Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка автономной системы управления освещением для парков и придомовых участков»

по дисциплине «Проектный практикум»

Заказчик: Папуловская Наталья Владимировна _____

Куратор: Папуловская Наталья Владимировна
доцент, к.пед.н, доцент ИРИТ-РтФ _____

Студенты команды _____

Колмаков К.В. _____

Коростелев И.Е. _____

Демин И.Д. _____

Новиков Е.С. _____

Петров И.Е. _____

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Парковые зоны требуют качественного и надёжного освещения для обеспечения безопасности посетителей в тёмное время суток, а также для создания комфортной и привлекательной атмосферы.

Основная задача проекта — разработать полностью автономную систему освещения парковых зон с возможностью удаленного управления из веб-интерфейса.

Ключевые характеристики системы:

– Управление освещением: датчики движения, диммеры и фоторезисторы.

– Мониторинг, обслуживание и управление: беспроводная система сбора телеметрии и удалённое управление через веб-интерфейс.

Целью работы является разработка автономной системы управления освещением для парковых зон, которая будет иметь возможность контроля и управления через веб-интерфейс.

1 Основная часть

1.1 Обзор аналогов автоматических систем освещения парковых зон

В последние годы на рынке появилось несколько решений, направленных на создание энергоэффективных и автономных систем наружного освещения для городских и пригородных парков. Основные характеристики таких систем включают использование возобновляемых источников энергии, интеллектуальные механизмы управления светильниками и возможности удаленного мониторинга:

а) Philips CityTouch (Signify): платформа IoT-управления уличным освеще-нием, позволяющая динамически изменять уровни освещенности на ос-нове расписаний и данных датчиков света и движения. Система интегрируется с солнечными панелями и аккумуляторами для обеспечения автономности в удаленных зонах без доступа к электросети.

б) Schréder SmartPARK: решение, сочетающее в себе солнечные панели и светодиодные светильники с интеллектуальным контроллером. Предусматривает адаптивную регулировку яркости в зависимости от загруженности парка (датчики движения) и погодных условий (фотореле).

в) SolarPark Light (SolarLED): комплекс на базе ветрогенератора и солнеч-ных панелей с возможностью резервного питания от сети. Отличается модульной конструкцией и возможностью быстрого монтажа без залив-ки фундамента.

г) Локальные разработки: многие муниципалитеты и стартапы реализуют собственные решения на базе микроконтроллеров Arduino/ESP32 в соче-тании с недорогими солнечными панелями и Li-Ion аккумуляторами, ис-пользуя беспроводные протоколы связи (LoRaWAN, Zigbee) для централизованного управления.

Каждое из указанных решений демонстрирует успешное применение ком-бинации возобновляемой энергии и интеллектуального управления, однако нередко требует дорогой инфраструктуры или имеет ограничения по масшта-бируемости и гибкости монтажа.

1.2 Обзор решений в программировании микроконтроллеров

Ключевой элемент автономных систем освещения — контроллер, отвечаю-щий за сбор телеметрии, управление зарядом аккумуляторов и включе-ние/выключение светильников. Для реализации подобных задач чаще всего используются следующие подходы:

д) Платформы и аппаратные модули:

– Arduino: популярная платформа для быстрого прототипирования. Модели типа Arduino Uno или Nano обычно оснащаются базовым набором GPIO и могут работать с датчиками освещенности и движения.

– ESP8266: обеспечивает встроенный Wi-Fi/Bluetooth, поддерживает энергосберегающие режимы и может взаимодействовать с облачными сервисами для сбора и визуализации данных.

– STM8266 (ARM Cortex-M): предпочтительны для промышленных решений благодаря высокой производительности, наличию встроенных АЦП и низкому энергопотреблению.

е) Программные подходы:

– RTOS (FreeRTOS, Zephyr): использование операционных систем реального времени позволяет одновременно обрабатывать несколько задач — измерение параметров аккумулятора, опрос датчиков, управление PWM для светодиодов, обмен данными по LoRaWAN/MQTT.

– Bare-metal программирование: для упрощенных систем, где приоритетом является минимальное энергопотребление, используют

циклические опросы и прерывания без «тяжелой» инфраструктуры RTOS.

ж) Протоколы связи и удаленный мониторинг:

– MQTT over Wi-Fi/GSM: стандарт для IoT-устройств, позволяет передавать информацию о состоянии системы в облачный сервис.

– LoRaWAN: обеспечивает энергоэффективную сотовую связь на большие расстояния для парков без покрытия сети.

– Modbus RTU/Over Ethernet: применим в более традиционных промышленных сетях для взаимодействия с SCADA-системами.

з) Алгоритмы управления энергией:

– MPPT (Maximum Power Point Tracking): алгоритмы для оптимального заряда солнечных панелей.

– SOC (State of Charge) estimation: вычисление текущего состояния заряда батареи с учетом температуры и циклов разряда.

– Adaptive Dimming: интеллектуальное снижение яркости в период низкой посещаемости парка.

Комбинирование аппаратных платформ и современных программных решений позволяет создавать гибкие и масштабируемые контроллеры для автономных систем освещения, которые легко интегрируются в существующую инфраструктуру и обеспечивают длительную бесперебойную работу.

1.3 Разработка системы

1.3.1 Постановка задачи

В рамках проекта необходимо разработать комплексное решение автоном-ной системы освещения парковых зон, которые встречаются в городской и пригородной инфраструктуре, обеспечивающее:

а) Интеллектуальное управление яркостью и включением осветительных приборов в зависимости от времени суток, уровня естественного освеще-ния и присутствия людей.

б) Удалённый мониторинг и управление через веб-интерфейс: веб-интерфейс отображает включена/выключена система, в каком режиме она находится, состояние зон освещения, позволяет переключать режи-мы работы системы, включать/выключать систему, просматривать логи системы

1.3.2 Описание предметной области

Условия функционирования системы:

а) Электропитание:

– Система подключается к стандартной электросети (220 В).

– Бесперебойное питание важно для стабильной работы контроллера, дат-чиков и веб-интерфейса.

б) Сетевое подключение:

1) Микроконтроллер ESP8266 подключается к Wi-Fi сети, параметры подключения задаются в прошивке (ssid, password).

2) Стабильное соединение необходимо для:

– Работы веб-интерфейса управления.

– Синхронизации времени через NTP-сервер.

– Логирования и удалённого мониторинга.

в) Окружающая среда:

– Устройство должно находиться в защищённом корпусе с уровнем защиты не ниже IP54 для эксплуатации на улице.

– Температурный диапазон работы компонентов должен соответствовать климатическим условиям региона.

– Датчик движения должен быть установлен так, чтобы не срабатывать на животных или мелкие объекты.

– Датчик освещённости должен быть размещён так, чтобы на него не по-падал искусственный свет от собственного источника.

г) Условия включения освещения:

1) Освещение включается при соблюдении одновременно двух условий:

– Уровень освещённости превышает заданный порог.

– Зафиксировано движение.

2) При этом включение осуществляется на фиксированное время после по-следнего обнаруженного движения.

3) При продолжительном движении — свет остаётся включённым.

4) Если уровень освещённости ниже порога, система игнорирует сигналы движения и отключает свет.

д) Условия выключения освещения

- Свет отключается:

o по истечении заданного времени с момента последнего движения;

o если уровень освещённости стал ниже порога.

е) Режим MANUAL:

- Веб-интерфейс позволяет включать и выключать свет вручную.
- При включённом ручном режиме автоматические функции полностью блокируются до повторного переключения.
- Переключение режимов также осуществляется через веб-интерфейс.

ж) Веб-интерфейс:

- Доступ к веб-сервису осуществляется из локальной сети
- Функции:

o Включение/выключение освещения вручную.

o Переключение между режимами.

o Отслеживание состояния системы

o Просмотр логов

з) Логирование:

- Все важные события (включение/выключение света, движение, смена ре-жима) фиксируются в журнале доггирования.
- Каждое событие сопровождается отметкой времени.

1.3.3 Проектирование системы

• Источник питания — сеть 220 В через понижающий блок питания (адап-тер 5 В или 3.3 В) для питания контроллера и датчиков.

• Микроконтроллер ESP8266 с интегрированным Wi Fi/Bluetooth как цен-тральный узел управления.

• Датчики освещенности для фиксации уровня окружающего света.

• Датчики движения для обнаружения присутствия посетителей и актива-ции светильников.

• Осветительные приборы мощностью 10–60 Вт с возможностью димми-рования через ШИМ-пины ESP8266.

• Модуль связи — встроенный Wi Fi для отправки телеметрии на сервер или через MQTT-брокер.

• Схема прототипа системы, имитирующий 3 зоны освещения парка пред-ставлена на рисунке 1:

Рис. 1.

• На рисунке 2 представлен прототип системы для одной зоны освещения:

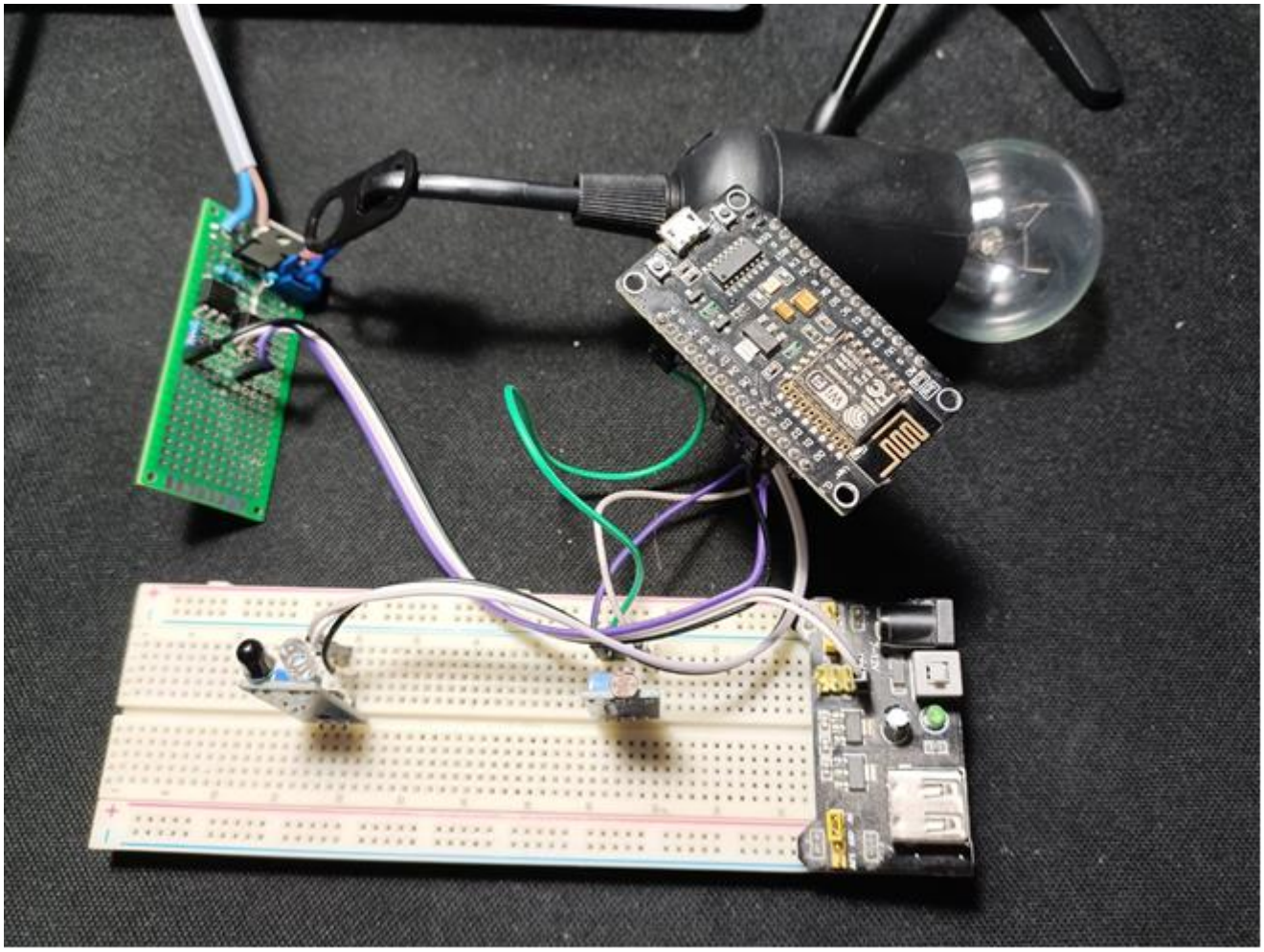

Рис. 2. – Прототип системы для одной зоны

• На рисунке 3 представлен самодельный диммер, использующийся в системе:

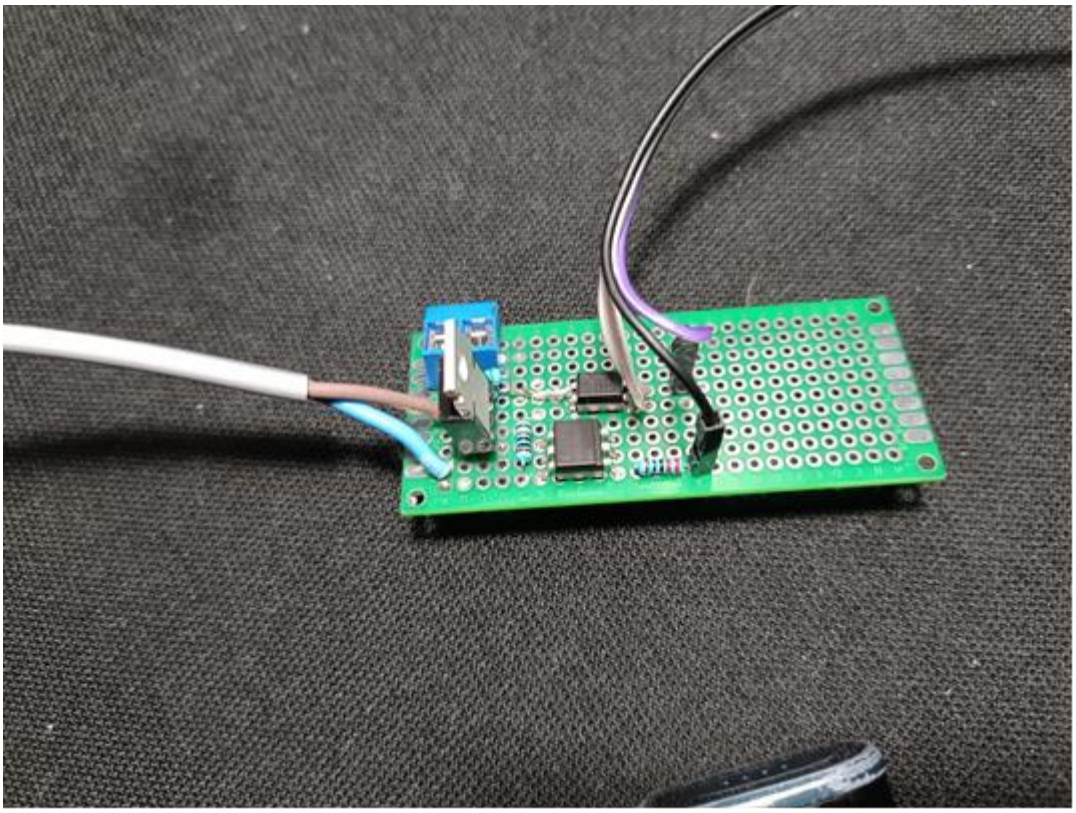

Рис. 3. – Самодельный диммер

1.3.4 Программирование системы

Код системы см. Приложение А.

Основные функции и обработчики:

1) logEvent(String message) — вывод с отметкой времени NTP.
2) turnLedOn(), turnLedOff() — включение/выключение светодиода с логи-рованием.
3) HTTP-обработчики handleRoot(), handleTurnOn(), handleTurnOff(), handleToggleMode() для управления через веб-интерфейс.
4) setupWebServer() — инициализация маршрутов и запуск сервера.
5) setup():

– Инициализация Serial — Serial.begin(115200).

– Настройка пинов PIR и LED (pinMode).

– Подключение к Wi-Fi (WiFi.begin, ожидание подключения).

– Запуск NTP-клиента (timeClient.begin() и timeClient.update()).

– Запуск веб-сервера.

6) loop():

а) 1. server.handleClient() и timeClient.update() для обслуживания HTTP и об-новления времени.

б) 2. В режиме AUTO:

– analogRead(LDR_PIN) — чтение освещенности.

– digitalRead(PIR_PIN) — чтение PIR (LOW — движение).

– Логирование изменений уровня света и движения при значительной разнице.

Логика:

1) Если темно (уровень > порога) и есть движение — включить LED и запомнить время.

2) Если просто темно — включить LED.

3) Если время от последнего движения более MOTION_DURATION — выключить.

4) Если светло (уровень <= порога) — выключить.

в) 3. При manualMode все автоматические функции отключены, и свет управ-ляется через веб.

г) 4. Задержка delay(100) для стабилизации.

Параметры для настройки скетча:

– SSID и пароль Wi-Fi

– Пороги LIGHT_THRESHOLD и MOTION_DURATION

– Пины LDR_PIN, PIR_PIN и LED_PIN

– Часовой пояс и сервер NTP

– Настройки веб-интерфейса (порт, заголовки HTML)

1.3.5 Реализация веб-интерфейса

Для управления системой освещения разработан веб интерфейс на базе фреймворка Django.

Структура веб-сервиса:

а) Маршрутизация (parks/urls.py):

– Путь / ведет на главную страницу управления.

– Дополнительные эндпоинты /on/, /off/, /toggle/ — для отправки команд включения/выключения и переключения режимов.

б) Представления (parks/views.py):

– index(request) — формирует контекст с текущим состоянием систе-мы (авто/ручной режим, статус светильника), передает его в шаб-лон.

– turn_on(request), turn_off(request) — обрабатывают GET запросы для активации команд и перенаправляют обратно на главную.

– Логика взаимодействия с ESP устройствами реализована через HTTP запросы к их локальным IP-адресам.

в) Шаблоны (parks/templates/parks/index.html):

– HTML страница на основе Bootstrap для адаптивного отображения на мобильных и десктопных устройствах.

– Отображение текущего режима работы.

– Кнопки для ручного управления: «Включить», «Выключить», «Переключить режим».

г) Статические файлы (parks/static/parks/):

– CSS стили для визуального оформления.

– JS скрипт на основе AJAX (fetch API) для асинхронной отправки команд и опроса статуса без перезагрузки страницы.

д) Интеграция с NTP:

– При загрузке страницы время синхронизируется через клиентскую библиотеку JavaScript с тем же NTP сервером, что в прошивке ESP.

– Безопасность и доступ:

– Интерфейс доступен только в локальной сети.

1.3.6 Тестирование

В тестирование входили такие процессы как:

1) Проверка корректности работы датчиков в разных режимах освеще-ния.
2) Проверка стабильности беспроводной связи.
3) Мониторинг работы прототипа.
4) Корректировка параметров алгоритмов диммирования.

1.4 Основные этапы/итерации реализации проекта

1.4.1 Блок описания решенных задач проекта

В ходе выполнение проекта были решены следующие задачи:

5) Анализ и постановка задачи
6) Выбор аппаратной платформы
7) Проектирование системы
8) Разработка ПО системы
9) Реализация веб-интерфейса
10) Тестирование
11) Документация

1.4.2 Результаты проекта с указанием вклада каждого студента

1) К. В. Колмаков:

Организовывал работу команды, следил за прогрессом проекта и занимался документацией

2) И. Е. Коростелев:

Проектировал систему и разрабатывал ПО системы

3) И. Д. Демин:

Проектировал систему и разрабатывал веб-интерфейс

4) Е. С. Новиков:

Подбирал компоненты системы и тестировал систему для выявле-ния её недочётов

5) И. Е. Петров:

Разрабатывал веб-интерфейс и его взаимодействие с системой

1.4.3 Описание ролей в проекте, схемы взаимодействия с руководителем и заказчиком

- Колмаков К.В. – тимлид, руководил работой команды
- Коростелев И.Е. – разработчик устройства, был ответственен за проек-тирование системы и разработку ПО системы
- Демин И.Д. – разработчик веб-интерфейса/проектировщик устройства, был ответственен за проектирование системы и разработку веб-сервиса
- Новиков Е. С. – аналитик-тестировщик, подбирал компоненты системы и тестировал работоспособность системы
- Петров И. Е. – разработчик веб интерфейса, реализовывал работу веб-интерфейса и его взаимодействие с системой

Связь с руководителем проекта шла через тимлида проекта, который, в свою очередь, выдавал задания другим участникам команды.

ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта была разработана автономная система освещения парковых зон на основе микроконтроллера ESP8266. Система успешно реализует адаптивное освещение на основе данных с датчиков освещённости и движения, что позволяет повысить энергоэффективность и комфорт использования общественных пространств в тёмное время суток.

Программная часть проекта реализована с применением среды Arduino IDE. Основной функционал включает автоматическое и ручное управление светильниками, веб-интерфейс для дистанционного переключения режимов, а также ведение журнала событий с синхронизацией. Это позволяет контролировать и управлять системой без необходимости физического доступа к устройству.

Проект может быть масштабирован на большее количество осветительных точек и его возможно адаптировать к различным условиям и требованиям городской инфраструктуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. NTP Working Group. RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification. URL: https://tools.ietf.org/html/rfc5905 (дата доступа: 14.05.2025).

2. Espressif Systems. ESP8266EX Datasheet. URL: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf (дата доступа: 14.05.2025).

3. Arduino. Arduino Uno Rev3 Manual. URL: https://store.arduino.cc/arduino-uno-rev3 (дата доступа: 14.05.2025).

4. Schréder. SmartPARK – Solar-powered LED lighting solution. URL: https://www.schreder.com/products/smartpark (дата доступа: 14.05.2025).

5. Signify. Global. URL: https://www.signify.com/global/ (дата доступа: 14.05.2025).

ПРИЛОЖЕНИЕ А

Исходный код модуля ESP8266

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <WiFiUdp.h>
#include <NTPClient.h>

// Настройки WiFi
const char* ssid = "***";
const char* password = "***";

const int LDR_PIN = A0;
const int PIR_PIN = D5;

const int LIGHT_THRESHOLD_LOW = 200;
const int LIGHT_THRESHOLD_MID = 500;
const unsigned long MOTION_TIMEOUT = 5000; // 5 секунд

const int BRIGHT_OFF = 450;
const int BRIGHT_MID = 255;
const int BRIGHT_HIGH = 10;

const float SMOOTHING_FACTOR = 0.2;
int smoothedLightLevel = 0;

bool motionDetected = false;
bool manualMode = false;
unsigned long motionStartTime = 0;
int currentBrightness = BRIGHT_OFF;
int lastLightLevel = -1;

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 3 * 3600, 60000);
ESP8266WebServer server(80);
```

```
void logEvent(String message) {
Serial.println("[" + timeClient.getFormattedTime() + "] " + mes-
sage);
}

int getSmoothedLightLevel() {
int rawValue = analogRead(LDR_PIN);
smoothedLightLevel = SMOOTHING_FACTOR * rawValue + (1 - SMOOTH-
ING_FACTOR) * smoothedLightLevel;
return smoothedLightLevel;
}

void setBrightness(int level) {
if(currentBrightness != level) {
Serial.println(level);
currentBrightness = level;
}
}

void updateLighting() {
int light = getSmoothedLightLevel();

if (light <= LIGHT_THRESHOLD_LOW) {
setBrightness(BRIGHT_OFF);
}
else if (light <= LIGHT_THRESHOLD_MID) {
setBrightness(BRIGHT_MID);
}
else {
setBrightness(BRIGHT_HIGH);
}
}

void handleRoot() {
```

```
String html = "<html><head><meta charset='UTF-8'><ti-
tle>Управление светом</title></head><body>";
html += "<h2>Режим: " + String(manualMode ? "Ручной" : "Авто") +
"</h2>";
html += "<p>Освещенность: " + String(smoothedLightLevel) +
"</p>";
html += "<p>Текущая яркость: " + String(currentBrightness) +
"</p>";
html += "<p>Состояние: " + String(motionDetected ? "Движение" :
"Нет движения") + "</p>";
html += "<form action='/toggle_mode'><button>Переключить
режим</button></form>
";
html += "<form action='/on'><button>Включить максимум</but-
ton></form>";
html += "<form action='/off'><button>Выключить</button></form>";
html += "</body></html>";
server.send(200, "text/html", html);
}

void setup() {
Serial.begin(9600);
pinMode(PIR_PIN, INPUT_PULLUP);

smoothedLightLevel = analogRead(LDR_PIN);

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("\nConnected. IP: " + WiFi.localIP().toString());

timeClient.begin();
server.on("/", handleRoot);
```

```
server.on("/on", []() { if (manualMode) setBright-
ness(BRIGHT_HIGH); server.sendHeader("Location", "/");
server.send(302); });
server.on("/off", []() { if (manualMode) setBright-
ness(BRIGHT_OFF); server.sendHeader("Location", "/");
server.send(302); });
server.on("/toggle_mode", []() { manualMode = !manualMode;
server.sendHeader("Location", "/"); server.send(302); });
server.begin();
}

void loop() {
server.handleClient();
timeClient.update();

if (!manualMode) {
int light = getSmoothedLightLevel();
bool motion = (digitalRead(PIR_PIN) == LOW);

if (motion) {
setBrightness(BRIGHT_HIGH);
motionStartTime = millis();
motionDetected = true;
}
else if (motionDetected && (millis() - motionStartTime >= MO-
TION_TIMEOUT)) {
motionDetected = false;
updateLighting();
}
else if (!motionDetected) {
updateLighting();
}
}
```

```
delay(100);
}
```