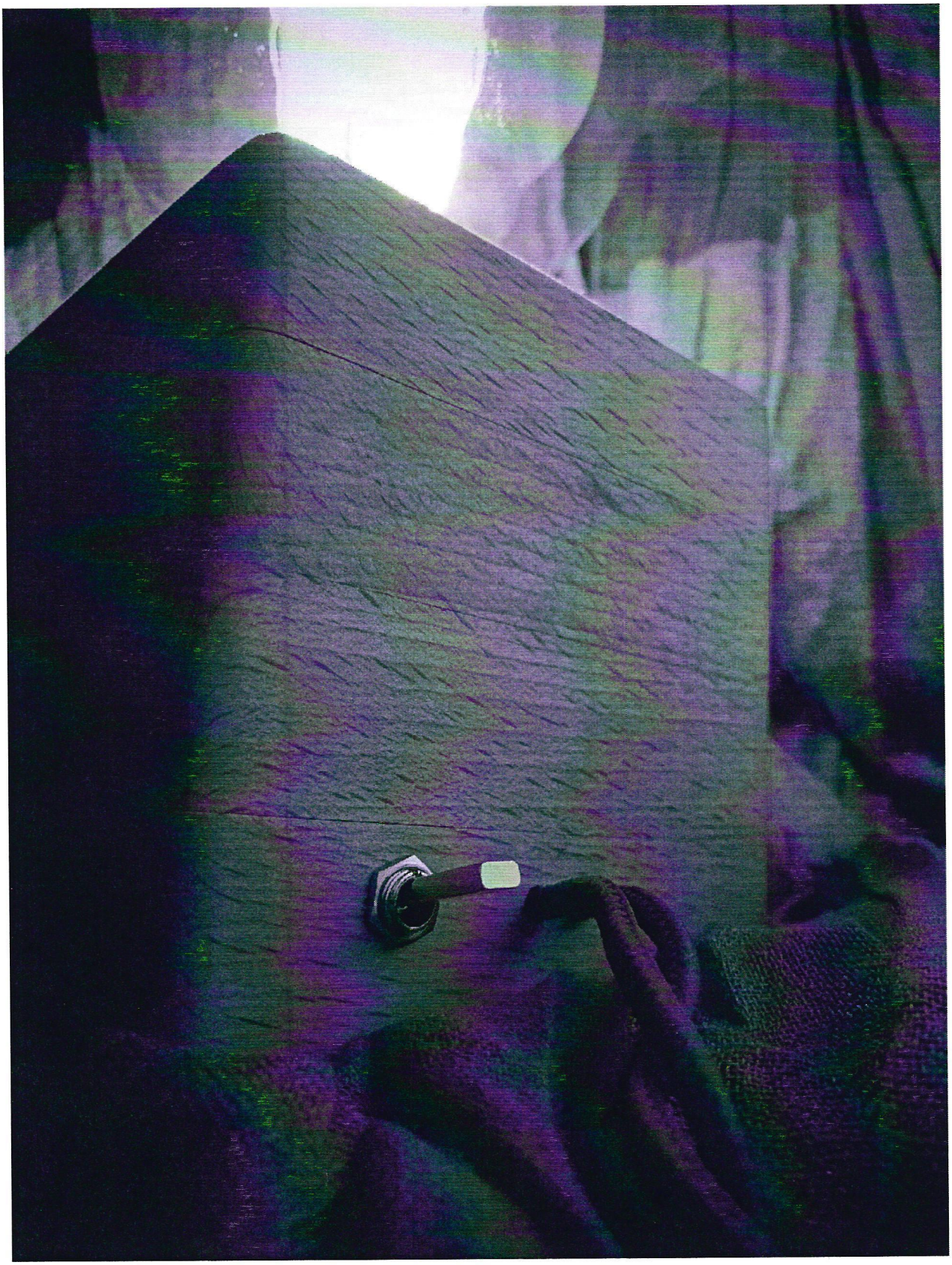


Рисунок 2.7 – Конечный результат устройства.

2.3 Описание логической структуры

Название файла с управляющей программой для микроконтроллера ESP8266: I2C_Dimmer_ESP8266.ino

Название файла с управляющей программой для микроконтроллера ATTINT85: Attiny85_Dimmer.ino

В теле программы “I2C_Dimmer_ESP8266.ino” реализуется развертка веб сервера в локальной сети Wi-Fi с web страницей управления устройством, обработкой физического энкодера, отправка значений по i2c микроконтроллеру ATTiny85.

В теле программы “Attiny85_Dimmer.ino” реализуется прием пакетов по линии i2c, отслеживание прохождения нуля переменным током, управление симистором для коммутации лампы.

Текст программы “I2C_Dimmer_ESP8266.ino” приведен в Приложении А.

Текст программы “Attiny85_Dimmer.ino” приведен в Приложении Б.

2.4 Используемые технические средства

Микроконтроллеры ESP8266, ATTINY85

2.5 Вызов и загрузка

Прошивка микроконтроллера ESP8266 через программатор UATR

Прошивка микроконтроллера ATTINY85 через программатор ISP

3 Руководство оператора

3.1 Назначение программы;

Диммирование нагрузки переменного тока с возможностью регулировки физическим энкодером, web-интерфейсом.

3.2 Условия выполнения программы;

3.3 Выполнение программы;

Плавная регулировка яркости лампы накаливания.

ПРИЛОЖЕНИЕ А

(обязательное)

Текст программы “ I2C_Dimmer_ESP8266.ino”

Файл Правка Скетч Инструменты Помощь

I2C_Dimmer_ESP8266 | FS | FileConfig | SSDP | Setup | Voids | WiFi | update

```
1  #include <GyverEncoder.h>
2  #include <ESP8266WiFi.h>
3  #include <ESP8266WebServer.h>
4  #include <ESP8266SSDP.h>
5  #include <FS.h>
6  #include <DNSServer.h>
7  #include <ArduinoJson.h>
8  #include <Wire.h>
9
10 #define device (0x41)
11
12
13
14 #define CLK 12
15 #define DT 14
16 #define SW 13
17
18 int level;
19 int state;
20 int rang;
21 int ocr;
22 boolean pres;
23 int buttonState = 0;
24
25 String data, Wcolor, Dcolor, XML;
26
27 String _ssid = "            ";
28 String _password = "          ";
29 String _ssidAP = "WiFi";
30 String _passwordAP = "";
31 String SSDP_Name="Wireless Pc Power";
32 String jsonConfig = "{}";
33 IPAddress apIP(              );
34 byte arduino_mac[] = {                                    };
35
36 Encoder encl(CLK, DT, SW);
37
38 ESP8266WebServer HTTP(80);
39 File fsUploadFile;
40
41 void loop(){
42  HTTP.handleClient();
43
44  encl.tick();  // отработка
45
46 if (encl.isRight()){
47   if (level < 230){
48    level = level + 6;
49   }
50   maping(level);
51 }
52 if (encl.isLeft()){
53   if (level > 0){
54     level = level - 6;
55   }
56   maping(level);
57 }
58 if (encl.isPress()){
59   Serial.println("Press");
60   if(pres == true){
61     rang = 255;
62     Power(rang);
63   }
64   else{
65     if (ocr > 0){
```

Файл Правка Скетч Инструменты Помощь

I2C_Dimmer_ESP8266 | FS | FileConfig | SSDP | Setup | Voids | WiFi | update

```
// Инициализация FFS
void FS_init(void) {
  SPIFFS.begin();
  {
    Dir dir = SPIFFS.openDir("/");
    while (dir.next()) {
      String fileName = dir.fileName();
      size_t fileSize = dir.fileSize();
    }
  }
  //HTTP страницы для работы с FFS
  //list directory
  HTTP.on("/list", HTTP_GET, handleFileList);
  //загрузка редактора editor
  HTTP.on("/edit", HTTP_GET, []() {
    if (!handleFileRead("/edit.htm")) HTTP.send(404, "text/plain", "FileNotFound");
  });
  //Создание файла
  HTTP.on("/edit", HTTP_PUT, handleFileCreate);
  //Удаление файла
  HTTP.on("/edit", HTTP_DELETE, handleFileDelete);
  //first callback is called after the request has ended with all parsed arguments
  //second callback handles file uploads at that location
  HTTP.on("/edit", HTTP_POST, []() {
    HTTP.send(200, "text/plain", "");
  }, handleFileUpload);
  //called when the url is not defined here
  //use it to load content from SPIFFS
  HTTP.onNotFound([]() {
    if (!handleFileRead(HTTP.uri()))
      HTTP.send(404, "text/plain", "FileNotFound");
  });
}
// Здесь функции для работы с файловой системой

bool handleFileRead(String path) {
  if (path.endsWith("/")) path += "index.htm";
  String contentType = loadFromSpiffs(path);
  String pathWithGz = path + ".gz";
  if (SPIFFS.exists(pathWithGz) || SPIFFS.exists(path)) {
    if (SPIFFS.exists(pathWithGz))
      path += ".gz";
    File file = SPIFFS.open(path, "r");
    size_t sent = HTTP.streamFile(file, contentType);
    file.close();
    return true;
  }
  return false;
}

void handleFileUpload() {
  if (HTTP.uri() != "/edit") return;
  HTTPUpload& upload = HTTP.upload();
  if (upload.status == UPLOAD_FILE_START) {
    String filename = upload.filename;
    if (!filename.startsWith("/")) filename = "/" + filename;
    fsUploadFile = SPIFFS.open(filename, "w");
    filename = String();
  } else if (upload.status == UPLOAD_FILE_WRITE) {
    //DBG_OUTPUT_PORT.print("handleFileUpload Data: "); DBG_OUTPUT_PORT.println(upload.currentSize);
    if (fsUploadFile)
      fsUploadFile.write(upload.buf, upload.currentSize);
  } else if (upload.status == UPLOAD_FILE_END) {
    if (fsUploadFile)
      fsUploadFile.close();
```

Файл Правка Скетч Инструменты Помощь

I2C_Dimmer_ESP8266 | FS | FileConfig | SSDP | Setup | Voids | WiFi | update

```
1  // Загрузка данных сохраненных в файл  config.json
2  bool loadConfig() {
3    // Открываем файл для чтения
4    File configFile = SPIFFS.open("/config.json", "r");
5    if (!configFile) {
6    // если файл не найден
7      Serial.println("Failed to open config file");
8    //  Создаем файл запиав в него аные по умолчанию
9      saveConfig();
10     return false;
11   }
12   // Проверяем размер файла, будем использовать файл размером меньше 1024 байта
13   size_t size = configFile.size();
14   if (size > 1024) {
15     Serial.println("Config file size is too large");
16     return false;
17   }
18
19 // загружаем файл конфигурации в глобальную переменную
20   jsonConfig = configFile.readString();
21   // Резервируем память для json обекта буфер может рости по мере необходимти предпочтительно для ESP8266
22     DynamicJsonBuffer jsonBuffer;
23   //  вызовите парсер JSON через экземпляр jsonBuffer
24   //  строку возьмем из глобальной переменной String jsonConfig
25     JsonObject& root = jsonBuffer.parseObject(jsonConfig);
26   // Теперь можно получить значения из root
27     _ssid = root["ssidName"].as<String>();
28     _password = root["ssidPassword"].as<String>();
29     return true;
30 }
31
32 // Запись данных в файл config.json
33 bool saveConfig() {
34   // Резервируем память для json обекта буфер может рости по мере необходимти предпочтительно для ESP8266
35   DynamicJsonBuffer jsonBuffer;
36   //  вызовите парсер JSON через экземпляр jsonBuffer
37   JsonObject& json = jsonBuffer.parseObject(jsonConfig);
38   // Заполняем поля json
39   json["ssidName"] = _ssid;
40   json["ssidPassword"] = _password;
41   // Помещаем созданный json в глобальную переменную json.printTo(jsonConfig);
42   json.printTo(jsonConfig);
43   // Открываем файл для записи
44   File configFile = SPIFFS.open("/config.json", "w");
45   if (!configFile) {
46     //Serial.println("Failed to open config file for writing");
47     return false;
48   }
49   // Записываем строку json в файл
50   json.printTo(configFile);
51   return true;
52   }
```

Файл Правка Скетч Инструменты Помощь

I2C_Dimmer_ESP8266 | FS | FileConfig | SSDP | Setup | Voids | WiFi | update

```
1  void layer1() {
2    if(rang == 255){
3      data = rang - 255;
4    }
5    else{
6    data = ocr / 2;
7  }
8  background();
9  }
10
11 void background() {
12   XML ="<?xml version='1.0'?>";
13   XML+="<data>";
14   XML+=data;
15   XML+="</data>";
16   HTTP.send(200, "text/xml", XML);
17 }
18
19 void buttonclick() {
20   state = HTTP.arg("state").toInt();
21   HTTP.send(200, "text/plain", "OK");
22  level = state;
23   maping(level);
24 }
25
26 void maping(int level){
27   ocr = level;
28   //ocr = map(level, 0, 230, 0, 240);
29   lev(ocr);
30   Serial.print("ocr:");
31   Serial.println(ocr);
32
33 }
34
35 void lev(int ocr){
36     if (ocr <= 0){
37    rang = 255;
38    pres = false;
39   }
40   else {
41    rang = ocr;
42    pres = true;
43   }
44   //Serial.println(level);
45   Power(rang);
46   Serial.print("rang:");
47   Serial.println(rang);
48 }
49
50 void Power(int rang) {
51
52   Wire.beginTransmission(device);
53   Wire.write(rang);
54   Wire.endTransmission();
55   Wire.requestFrom(device, 1);
56 }
57
58 void handle_Set_Ssid() {
59   _ssid = HTTP.arg("ssid");            // Получаем значение ssid из запроса сохраняем в глобальной переменной
60   _password = HTTP.arg("password");    // Получаем значение password из запроса сохраняем в глобальной переменной
61   saveConfig();
62   HTTP.send(200, "text/plain", "OK");
63
64 }
65
```

Файл Правка Скетч Инструменты Помощь

I2C_Dimmer_ESP8266 | FS | FileConfig | SSDP | Setup | Voids | WiFi | update

```
1  void WIFIinit() {
2  IPAddress ipAdress(192,168,1,38);
3  IPAddress gateway(192,168,1,1);
4  IPAddress subnet(255,255,255,0);
5    // Попытка подключения к точке доступа
6    WiFi.mode(WIFI_STA);
7    byte tries = 11;
8    WiFi.begin(_ssid.c_str(), _password.c_str());
9    WiFi.config(ipAdress, gateway, subnet);
10   // Делаем проверку подключения до тех пор пока счетчик tries
11   // не станет равен нулю или не получим подключение
12   while (--tries && WiFi.status() != WL_CONNECTED)
13   {
14     Serial.print(".");
15     delay(1000);
16   }
17   if (WiFi.status() != WL_CONNECTED)
18   {
19     // Если не удалось подключиться запускаем в режиме AP
20     Serial.println("");
21     Serial.println("WiFi up AP");
22     StartAPMode();
23   }
24   else {
25     // Иначе удалось подключиться отправляем сообщение
26     // о подключении и выводим адрес IP
27     Serial.println("");
28     Serial.println("WiFi connected");
29     Serial.println("IP address: ");
30     Serial.println(WiFi.localIP());
31     Serial.println(WiFi.SSID());
32   }
33 }
34
35 bool StartAPMode()
36 { // Отключаем WIFI
37   WiFi.disconnect();
38   // Меняем режим на режим точки доступа
39   WiFi.mode(WIFI_AP);
40   // Задаем настройки сети
41   WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
42   // Включаем WIFI в режиме точки доступа с именем и паролем
43   // хронящихся в переменных _ssidAP _passwordAP
44   WiFi.softAP(_ssidAP.c_str(), _passwordAP.c_str());
45   return true;
46 }
```

Файл Правка Скетч Инструменты Помощь

Проверить

I2C_Dimmer_ESP8266 | FS | FileConfig | SSDP | Setup | Voids | WiFi | update

```
1  void update(void){
2   //Обновление
3    HTTP.on("/update", HTTP_POST, []() {
4      HTTP.sendHeader("Connection", "close");
5      HTTP.sendHeader("Access-Control-Allow-Origin", "*");
6      HTTP.send(200, "text/plain", (Update.hasError()) ? "FAIL" : "OK");
7      ESP.restart();
8    }, []() {
9      HTTPUpload& upload = HTTP.upload();
10     if (upload.status == UPLOAD_FILE_START) {
11       Serial.setDebugOutput(true);
12       WiFiUDP::stopAll();
13       Serial.printf("Update: %s\n", upload.filename.c_str());
14       uint32_t maxSketchSpace = (ESP.getFreeSketchSpace() - 0x1000) & 0xFFFFF000;
15       if (!Update.begin(maxSketchSpace)) { //start with max available size
16         Update.printError(Serial);
17       }
18     } else if (upload.status == UPLOAD_FILE_WRITE) {
19       if (Update.write(upload.buf, upload.currentSize) != upload.currentSize) {
20         Update.printError(Serial);
21       }
22     } else if (upload.status == UPLOAD_FILE_END) {
23       if (Update.end(true)) { //true to set the size to the current progress
24         Serial.printf("Update Success: %u\nRebooting...\n", upload.totalSize);
25       } else {
26         Update.printError(Serial);
27       }
28       Serial.setDebugOutput(false);
29     }
30     yield();
31   });
32 }
```

Общее количество строк в программе: 506

ПРИЛОЖЕНИЕ Б

(информационное)

Текст программы “Attiny85_Dimmer.ino”

Файл Правка Скетч Инструменты Помощь

Attiny85_Dimmer

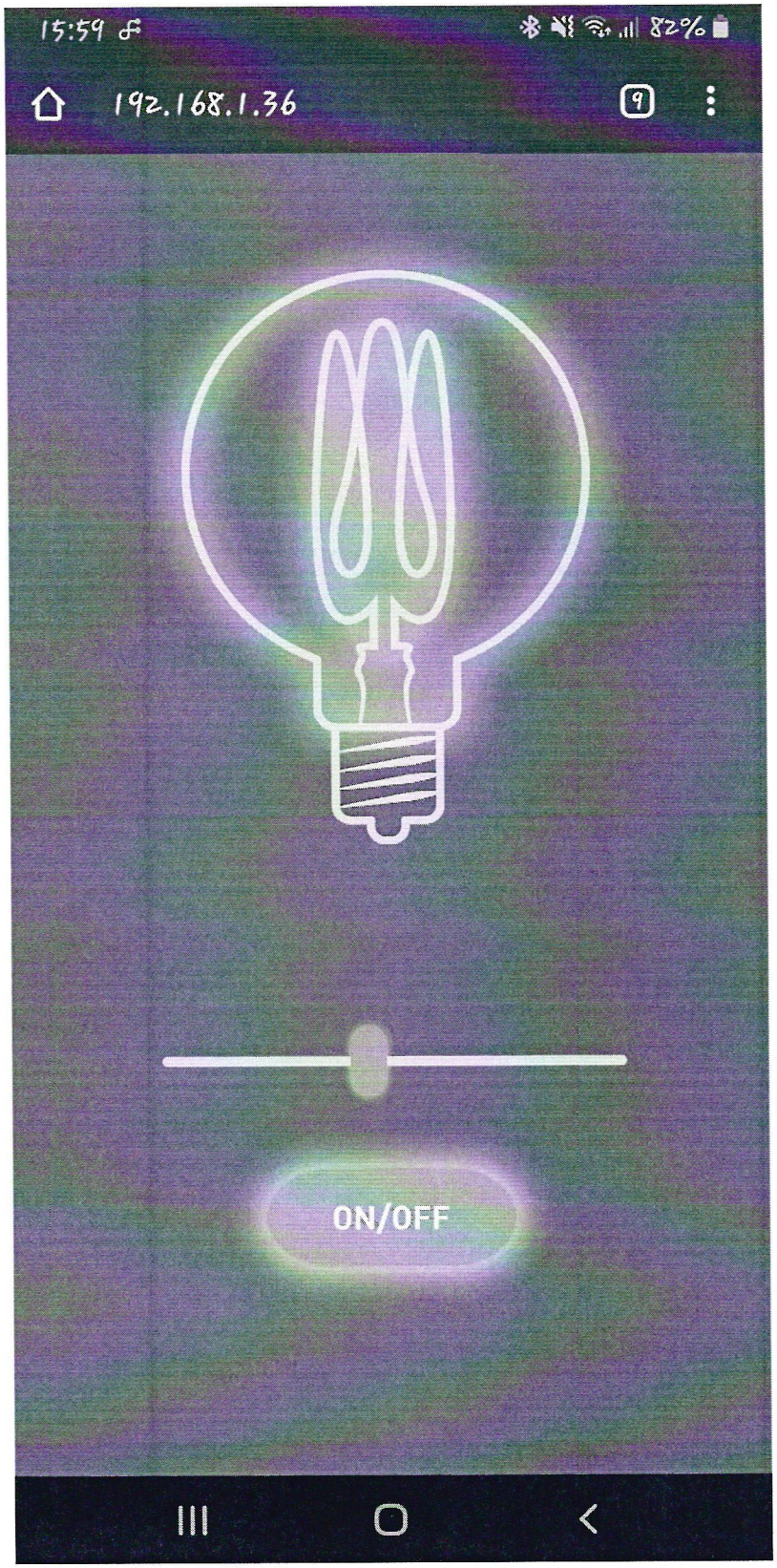
```
1  #include <TinyWireS.h>
2  #define I2C_SLAVE_ADDR (0x41)
3  #define DETECT 1        //zero cross detect, interrupt 0, is physical pin 6
4  #define GATE 3          //triac gate is physical pin 2
5  #define PULSE 5         //trigger pulse width (counts)
6  // I2C pins: PB2 - SCL, PB0 - SDA
7  volatile byte i2cValue, ocr, jj;
8  void setup() {
9    // set up pins
10
11   TinyWireS.begin(I2C_SLAVE_ADDR);
12   DDRB &= ~(1 << DETECT); //zero cross detect
13   PORTB |= (1 << DETECT); //enable pull-up resistor
14   DDRB |= 1 << GATE; //triac gate control
15   GIMSK = 1 << PCIE;
16   PCMSK = 1 << PCINT1;
17
18   OCR1A = 50;    //initialize the comparator
19   TIMSK = _BV(OCIE1A) | _BV(TOIE1);  //interrupt on Compare Match A | enable timer overflow interrupt
20   TinyWireS.onRequest(requestEvent);
21   TCCR1=0;
22   while((PINB&(1<<PINB1))!=0);
23   while((PINB&(1<<PINB1))==0);
24   for(uint8_t ii=0; ii<255; ii++) jj++;
25   TIFR=0xff;
26   for(uint8_t ii=0; ii<200; ii++) jj++;
27   sei();  // enable interrupts
28 }
29
30 ISR (PCINT0_vect) {
31 //for(uint8_t ii=0; ii<200; ii++) jj++;
32 //if (PINB&(1<<PINB1)) { return;}
33 if (TCCR1>0) return;
34 TCCR1 = 0;
35 TCNT1 = 0;                  //reset timer - count from zero
36 OCR1A=ocr;
37 TCCR1 = B00001010;          // prescaler on 1024, see table 12.5 of the tiny85 datasheet
38
39 }
40
41 ISR(TIMER1_COMPA_vect) {    //comparator match
42   PORTB |= (1 << GATE); //set triac gate to high
43   TCCR1 =0;
44   TCNT1 = 255 - PULSE;      //trigger pulse width, when TCNT1=255 timer1 overflows
45   TCCR1 = B00001010;
46 }
47
48 ISR(TIMER1_OVF_vect) {       //timer1 overflow
49   PORTB &= ~(1 << GATE); //turn off triac gate
50   TCCR1 = 0;                  //disable timer stop unintended triggers
51 }
52
53 void loop() {    // use analog input to set the dimmer
54
55   if (TinyWireS.available())  i2cValue = TinyWireS.receive();
56   ocr = map(i2cValue, 0, 254, 245, 0);
57 //  ocr=100;
58   TinyWireS_stop_check();
59 }
60
61 void requestEvent() {
62
63   TinyWireS.send(i2cValue);
64 }
```

Общее количество строк в программе: 64

ПРИЛОЖЕНИЕ В

(информационное)

Web-интерфейс



Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

УТВЕРЖДАЮ	СОГЛАСОВАНО
Декан ФЭУ	Заведующий кафедрой ПЭ
_____ А.С. Гудим	_____ Н.Н. Любушкина
(подпись)	(подпись)
«19» 03 2020 г.	«19» 03 2020 г.

АКТ

о приемке в эксплуатацию управляющей программы для автоматизированной/роботизированной системы

«Настольная лампа переменного тока с управлением по Wi-Fi.»

г. Комсомольск-на-Амуре «19» 03 2020г.

Комиссия в составе представителей:

заказчика

- С.И. Сухоруков – руководитель проекта / СКБ ,
- Н.Н. Любушкина – Заведующий кафедрой ПЭ,
- А.С. Гудим – декан ФЭУ

исполнителя

- А.Р. Овсянников – 8МРб-1

составила акт о нижеследующем:

«Исполнитель» передает управляющую программу для автоматизированной/роботизированной системы «Настольная лампа переменного тока с управлением по Wi-Fi.», в составе:

Программное обеспечение:

- Текст управляющей программы

Эксплуатационная документация:

- Паспорт управляющей программы для автоматизированной/роботизированной системы

Управляющая программа для автоматизированной/роботизированной системы «Настольная лампа переменного тока с управлением по Wi-Fi.» прошла апробацию с «16» 03.20 по «29» 03 20 20 г. и признана годной к эксплуатации. Были протестированы все режимы функционирования, отказы системы, а также аварийные отключения по вине системы не наблюдались.

Руководитель СКБ / проекта	Ответственный исполнитель
_____/С.И. Сухоруков/	_____ /А.Р. Овсянников/

Таблица учета проектной работы в учебных дисциплинах

Дисциплина	Форма учтенной работы (номер ЛР, КП, КР, РГЗ, зачет, итоговая оценка , экзамен)	Преподаватель (дата, ФИО, подпись)	Примечание (ЗУН полученные при выполнении проекта)