МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д. В. Топольский
«___»_____ 2022 г.

# ЦИФРОВИЗАЦИЯ СИСТЕМЫ «УМНОГО» ДОМАШНЕГО ОСВЕЩЕНИЯ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-09.03.01.2022.308-233.ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____Ю. Г. Плаксина
«___»_____2022 г.

Автор работы,
студент группы КЭ-406
_____И. А. Зыков
«___»_____2022 г.

Нормоконтролёр,
к.п.н., доцент каф. ЭВМ
_____ М. А. Алтухова
«___»_____2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д. В. Топольский
«___»_____2022 г.

# ЗАДАНИЕ

**на выпускную квалификационную работу бакалавра**

студенту группы КЭ-406
Зыкову Ивану Андреевичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Цифровизация системы «умного» домашнего освещения» утверждена приказом по университету от 12 декабря 2021 г. №308/141

2. **Срок сдачи студентом законченной работы:** 1 июня 2022 г.

3. **Исходные данные к работе**

3.1 Напряжение: 220-240 В.

3.2 Частота тока: 50-60 Гц.

3.3 Тип цоколя лапочек: e27.

3.4 Количество источников света: 2.

3.5 Тип беспроводного подключения: Wi-Fi и BLE 4.2.

3.6 Пылевлагозащита: IP20.

3.7 Температура эксплуатации: 0 °С до +40 °С.

4. **Перечень подлежащих разработке вопросов:**

– рассмотрение существующих аналогов интеллектуального управления освещения;

– установка требований к системе;

– проектирование архитектуры разрабатываемого устройства;

– выбор микроконтроллера и сопутствующих модулей;

– разработка модели создаваемого устройства;

– разработка алгоритмов автоматического управления;

– оценка работоспособности программно-аппаратного комплекса.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы _____/*Ю. Г. Плаксина*/

Студент _____/*И. А. Зыков* /

## КАЛЕНДАРНЫЙ ПЛАН

| Этап | Срок сдачи | Подпись руководителя |
|---|---|---|
| Введение и обзор литературы | 04.04.2022 | |
| Разработка модели, проектирование | 24.04.2022 | |
| Реализация системы | 20.05.2022 | |
| Тестирование и отладка | 23.05.2022 | |
| Компоновка текста работы и сдача на нормоконтроль | 26.05.2022 | |
| Подготовка презентации и доклада | 31.05.2022 | |

Руководитель работы _____/*Ю. Г. Плаксина*/

Студент _____/*И. А. Зыков*/

# АННОТАЦИЯ

И. А. Зыков. Цифровизация системы «умного» домашнего освещения. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2022, 55 с., 18 ил., библиогр. список – 29 наим.

В рамках выпускной квалификационной работы производится исследование представленных решений в области автоматического управления домашнего освещения. Анализируются доступные технологические решения и выбираются подходящие по определенным критериям. Прорабатывается алгоритм управления. В итоге создается модель всей системы на макетной плате, программируется и тестируется.

## СОДЕРЖАНИЕ

## ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API (Application Programming Interface – «программный интерфейс приложения») – описание способов (наборов классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой [1]

CAD (Computer-Aided Design – «компьютерная поддержка проектирования») – это система автоматизированного проектирования, предназначенная для выполнения проектных работ с применением компьютерной технике, а также позволяющая создавать конструкторскую и технологическую документацию на отдельные изделия, здания и сооружения [2]

ECDH (Elliptic Curve Diffie–Hellman – протокол Диффри-Хеллмана на эллиптических кривых) – криптографический протокол, позволяющий двум сторонам, имеющим пары открытый/закрытый ключ на эллиптических кривых, получить общий секретный ключ, используя незащищённый от прослушивания канал связи [3]

ECDSA (Elliptic Curves Digital Signature Algorithm – алгоритм построения цифровой подписи с использованием эллиптических кривых) – алгоритм с открытым ключом для создания цифровой подписи, аналогичный DSA (Digital Signature Algorithm – «алгоритм цифровой подписи»), но определенный, в отличие от него, не над конечным числовым полем, а в группе точек эллиптической кривой [4]

GMT (Greenwich Mean Time – «среднее время по Гринвичу») – одно из общеизвестных названий для UTC+0 часового пояса, который на 0 часов впереди UTC [5]

GND (Ground – «земля») – точка нулевого потенциала микросхемы [6]

HTTP (HyperText Transfer Protocol – «протокол передачи гипертекста») – прикладной протокол для передачи гипертекстовых документов [7]

IDE (Integrated Development Environment – «интегрированная среда разработки») – это интегрированная, единая среда разработки, которая используется разработчиками для создания различного программного обеспечения [8]

IoT (Internet of Things – «интернет вещей») – это множество физических объектов, подключенных к интернету и обменивающихся данными [9]

JSON (JavaScript Object Notation – «обозначение объектов JavaScript») – текстовый формат обмена данными, основанный на JavaScript [10]

PSM (Pulse-Skip Modulation – «модуляция с пропуском импульсов») – модуляция, при которой импульсы, передаваемы на переключатель, блокируются или разблокируются при превышении выходным напряжением заданного значения [11]

REST (Representational State Transfer – «передача состояния представления») – способ создания API с помощью протокола HTTP [12]

SDK (Software Development Lit – «комплект для разработки программного обеспечения») – набор средств для разработки программного обеспечения под определенную платформу [13]

UTC (Universal Time Coordinated – «всемирное координирование времени») – стандарт, по которому общество регулирует часы и время [14]

VCC (Voltage Common Collector – «коллектор напряжения») – плюс питания относительно GND [6]

Z-C (Zero-Cross – «пересечение нуля») – пин управления нулевой фазой переменного тока, используется для инициирования сигнала прерывания [15]

# ВВЕДЕНИЕ

В течение последних нескольких лет наблюдается тенденция за активной заботой о своем физическом здоровье. Производители смартфонов и некоторых ноутбуков поддерживают эту тенденцию и предлагают функцию по изменению температуры света экрана. Ближе к ночи цвет экрана становится более желтым (теплым), тем самым снижается напряжение глаза и помогает поддерживать циркадный ритм, а ближе к полудню цвет, наоборот, становится более синим (холодным).

Однако освещение в квартирах по-прежнему остается монотонным в плане цветовой температуры. Пандемия вынудила людей переходить на удаленную работу, из-за чего стали проводить больше времени в наших квартирах при монотонном освещении. Причем у некоторых может быть установлены источники с холодным светом, что не благоприятно сказывается на поддержании циркадного ритма.

Различные компании предлагают свои решения для умного дома, которые включают интеллектуальные системы освещения, способные менять цвет освещения, а также цветовую температуру, однако они не лишены недостатков.

Именно решение ряда основных недостатков предлагаемых решений и обусловлена актуальность впускной квалификационной работы.

Выпускная квалификационная работа посвящена разработке системы по управлению домашнего освещения, а именно будет разработана аппаратная часть и реализована программа по управлению.

Задачи работы:

– найти существующие аналоги и схожие по функционалу системы;
– провести анализ с целью выявления недостатка данных систем;
– составить техническое задание и эскизный проект системы;

– разработать аппаратную платформу, а именно часть, где будет выполнятся программная составляющая;

– выполнить программную реализацию проекта;

– провести тестирование программы управления на готовой аппаратной платформе;

– определить методы тестирования и провести тестирование полного комплекса системы.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Обзор аналогов

На рынке представлены различные системы для управления освещения, в частности имеющие возможность регулировки цветовой температуры освещения. Предлагаемые решения имеют как преимущества, так и недостатки. Разберем более подробно предлагаемые решения и выявим их достоинства и недостатки.

Ассортимент компании WiZ достаточно обширен, предлагаются, как и готовые решения светильников различного форм-фактора и назначения, так и разные умные лампы для использования в уже имеющихся светильниках. В предлагаемых источниках имеется как полноцветная регулировка света, так и просто регулировка белого света, то есть цветовой температуры света [16]. Часть ассортимента представлена на рисунке 1.

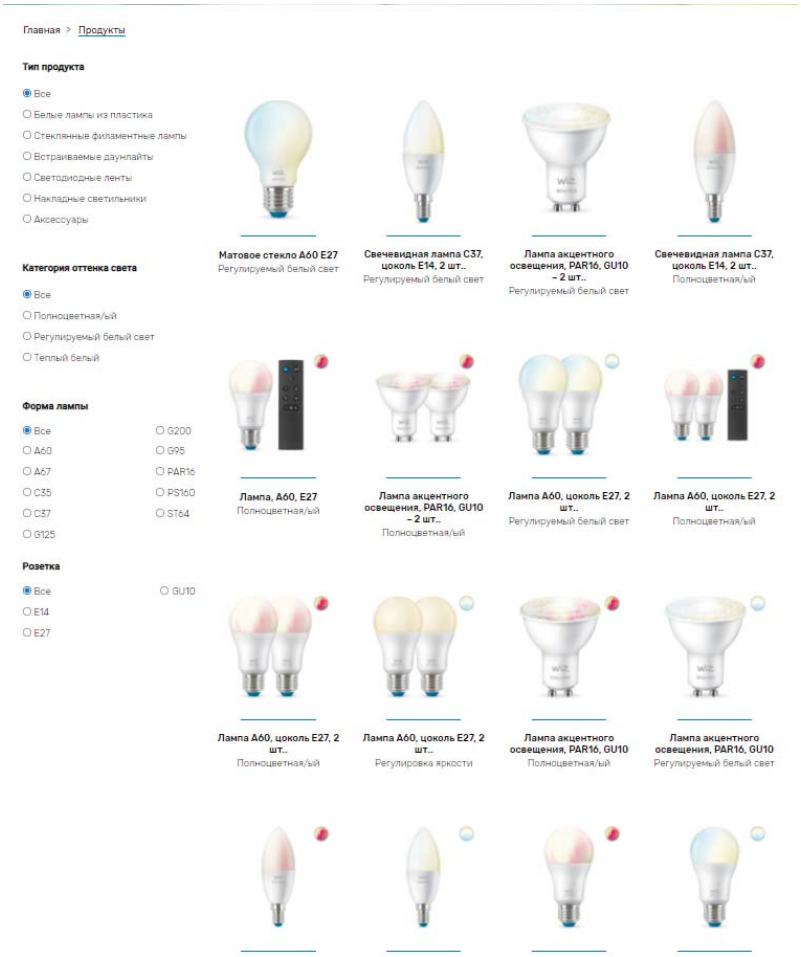


Рисунок 1 – Ассортимент товаров компании WiZ

Достоинства:

– представлены как готовые светильники, так и отдельно умные лампы;

– имеется возможность полноцветной регулировки и белого света;

– заявлена поддержка голосовых ассистентов;

– возможно задать время, в пределах которого будет осуществляться автоматическая регулировка цветовой температуры для поддержки циркадного ритма.

Недостатки:

– не заявлена возможность замены источника света в готовых решениях – светильниках;

– для умных ламп не заявлена возможность создания группы, то есть одновременного управления несколькими лампами;

– не заявлена функция автоматического изменения цветовой температуры в зависимости от времени восхода/заката.

Компания Xiaomi известна своим широким ассортимент умных устройств, поэтому компания также предлагает свои светильники и лампы, которые имеют возможность регулировки белого света. Для управления таким большим количеством умных устройств компания предлагает собственную системы умного дома Xiaomi Mi Home [17]. Некоторые товары представлены на рисунке 2.

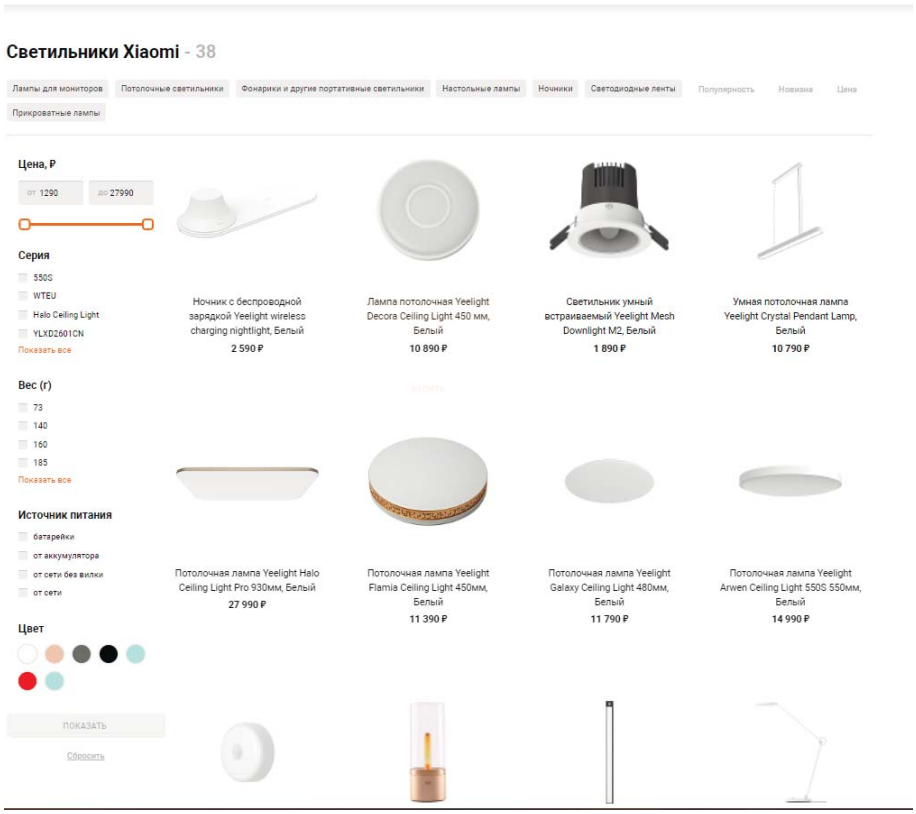


Рисунок 2 – Ассортимент товаров компании Xiaomi

Достоинства:

– представлены как готовые светильники, так и отдельно умные лампы;

– имеется возможность полноцветной регулировки и белого света;

– заявлена возможность объединения ламп в группы;

– имеется поддержка голосовых ассистентов.

Недостатки:

– не заявлена возможность замены источника света в готовых решениях – светильниках;

– не заявлена функция автоматического изменения цветовой температуры.

Philips Hue, как и компания WiZ, входит в состав компании Signify – много национальной светотехнической корпорации. Среди представленных товаров так же имеется большой выбор лам и различных светильником имеющих технологию регулировки цветовой температуры [18]. Некоторые представленные ассортимент показан на рисунке 3.

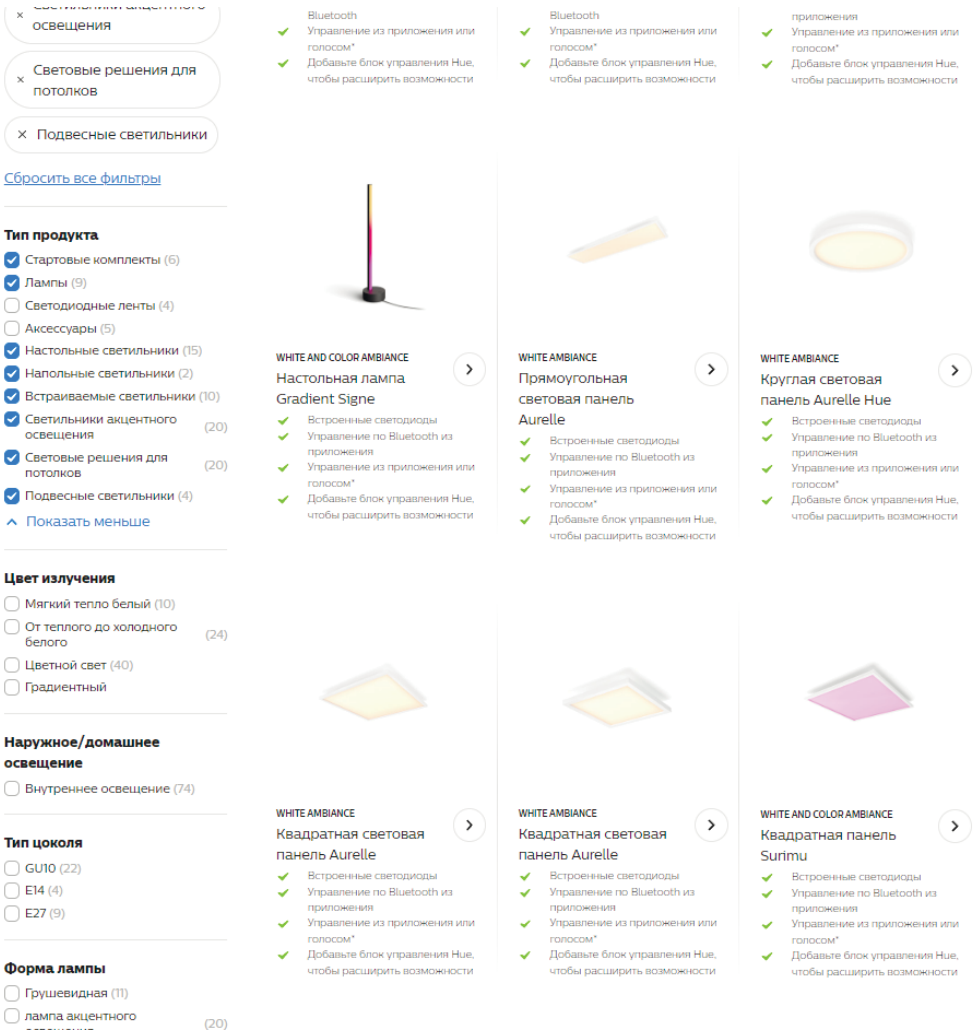


Рисунок 3 – Ассортимент товаров компании Philips Hue

Достоинства:

– представлены как готовые светильники, так и отдельно умные лампы;

– имеется возможность полноцветной регулировки и белого света;

– заявлена поддержка голосовых ассистентов.

– заявлена возможность объединения ламп в группы;

– возможно задать время, в пределах которого будет осуществляться автоматическая регулировка цветовой температуры для поддержки циркадного ритма.

Недостатки:

– не заявлена возможность замены источника света в готовых решениях – светильниках;

– не заявлена функция автоматического изменения цветовой температуры в зависимости от времени восхода/заката.

Сравнительный анализ приведен в таблицах 1 и 2.

Таблица 1 – Сравнительный анализ существующих готовых светильников

| **Производитель** | **Наличие функционала** | | | | |
|---|---|---|---|---|---|
| | Регулирование белого света | Автоматическое регулирование белого света | Автоматическое регулирование белого света в зависимости от восхода/заката | Возможность замены источника света | Поддержка голосовых ассистентов |
| WiZ | Да | Да | Нет | Нет | Да |
| Xiaomi | Да | Нет | Нет | Нет | Да |
| Philips Hue | Да | Да | Нет | Нет | Да |

Таблица 2 – Сравнительный анализ существующих умных ламп

| **Производитель** | **Наличие функционала** | | | | |
|---|---|---|---|---|---|
| | Регулирование белого света | Автоматическое регулирование белого света | Автоматическое регулирование белого света в зависимости от восхода/заката | Создание группы | Поддержка голосовых ассистентов |
| WiZ | Да | Да | Нет | Да | Да |
| Xiaomi | Да | Нет | Нет | Да | Да |
| Philips Hue | Да | Да | Нет | Да | Да |

### 1.1.1 Вывод по обзору аналогов

Разрабатываемая система должна предлагать собой светильник, используемый в качестве основного источника света домашнего освещения. Система должна иметь возможность регулировки температуры автоматически, в зависимости от времени суток. Источник света должен быть заменяемым, с учетом основных требований, этого можно достичь, используя две лампы различной цветовой температуры, корректируя яркость которых можно достичь различной температуры света, к сожалению, простые светодиодные лампы в

данном случае непригодны, так как не позволяют регулировать яркость освещения. Регулировка яркости возможности только в определенных светодиодных лампах с возможностью диммирования, то есть изменения яркости.

## 1.2 Анализ основных технологических решений

### 1.2.1 Выбор аппаратной платформы

Основой системы является микроконтроллер, именно на нем будет происходить вычисление и управление всей системой. Для удобства проектирования принято решение использовать готовые платы с уже установленными микроконтроллерами, а также сопутствующей обвязкой для их программирования и питания. Для разработки можно использовать следующие платы: Arduino, Iskra и STM32 Discovery/Nucleo. Далее рассмотрим их подробно.

Arduino – это платформа, пользующаяся популярностью во всем мире благодаря удобству и простоте языка программирования, а также открытой архитектуре и программному коду. Устройства на базе Arduino могут получать информацию об окружающей среде посредством различных датчиков, а также могут управлять различными исполнительными устройствами [19].

Микроконтроллер на плате программируется при помощи языка Arduino и среды разработки Arduino. Проекты устройств, основанные на Arduino, могут работать самостоятельно, либо же взаимодействовать с программным обеспечением на компьютере. Платы могут быть собраны пользователем самостоятельно или куплены в сборе. Программное обеспечение доступно для бесплатного скачивания. Исходные чертежи схем (файлы CAD) являются общедоступными, пользователи могут применять их по своему усмотрению [19].

Основные преимущества платформы [20]:

– низкая стоимость;

– кроссплатформенность;

– простая и понятная среда проектирования;

– удобное подключение к компьютеру для программирования и питания.

Недостатки [20]:

– отсутствует простой способ регулировки тактовой частоты;

– обладает скромными возможностями для реализации сложных проектов.

Наиболее популярными платами Arduino являются:

1) Uno;

2) Nano;

3) Mini;

4) Mega 2560.

Более подробно с основными техническими характеристиками данных плат можно ознакомиться в работе [20].

Iskra – собственный бренд компании «Амперка», являющийся альтернативой платам Arduino [21].

На данный момент выпускаются следующие серии микроконтроллеров:

– Iskra Neo – аналог Arduino Leonardo;

– Iskra Nano Pro – аналог Arduino Nano с более продвинутым микроконтроллером;

– Iskra Mega – аналог Arduino Mega 2560, имеющий более мощный регулятор, выдающий суммарный ток 3 А;

– Iskra Mini – прямой аналог Arduino Mini с эквивалентными характеристиками;

– Iskra JS / JS Mini – контроллеры с поддержкой JavaScript на платформе Espruino.

STM32 Discovery – это дешевое и комплексное решение для оценки возможностей микроконтроллеров и микропроцессоров STM32. Платы содержат необходимую инфраструктуру для демонстрации конкретных характеристик устройств. Разъемы расширения обеспечивают доступ к большинству входов/выходов устройства и делают возможным подключение дополнительного оборудования [22]. Плата содержит в своем составе микроконтроллер семейства STM32, программатор-отладчик ST-Link/V2 и периферийные устройства, набор которых зависит от конкретной платы [23]

STM32 Nucleo отладочные платы этой серии предназначены для прототипирования устройстве на базе микроконтроллеров семейства STM32. Платы STM32 Nucleo могут быть легко расширены при помощи большого количества специализированных аппаратных надстроек (Nucleo-64 включает разъемы Arduino Uno rev3 и ST morpho, Nucleo-32 включает разъемы Arduino Nano). В платы STM32 Nucleo встроен отладчик/программатор ST-Link/V2. Все пользователи STM32 Nucleo имеют бесплатный доступ к онлайн-ресурсам mbed (компилятор, C/C++ SDK и сообщество разработчиков) [22-23].

Таким образом в качестве основной аппаратной базы была выбрана плата Arduino Nano 33 IoT, так как данная плата в своем составе уже имеет беспроводной модуль связи для обмена данными по Wi-Fi; крипто-чип, интегрирующий протокол безопасности ECDH, позволяющий обеспечить согласование ключей шифрования/дешифрования, наряду с ECDSA для проверки подлинности с подписью для Интернета вещей, включая домашнюю автоматизацию; также платформа Arduino имеет множество библиотек и учебных пособий, что позволяет облегчить процесс разработки.

Для изменения мощности в лампочках, тем самым изменения их яркости был выбран регулятор мощности Arduino Dimmer [15]. Этот модуль Arduino

создан специально для того, чтобы при помощи любого микроконтроллера можно было управлять модулем регулятора.

### 1.2.2 Выбор IDE

Для создания и загрузки программ будет использоваться Arduino IDE – программное обеспечение с открытым исходным кодом, которое можно использовать с любой платой Arduino. Поддерживает языки C и C++ с использованием специальных правил структурирования кода.

### 1.2.3 Выбор API

Для получения информации о времени восхода, заката, а также времени полудня необходимо воспользоваться API сайта погоды либо же специализированного сайта. Далее рассмотрим некоторые.

sunset-sunrise.org – специальный сайт для получения информации о времени восхода и заката, также отображает сопутствующую информацию [24].

Достоинства:

1) API предоставляется на бесплатной основе;
2) нет ограничений на запросы;
3) не нужен ключ;
4) запрос в формате REST;
5) данные возвращаются в формате JSON.

Недостатки:

1) для получения информации необходимо указать координаты места в десятичном формате;
2) время представлено в формате UTC.

Единственное ограничение требуется указание авторства при использовании API [24].

weatherapi.org предоставляет текущие и 14-дневные данные о погоде, а также исторические данные о погоде и геоданные через REST API в формате JSON [25].

Достоинства:

1) предоставляет данные о погоде, астрономические данные, о географическом местоположении;

2) можно настроить какие поля будут возвращены, тем самым можно отключить ненужные поля;

3) имеется бесплатный тарифный план;

4) данные возвращаются в формате JSON;

5) запрос в формате REST;

6) возвращает текущее время в 24-часовом формате;

7) название города можно ввести кириллицей.

Недостатки:

1) для работы с API необходимо зарегистрироваться, чтобы получить ключ;

2) время представлено в формате UTC;

3) не представлена информация о времени полудня.

OpenWeather – это команда экспертов в области информационных технологий и специалистов по обработке и анализу данных, которые занимаются глубокой наукой о погоде. Для каждой точки земного шара OpenWeather предоставляет исторические, текущие и прогнозируемые данные о погоде через API [26].

Достоинства:

1) имеется бесплатный тарифный план;

2) возвращает данные в формате JSON;

3) среди возвращаемых данных есть информация о часовом поясе;

4) возвращает время в формате Unix и UTC;

5) запрос в формате REST;

6) возвращает текущее время.

Недостатки:

1) для получения даже бесплатного тарифа необходимо указать платежные реквизиты, что недоступно в условии санкций;

2) для получения информации необходимо ввести его координаты в десятичной форме.

Яндекс Погода показывает прогноз погоды в разных городах мира. Прогнозы строятся с помощью технологии Meteum. Meteum — это технология для расчета прогноза погоды, разработанная в Яндексе. С помощью алгоритмов машинного обучения Meteum сравнивает данные разных прогностических моделей, а затем строит собственный прогноз [27].

Достоинства:

1) возвращает время в 24-часовом формате;

2) имеется бесплатный тариф.

Недостатки:

1) запрос осуществляется не в формате REST;

2) для получения информации необходимо указывать координаты места.

Таким образом для получения информации о времени восхода и заката будет использоваться API сайта sunrise-sunset.org, а для получения координат weatherapi.com, информация о часовой зоне с которого в дальнейшем будет использоваться для получения информации о текущем времени и часовом поясе (в секундах между местным временем и временем в формате UTC) при помощи REST API worldtimeapi.org.

### 1.3 Вывод по анализу решений

Для решения поставленной задачи были выбраны следующие решения: аппаратные компоненты – плата Arduino Nano 33 IoT, регулятор мощности – AC Dimmer Module; среда разработки – Arduino IDE; API – sunrise-sunset.org, weatherapi.com, worldtimeapi.org.

# 2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

## 2.1 Основные требования к функционалу системы

Перечень функциональных требований к системе:

– система должна подключаться к беспроводной сети Wi-Fi;

– после подключения к беспроводной сети система в автоматическом режиме должна подключаться к API сайтов, которые были описаны в предыдущем пункте, получать данные и записывать их;

– в случае корректного подключения должна включаться лампа цветовой температурой, соответствующей времени суток;

– в случае некорректного подключения должен быть предусмотрен режим по умолчанию;

– в начале каждого дня должно происходит обновление данных о времени восхода и заката.

## 2.2 Нефункциональные требования

Перечень нефункциональных требований к системе в целом:

– для удобства процесса отладки данные о корректном подключении и получаемые данные при помощи API должны выводиться в консоль;

– устройство должно стабильно работать при температурах окружающей среды в пределах от 0 °C до +40 °C.

# 3 ПРОЕКТИРОВАНИЕ

## 3.1 Архитектура предлагаемого решения

Система управления в одном целом корпусе объединяет следующие функциональные модули: микроконтроллер с распаянным на одной плате модулем беспроводным модулем и устройства управления мощностью (диммеры). Помимо этого, предполагается использование блока питания для питания микроконтроллера от бытовой сети.

Одна из основных задач микроконтроллера – получение данных при помощи API и их последующая обработка их хранения. Для быстрой обработки получаемых JSON файлом предусматривается высокая производительность и большое количество встроенной оперативной памяти. Первое получение данных происходит при включении микроконтроллер, для чего достаточно просто задать название города. Получив все данные, устанавливается внутренние часы микроконтроллера для последующего управления освещением. Однако обращение к API сайта занимает некоторое время, поэтому для исключения накопления ошибки разницы во времени и исключения некорректной работы, вызванной расхождением текущего времени и внутреннего времени микроконтроллера, получение данных о времени во время обновления данных о восходе и закате, и в середине дня.

Изменение времени восхода и заката происходит каждый день, поэтому необходимо в начале каждого дня получать данные о текущем времени восхода и заката.

При достижении времени восхода происходит изменение яркости ламп с теплой на нейтрально белую, в случае достижения заката алгоритм обратен, изменяется яркость ламп с нейтрально белой (4000-5000 К) на теплую (2700-3500

К). При включении происходит проверка в какое именно время суток было включено освещение и в соответствии с этим включается необходимая лампа.

### 3.2 Алгоритм решения задачи

Для полноценного функционирования система должна получать данные при помощи API, для чего сперва необходимо подключиться к беспроводной сети Wi-Fi, после подключения проверяется, задан ли город. Если город задан, то идет обращение к api.weatherapi.com по средствам простого HTTP запроса, запрос выглядит следующим образом (пример приведен для города Челябинска): https://api.weatherapi.com/v1/timezone.json?key=0d8b99e61cf64bbda5234139221905&q=Chelyabinsk, где q – это название города латиницей, а key – уникальный ключ, который выдается после регистрации и получения доступа к API. Пример ответа для вышеуказанного запроса представлен в листинге 1.

Листинг 1 – Ответ после http запроса к api.weatherapi.com

```
{
  "location": {
    "name":"Chelyabinsk",
     "region":"Chelyabinsk",
     "country":"Russia",
     "lat":55.15,
     "lon":61.43,
     "tz_id":"Asia/Yekaterinburg",
     "localtime_epoch":1653990818,
     "localtime":"2022-05-31 14:53"
  }
}
```

Интересующие нас поля и их описание указано в таблице 3.

Таблица 3 – Описание используемых полей записи JSON

| **Название** | **Расшифровка** |
|---|---|
| lat | latitude – широта, географическая координата в градусах |
| lon | longitude – долгота, географическая координата в градусах |
| tz_id | Часовой пояс в котором находится указанный город |

После успешного получения информации о часовом поясе следует запрос к API сайта worldtimeapi.org по средствам простого http запроса, пример запроса для часового пояса Asia/Yekaterinburg, в котором находится город Челябинск: https://worldtimeapi.org/api/timezone/Asia/Yekaterinburg

Пример ответа для вышеуказанного запроса представлен в листинге 2.

Листинг 2 – Пример ответа после обращения к worldtimeapi.org


```
{
      "abbreviation":"+05",
      "client_ip":"198.16.76.67",
      "datetime":"2022-05-31T15:31:01.756935+05:00",
      "day_of_week":2,
      "day_of_year":151,
      "dst":false,
      "dst_from":null,
      "dst_offset":0,
      "dst_until":null,
      "raw_offset":18000,
      "timezone":"Asia/Yekaterinburg",
      "unixtime":1653993061,
      "utc_datetime":"2022-05-31T10:31:01.756935+00:00",
      "utc_offset":"+05:00",
      "week_number":22
}
```


Используемые поля, а также их описание указано в таблице 4.

Таблица 4 – Описание используемых полей записи JSON полученных от worldtimeapi.org

| **Название** | **Расшифровка** |
|---|---|
| unixtime | Текущее время в формате Unix Timestamp, без учета часового пояса, то есть после перевода в стандарт ISO 8601 часовой пояс GMT+0000 |
| raw_offset | Текущее смещение по времени относительно среднего времени по Гринвичу, указано в секундах |

Далее следует проверка были ли получены координаты указанного города, если координаты были получены, то идет обращение к API сайта sunrise-sunset.org по при помощи простого HTTP запроса, пример запроса для города Челябинск выглядит следующим образом: https://api.sunrise-sunset.org/json?lat=55.15&lng=61.43, где lat и lng это географические координаты, широта и долгота соответственно. Пример ответа указан в листинге 3.

Листинг 3 – Пример ответа после обращение к api.sunrise-sunset.org

```
{
     "results":{
          "sunrise":"11:20:22 PM",
          "sunset":"4:23:32 PM",
          "solar_noon":"7:51:57 AM",
          "day_length":"17:03:10",
          "civil_twilight_begin":"10:28:37 PM",
          "civil_twilight_end":"5:15:17 PM",
          "nautical_twilight_begin":"8:47:53 PM",
          "nautical_twilight_end":"6:56:01 PM",
          "astronomical_twilight_begin":"12:00:01 AM",
          "astronomical_twilight_end":"12:00:01 AM"
     },
     "status":"OK"
}
```

Поля, которые используются в работе, а также их описание представлено в таблице 5.

Таблица 5 – Описание используемых полей записи JSON, полученной от sunrise-sunset.org

| **Название** | **Расшифровка** |
|---|---|
| sunrise | Время восхода солнца в день формирования запроса для указанных координат в формате UTC |
| sunset | Время заката солнца в день формирования запроса для указанных координат в формате UTC |

На рисунке 4 показан упрощенный общий алгоритм подключения к API сайтов, а на рисунке 5 показан упрощенный общий алгоритм десериализации полученных записей JSON.


Рисунок 4 – Общий алгоритм подключения к API


Рисунок 5 – Общий алгоритм десериализации полученных JSON записей

После того как все данные были получены происходит сравнение текущего времени с временем восхода и заката, если текущее время находится в промежутке между восходом и закатом, то есть в дневное время суток, загорается лампа нейтрально белой цветовой температуры (4000-5000 К), если же время находится в промежутке от заката до восхода, то есть ночное время суток, то включается лампа с теплой цветовой температурой (2700-3500 К). В случае же

если данные не были получены, то по умолчанию включается лампа с теплой температурой.

Как только время становится равным времени восхода или заката происходит смена текущей активной лампы по средствам изменения яркости. Во время восхода сперва увеличивается яркость лампы с нейтрально белым светом, а затем происходит уменьшение яркости лампы теплого цвета, в случае с закатом алгоритм обратен – увеличение яркости теплой лампы, уменьшение яркости нейтрально белого.

Управление яркостью происходит при помощи диммера, к которому подключены лампы. Диммер основан на симисторе – полупроводниковый элемент, предназначенный для коммутации нагрузки в сети переменного тока. Он имеет три вывода: два силовых и управляющий. Для управления режимом работы симистора используется низковольтный сигнал, подаваемый на управляющий электрод симистора. При подаче напряжения на управляющий электрод симистор переходит из закрытого состояния в открытое и пропускает через себя ток [28]. Для избавления помех в сети и неплавного управления нагрузкой (в случае с лампочками – мерцания) в диммере установлена две оптопары, одна предназначена для детекции перехода напряжения через ноль. Оптопара фиксирует переход через ноль и отправляет сигнал на микроконтроллер, на котором срабатывает прерывание. Далее необходима вторая оптопара, при подаче сигнала на которую симистор открывается и начинает проводить ток, время между детекцией перехода нуля и подачей сигнала определяет то, насколько будет обрезана полуволна синусоиды переменного тока, что показано на рисунке 6 [29]. Таким образом происходит фазовое регулирование.

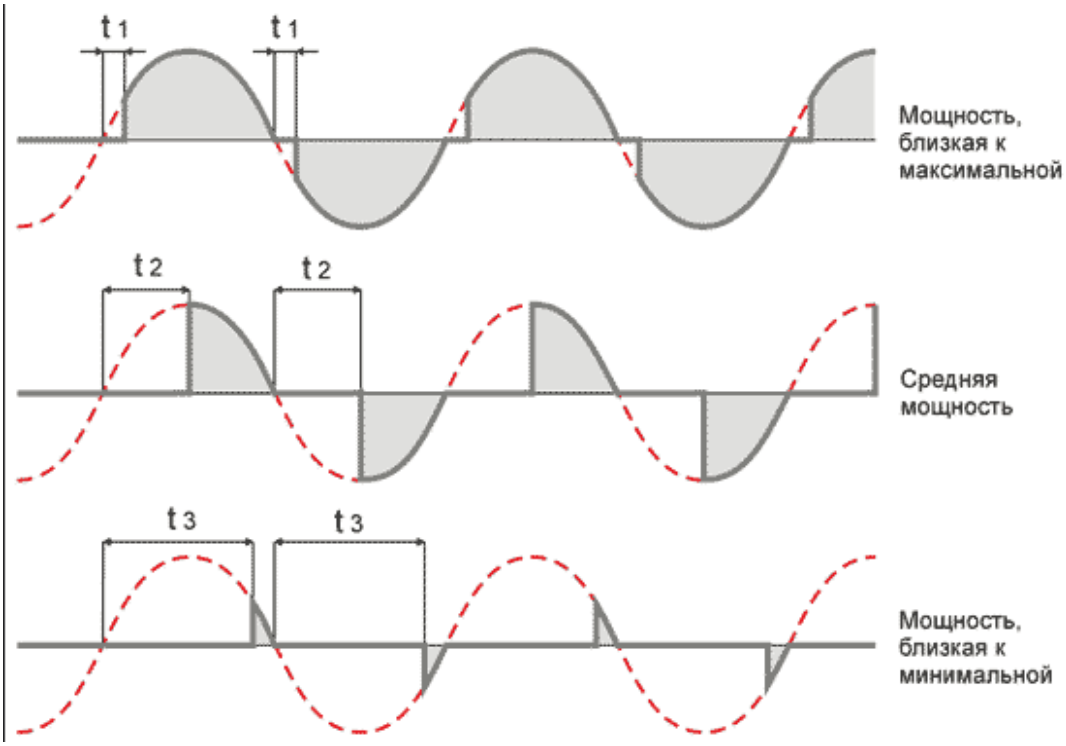


Рисунок 6 – Зависимость амплитуды синусоиды от времени подачи импульса

Для обеспечения плавного регулирования яркостью необходимо динамическое изменение времени, прошедшего с момента детекции нуля и подачи сигнала на симистор.

На рисунках 7 и 8 представлен упрощенный алгоритм работы всей системы в целом.


Рисунок 7 – Упрощенный алгоритм управления всей системы в целом

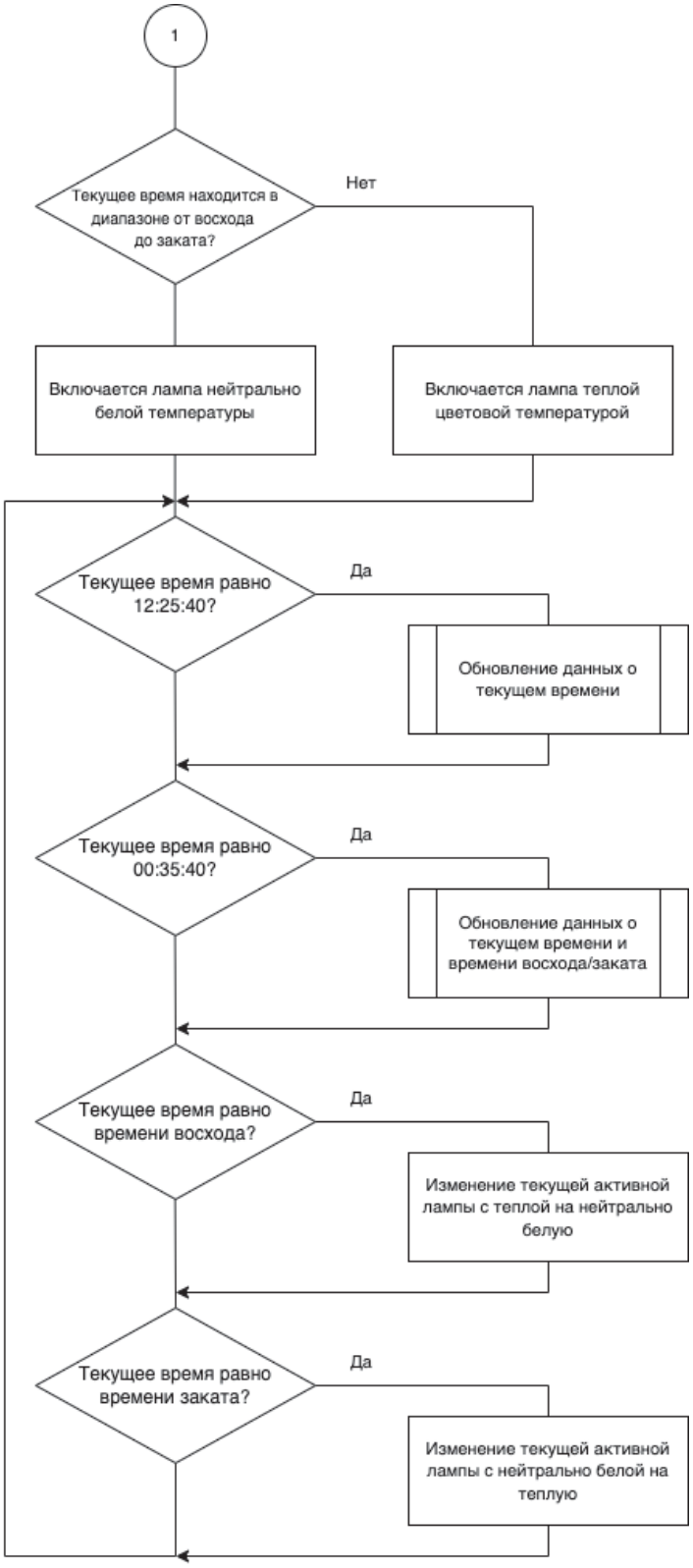


Рисунок 8 – Продолжение алгоритма управления системы

# 4 РЕАЛИЗАЦИЯ

Выполним сборку модели с простыми устройствами, чтобы проверить работоспособность системы, затем запрограммируем его

## 4.1 Сборка модели

Модель была собрана на беспаечной макетной плате, высоковольтные провода соединены при помощи клеммников. Для удобства работы и обеспечения безопасности при работе с высоким напряжением диммеры были подключены к сети при помощи штепсельной вилки, которая вставлена в розетку.

Микроконтроллер, а именно плата Arduino Nano 33 IoT для удобства отладки и финансовых ограничений подключена к компьютеру при помощи кабеля micro-USB, за счет которого также происходит её питание.

Электрическая схемы соединения компонентов прототипа показана на рисунке 9.

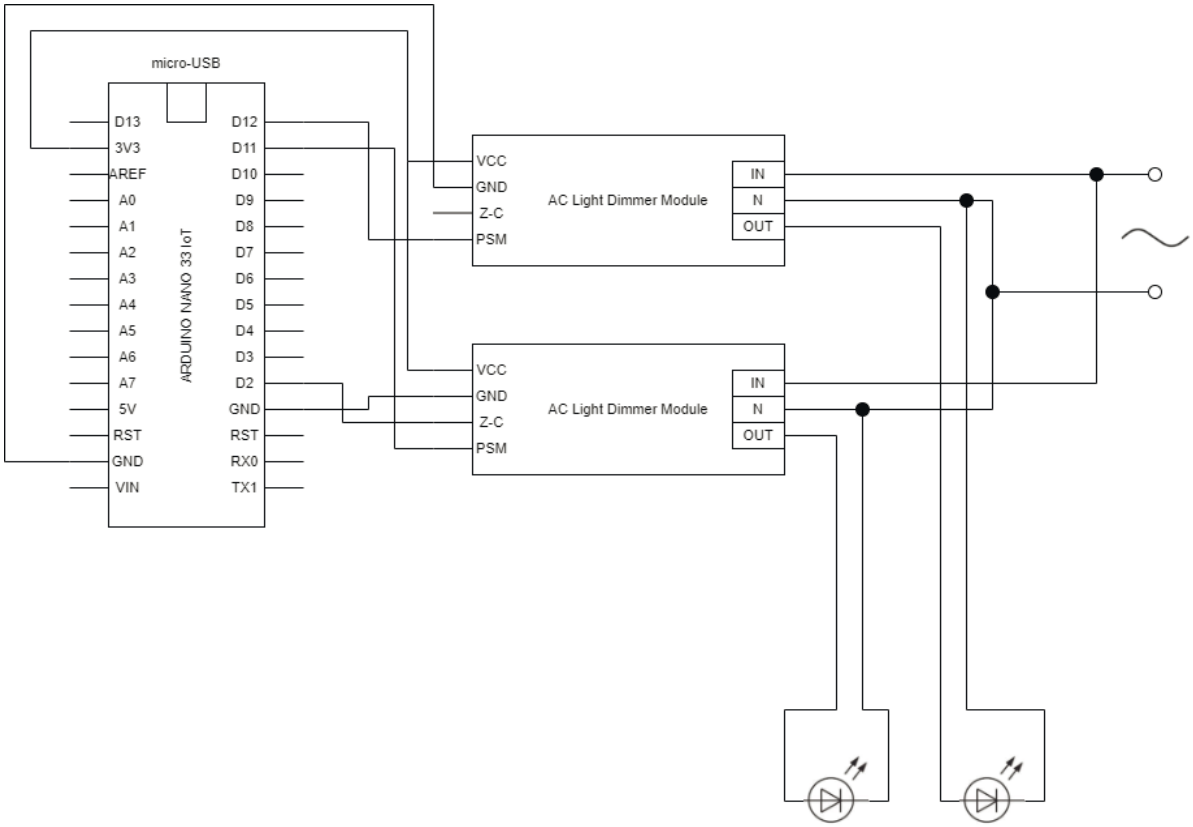


Рисунок 9 – Схема подключения модели

К микроконтроллеру подключены два диммера, их питание осуществляется при помощи пинов VCC подключенных к пину 3V3, который выдает постоянное напряжение 3 В. Пины GND подключены соответственно к пинам GND микроконтроллера. Так как в цепи отсутствует индуктивность или ёмкость, то соответственно на диммерах будет сигнал с одинаковой фазой, а значит переход через ноль будет происходить в одинаковый момент времени, таким образом достаточно фиксировать переход через ноль всего лишь на одном диммере, подключив пин Z-C к пину D2 микроконтроллера. Пины PSM, на которые подается сигнал для открытия симистора подключены к пинам D11 и D12.

На рисунках 10 и 11 представлен собранная модель системы.

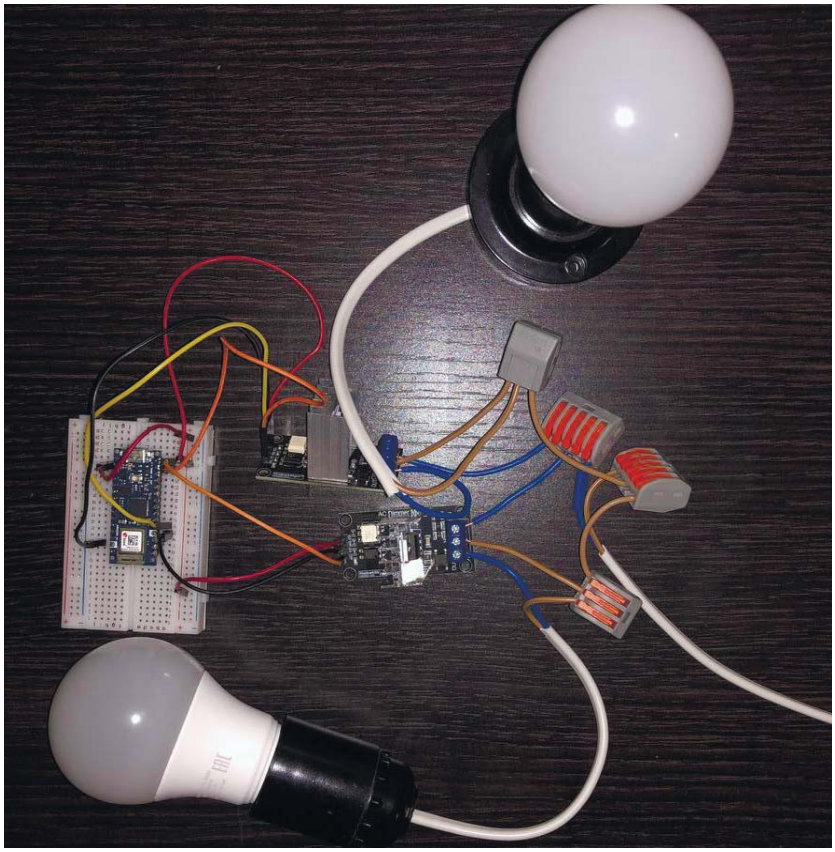

Рисунок 10 – Фотография модели сверху

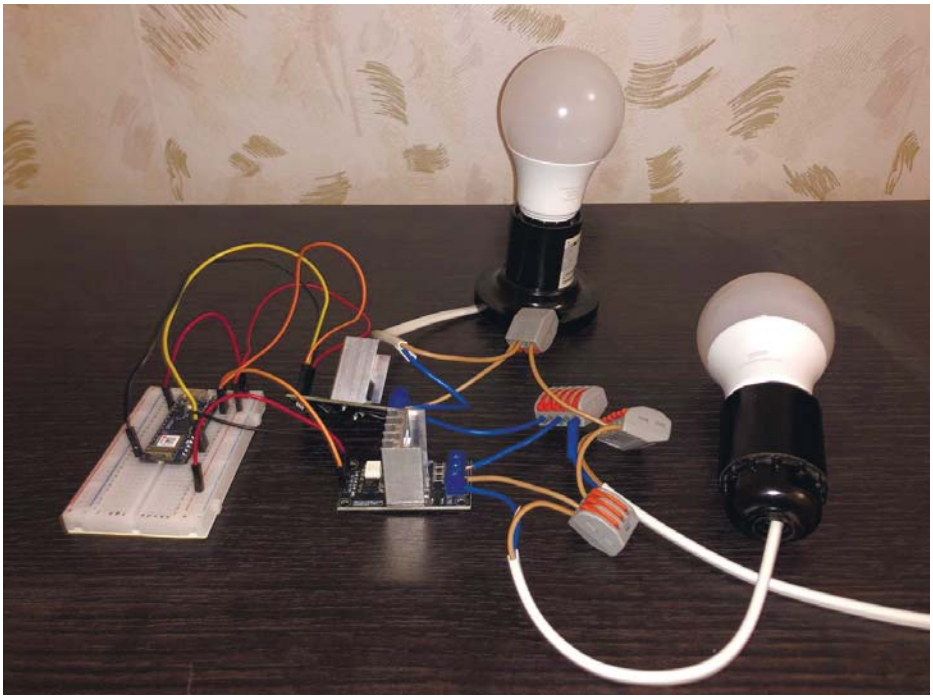

Рисунок 11 – Фотография модели сбоку

## 4.2 Программирование устройства

Написание кода для Arduino NANO 33 IoT осуществлялось в программе Arduino IDE, показана на рисунке 12.

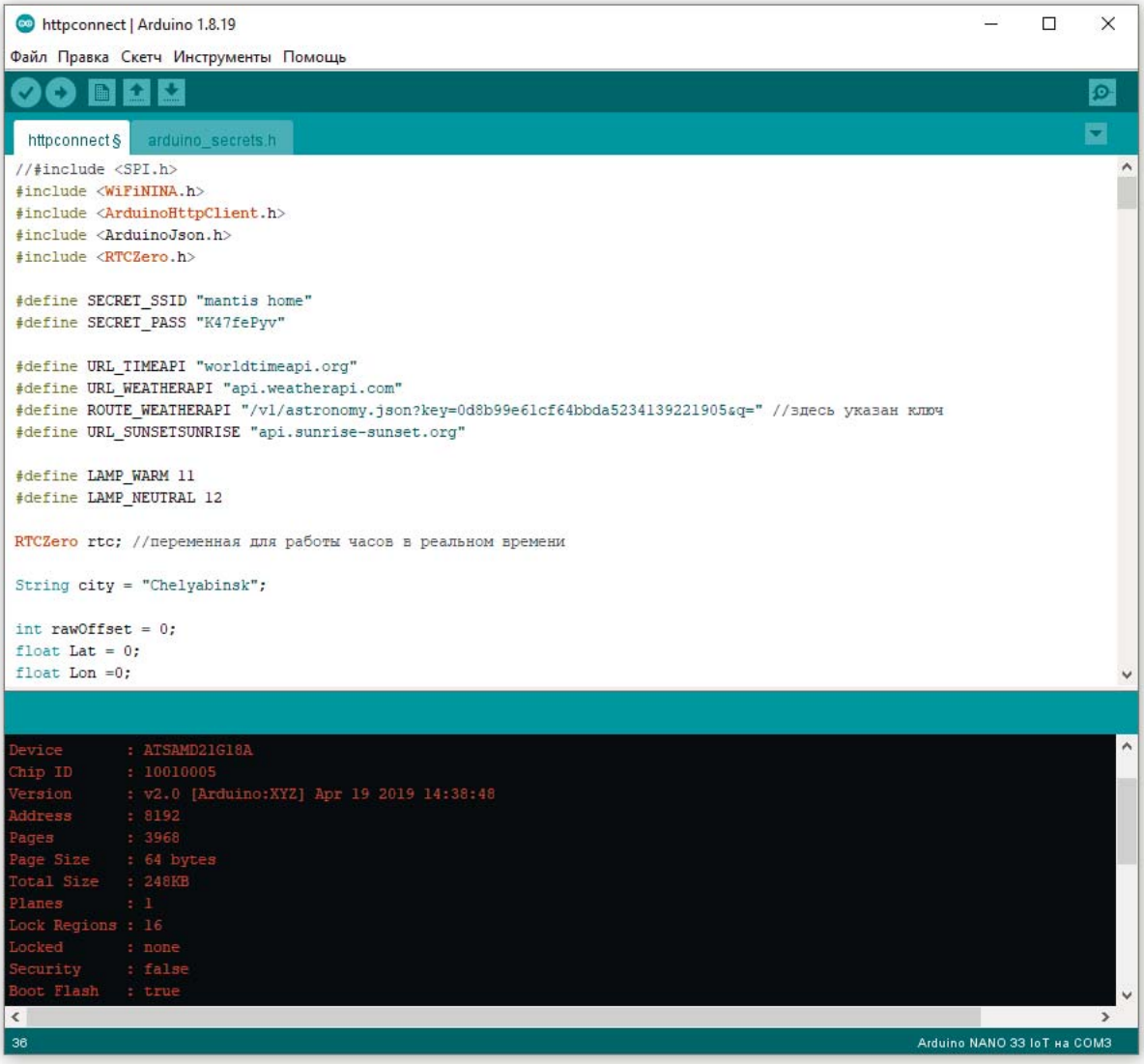


Рисунок 12 – Интерфейс программы Arduino IDE

Для реализации использовались следующие библиотеки: WiFiNINA.h для работы с распаянным на плате модулем Wi-Fi; ArduinoHttpClient.h для создания клиентских запросов; ArduinoJson.h для десериализации JSON структур;

RTCZero.h для работы часов реального времени в микронтроллере. Код представлен в листинге 1 приложения А.

# 5 ТЕСТИРОВАНИЕ

## 5.1 Методология тестирования

В рамках данной работы был реализован прототип системы и не реализовано взаимодействие со специально созданным мобильным приложением, благодаря которому осуществляется подключение к беспроводной сети и задание города. По этой причине возможно проведение альфа-тестирования основного алгоритма программы.

## 5.2 Проведение процедуры тестирования

Для удобства проверки тестирования на ключевых этапах программы был реализован вывод в консоль. В случае успешного подключения выводятся три ответа полученные в результате обращения к API, а также статус код, для отслеживания корректности подключения и другие важные параметры, например, такие как географические координаты, часовой пояс, часовой пояс в виде смещения по времени относительно GMT, время восхода и заката с учетом часового пояса.

Пример вывода в консоль для города Челябинск показан на рисунке 13.


Рисунок 13 – Пример вывода данных в консоль для г. Челябинск

Протестируем подключение и получение данных для других городов: Лондон, показан на рисунке 14, и Москва, показана на рисунке 15.


Рисунок 14 – Пример вывода данных в консоль для г. Лондон

Рисунок 15 – Пример вывода данных в консоль для г. Москва

Далее протестируем корректность включения ламп, включив микроконтроллер в момент времени между восходом и закатом (рисунок 16) и в момент времени между закатом и восходом (рисунок 17). Данные о времени восхода и заката получены для г. Челябинска.

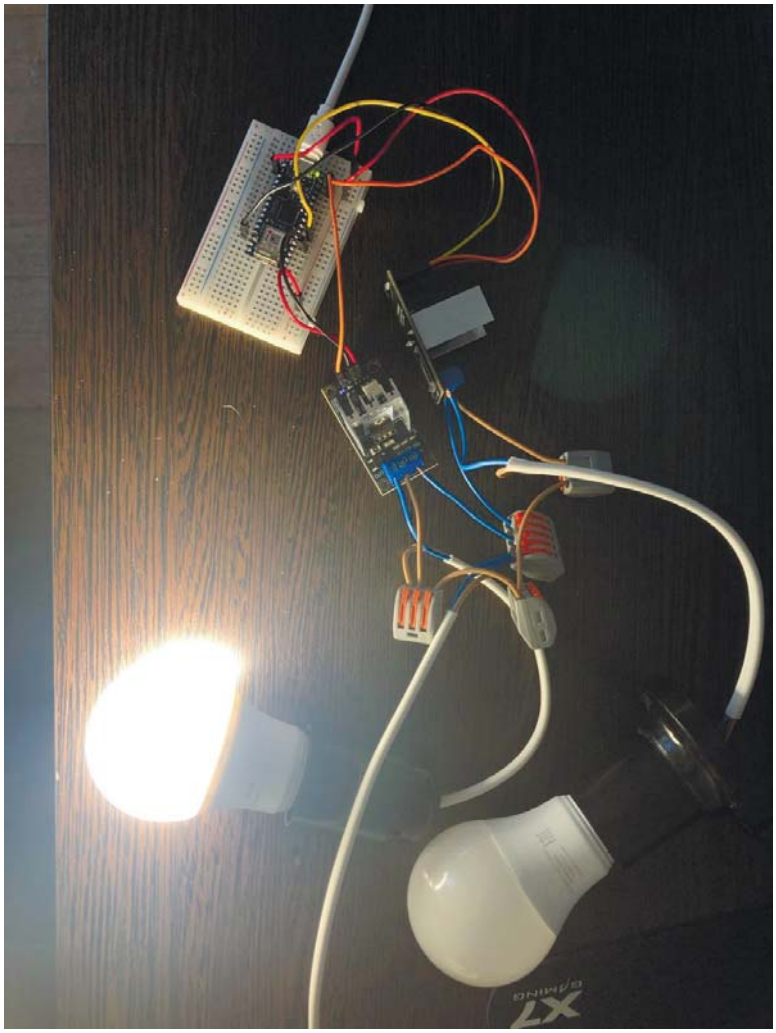

Рисунок 16 – Результат включения в промежутке между восходом и закатом

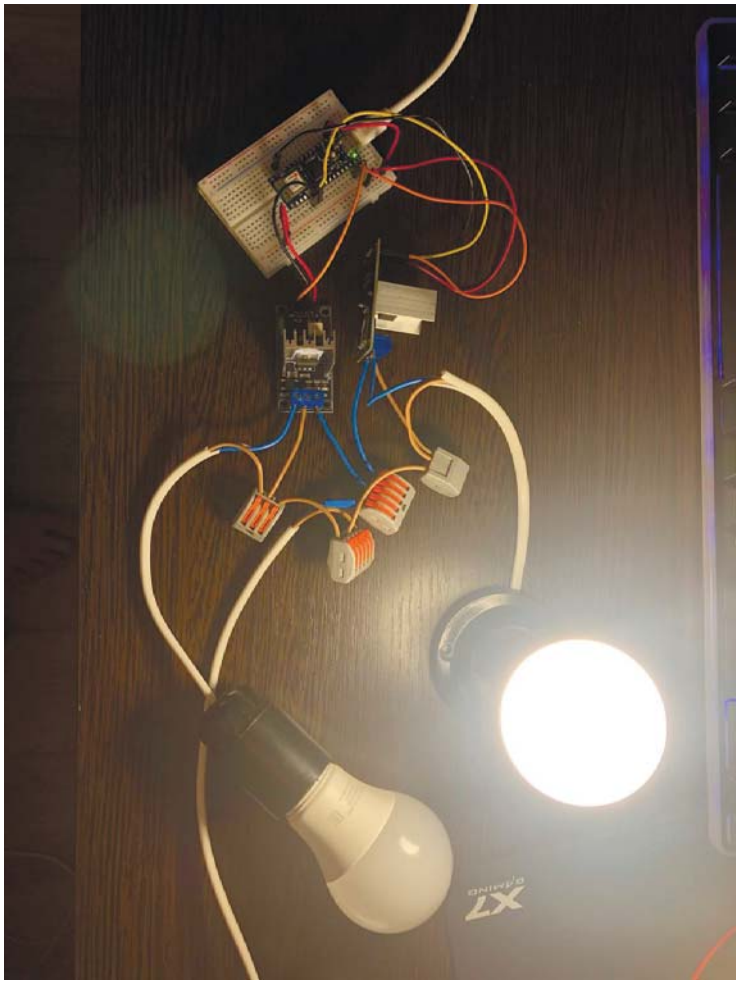

Рисунок 17 – Результат включения в промежутке между закатом и восходом

4пКак и требовалось была включена лампа нейтрально белого цвета, что видно на рисунке 16, также корректно была включена лампа теплого цвета, что показано на рисунке 17.

Теперь протестируем изменение яркости для чего дождемся времени восхода солнца и проверим, как происходит переключение. На рисунке 18 показано зафиксированное изменение яркости после наступления восхода.

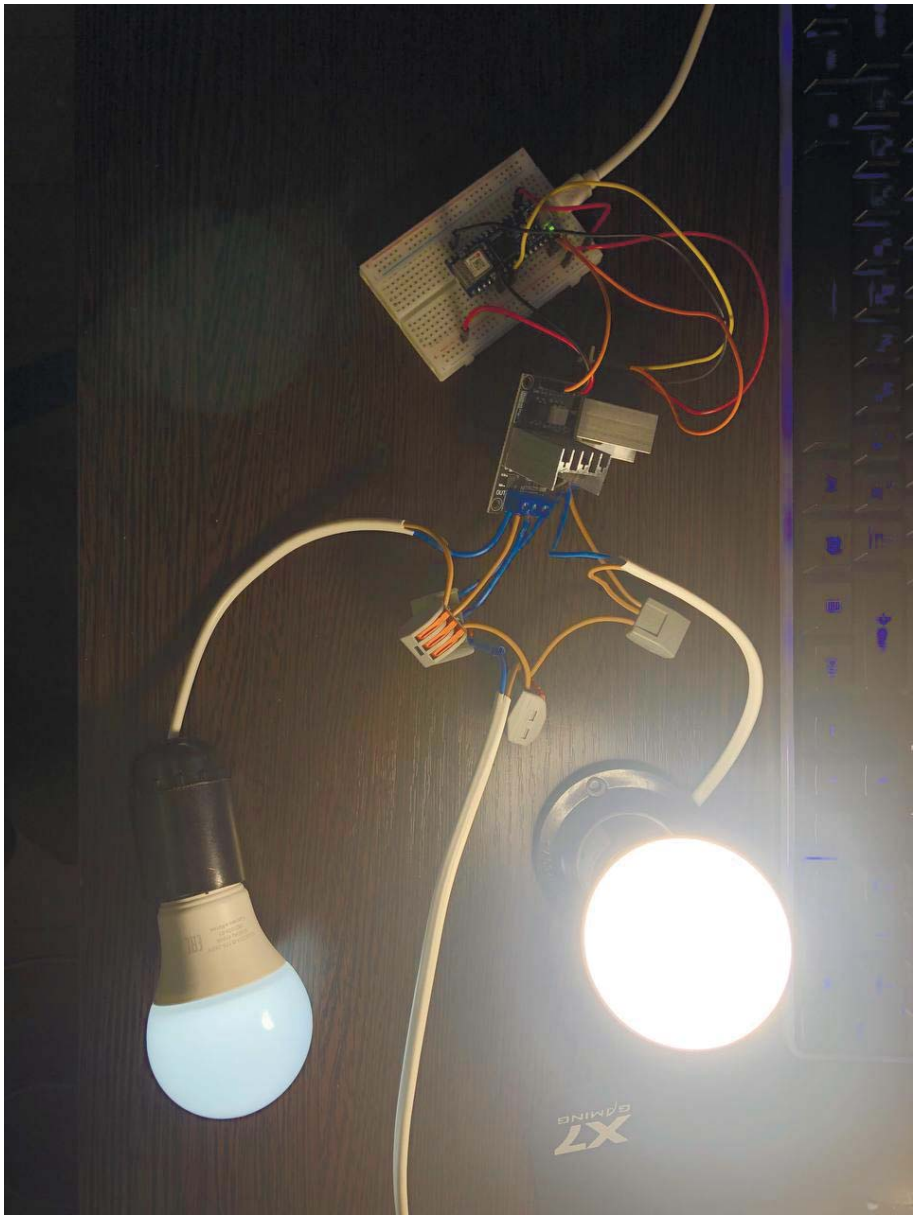

Рисунок 18 – Зафиксированное изменение яркости после восхода солнца

Данные тесты показали, что происходит корректное получение данных при помощи API и их последующая обработка. При включении включается соответствующая лампочка, при интервале от восхода до заката нейтрально белого света, при интервале от заката до восхода теплого цвета.

# ЗАКЛЮЧЕНИЕ

В данной выполненной выпускной квалификационной работе были рассмотрены и проанализированы существующие решения для автоматического управления домашнего освещения. Анализ выявил почти полное отсутствие возможности автоматического изменения цветовой температуры освещения в зависимости от времени восхода и заката.

Было проведено проектирование собственного решения. Разработаны аспекты архитектуры и основные алгоритмы. В результате был собран прототип системы, который прошел альфа-тестирование и в дальнейшем может быть использован с целью добавления функционала и его последующего удобного тестирования.

В дальнейшем предполагается разработка единой печатной платы, на которой будут размещены только необходимые компоненты, что позволит сделать систему более компактной, а также удешевить в производстве. Спроектировать и реализовать корпус светильника, куда будет устанавливаться система управления. А также дальнейшая оптимизация кода.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. API. – Текст : электронный // Википедия : свободная энциклопедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/API (дата обращения: 25.05.2022).

2. CAD-системы. – Текст : электронный // НПП «Фотограмметрия» : [официальный сайт]. – URL: https://photogrammetria.ru/100-cad-sistemy.html (дата обращения: 25.05.2022).

3. Протокол Диффри-Хеллмана на эллиптических кривых. – Текст : электронный // Википедия : свободная энциклопедия : [сайт]. – URL: https://en.wikipedia.org/wiki/Elliptic-curve_Diffie%E2%80%93Hellman (дата обращения: 25.05.2022).

4. ECDSA. – Текст : электронный // Википедия : свободная энциклопедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/ECDSA (дата обращения: 25.05.2022).

5. Среднее время по Гринвичу. – Текст : электронный // 24timezones : [сайт]. – URL: https://24timezones.com/chasovoy-poyas/gmt (дата обращения: 25.05.2022).

6. Что такое gnd на плате. – Текст : электронный // 4systems : [сайт]. – URL: https://4systems.ru/inf/chto-takoe-gnd-na-plate/ (дата обращения: 25.05.2022).

7. HTTP. – Текст : электронный // MDN Web Docs : [сайт]. – URL: https://developer.mozilla.org/ru/docs/Web/HTTP (дата обращения: 25.05.2022).

8. 10 лучших IDE. – Текст : электронный // timeweb : [сайт]. – URL: https://timeweb.com/ru/community/articles/5-luchshih-ide-1 (дата обращения: 25.05.2022).

9. Что такое IoT и что о нем следует знать. – Текст : электронный / блог компании OTUS // ХАБР : [сайт]. – 29 марта 2021. – URL: https://habr.com/ru/company/otus/blog/549550/ (дата обращения: 25.05.2022).

10. Что такое JSON. – Текст : электронный // ХАБР : [сайт]. – 2 мая 2021. – URL: https://habr.com/ru/post/554274/ (дата обращения: 25.05.2022).

11. Srinivasan, R. Pulse Skipping Modulated Buck Converter – Modeling and Simulation / R. Srinivasan, P. Vanaja Ranjan. – Текст : электронный // Circuits and Systems. – 2010. – Vol. 1 No. 2. – С. 59-64. – URL: https://www.scirp.org/journal/paperinformation.aspx?paperid=2832 (дата обращения: 25.05.2022).

12. REST API. – Текст : электронный // SkillFactory.Блог : [сайт]. – URL: https://blog.skillfactory.ru/glossary/rest-api/ (дата обращения: 25.05.2022).

13. Середина, В. SDK и API: в чем разница?. – Текст : электронный / В. Середина // ХАБР : [сайт]. – 13 августа 2021. – URL: https://habr.com/ru/company/ibm/blog/572754/ (дата обращения: 25.05.2022).

14. Всемирное координированное время. – Текст : электронный // Википедия : свободная энциклопедия : [сайт]. – URL: https://en.wikipedia.org/wiki/Coordinated_Universal_Time (дата обращения: 25.05.2022).

15. AC Dimmer Module // RobotDyn : [официальный сайт]. – URL: https://robotdyn.com/ac-light-dimmer-module-1-channel-3-3v-5v-logic-ac-50-60hz-220v-110v.html (дата обращения: 15.05.2022).

16. Продукция // WiZ : [официальный сайт]. – URL: https://www.wizconnected.com/ru-ru/products#filters=WIZ_MODERN_BULBS_SU (дата обращения: 4.04.2022).

17. Умный дом : Свет // Xiaomi : [официальный сайт]. – URL: https://www.mi.com/ru/smart-home#lighting (дата обращения: 4.04.2022).

18. Обзор продуктов // Philips-Hue : [официальный сайт]. – URL: https://www.philips-hue.com/ru-ru/products (дата обращения: 4.04.2022).

19. What is Arduino?. – Текст : электронный // Arduino : [официальный сайт]. – URL: https://www.arduino.cc/en/Guide/Introduction (дата обращения: 10.05.2022).

20. Карун, Д. П. Разработка прототипа устройства по обеспечению бесперебойного электроснабжения узла связи на базе аппартно-вычислительной платформы Arduino / Д. П. Карун, А. В. Тезин. – Текст : электронный // Universum: технические науки. – 2021. – №1 (82). – URL: https://7universum.com/ru/tech/archive/item/11166 (дата обращения: 10.05.2022)

21. Контроллеры Iskra // Amperka : [официальный сайт]. – URL: https://amperka.ru/collection/iskra (дата обращения: 15.05.2022).

22. STM32 MCU & MPU Eval Tools // STMicroelectronics : [официальный сайт]. – URL: https://www.st.com/en/evaluation-tools/stm32-mcu-mpu-eval-tools.html https://www.st.com/content/st_com/en.html(дата обращения: 15.05.2022).

23. Алексеевский, П. И. Обучение студентов программированию с использованием отладочных комплектов stm32 Discovery / П. И. Алексеевский. – Текст : электронный // Педагогическое образование в России. – 2018. – С. 12-17. – URL: http://elar.uspu.ru/handle/uspu/11065 (дата обращения: 15.05.2022).

24. Sunset and sunrise times API. – Текст : электронный // sunset sunrise : [сайт]. – URL: https://sunrise-sunset.org/api (дата обращения: 15.05.2022).

25. APIs. – Текст : электронный // weatherapi : [сайт]. – URL: https://www.weatherapi.com/ (дата обращения: 15.05.2022).

26. Weather API // OpenWeather : [официальный сайт]. – URL: https://openweathermap.org/api (дата обращения: 15.05.2022).

27. API Яндекс.Погоды. – Текст : электронный // Яндекс Погода : [официальный сайт]. – URL: https://yandex.ru/dev/weather/ (дата обращения: 15.05.2022).

28. Сафронов, А. Управление сетевым питанием с помощью pic10f204 и симистора / А. Сафронов. – Текст : электронный // Компоненты и технологии. – 2005. – №2 (46)’2005. – С. 130-132 – URL: https://kit-e.ru/protection/upravlenie-setevym-pitaniem-s-pomoshhyu-pic10f204-i-simistora/ (дата обращения: 25.05.2022)

29. Трехфазный регулятор мощности своими руками. – Текст : электронный // Мастерская Эдуарда Орлова : [сайт]. – URL: https://rustaste.ru/trekhfaznyjj-regulyator-moshhnosti-svoimi-rukami.html (дата обращения: 25.05.2022).

# ПРИЛОЖЕНИЕ А

## Исходный код системы

Листинг 1 – Исходный код системы управления светом

```
#include <WiFiNINA.h>
#include <ArduinoHttpClient.h>
#include <ArduinoJson.h>
#include <RTCZero.h>

#define SECRET_SSID "mantis home"           //имя беспроводной сети
#define SECRET_PASS "K47fePyv"              //пароль беспроводной сети

#define URL_TIMEAPI "worldtimeapi.org"
#define URL_WEATHERAPI "api.weatherapi.com"
#define ROUTE_WEATHERAPI "/v1/astronomy.json?key=0d8b99e61cf64bbda5234139221905&q="
#define URL_SUNSETSUNRISE "api.sunrise-sunset.org"

#define LAMP_WARM 11
#define LAMP_NEUTRAL 12

RTCZero rtc; //переменная для работы часов в реальном времени

String city = "Chelyabinsk";

int rawOffset = 0;
float Lat = 0;
float Lon =0;
char timeZone[20];
uint8_t sunriseHours;
uint8_t sunriseMinutes;
uint8_t sunsetHours;
uint8_t sunsetMinutes;

boolean isChange;


int dimming = 5;

WiFiClient client;

int CoordinateAndTimezoneFromJSON(String message){
  StaticJsonDocument<1000> doc;

  DeserializationError error = deserializeJson(doc, message);
  if(error){
    Serial.println("error code zone 1");
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error.c_str());
    return 1;
  }
  String location = doc["location"];
  error = deserializeJson(doc,location );
  if(error){
    Serial.println("error code zone 2");
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error.c_str());
    return 2;
  }
  Lat = doc["lat"];
  Serial.println(Lat);
  Lon = doc["lon"];
  Serial.println(Lon);

  strcpy(timeZone,doc["tz_id"]);

  Serial.println(timeZone);
  delete &doc;
```

```
  return 0;
}


void getLocationAndTimeZone(){
  Serial.println("making request");
  HttpClient httpWeatherAPI(client,URL_WEATHERAPI , 80);
  httpWeatherAPI.get(ROUTE_WEATHERAPI+city);


  while(httpWeatherAPI.connected()){
    if(httpWeatherAPI.available()){
      int statusCode = httpWeatherAPI.responseStatusCode();
      String result = httpWeatherAPI.responseBody();

      Serial.print("Status code: ");
      Serial.println(statusCode);
      Serial.println(result);

      if(statusCode == 200){

        if(CoordinateAndTimezoneFromJSON(result)==0){
          getTimeAPI();
        }

      }

    }
  }
  httpWeatherAPI.stop();
}

void getTimeFromAPI(String message){
  StaticJsonDocument<850> doc1;

  DeserializationError error1 = deserializeJson(doc1, message);
  if(error1){
    Serial.println("error code zone 3");
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error1.c_str());

    return;
  }
  rawOffset = doc1["raw_offset"];
  Serial.println(rawOffset);
  rtc.setEpoch((unsigned long int)doc1["unixtime"]+(unsigned long int)rawOffset+2);
  delete &doc1;
}

void getTimeAPI(){
  Serial.println("making request");
  HttpClient httpTimeAPI(client,URL_TIMEAPI , 80);
  char tempTimeZone [50] = "/api/timezone/";
  strcat(tempTimeZone,timeZone);
  Serial.println(tempTimeZone);
  httpTimeAPI.get(tempTimeZone);


  while(httpTimeAPI.connected()){
    if(httpTimeAPI.available()){
      int statusCode = httpTimeAPI.responseStatusCode();
      String result = httpTimeAPI.responseBody();

     Serial.print("Status code: ");
     Serial.println(statusCode);
      Serial.println(result);

      if(statusCode==200){
        getTimeFromAPI(result);
      }
```

```
    }
  }

  httpTimeAPI.stop();
}

void fromCharToInt(char sunTime[12], char Type){
  uint8_t tempH, tempM;

  if (sunTime[1] == ':'){
        tempH = sunTime[0] - '0';
        tempM = (sunTime[2] - '0') * 10;
        tempM += (sunTime[3] - '0');
        if (sunTime[8] == 'P')
            tempH += 12;
        tempH += (rawOffset / 3600);

        if (tempH >= 24)
            tempH -= 24;
    } else if (sunTime[2] == ':'){
        tempH = (sunTime[0] - '0')*10;
        tempH += (sunTime[1] - '0');
        tempM = (sunTime[3] - '0') * 10;
        tempM += (sunTime[4] - '0');
        if (sunTime[9] == 'P')
            tempH += 12;
        if (sunTime[9]=='A')
            tempH -= 12;
        tempH += (rawOffset / 3600);
        if (tempH >= 24)
            tempH -= 24;
    }
  switch(Type){
    case 'r':{
      sunriseHours = tempH;
      sunriseMinutes = tempM;
      Serial.print("sunrise: ");
      Serial.print(sunriseHours);
      Serial.print(':');
      Serial.println(sunriseMinutes);
      break;
    }
    case 's':{
      sunsetHours = tempH;
      sunsetMinutes = tempM;
      Serial.print("sunset: ");
      Serial.print(sunsetHours);
      Serial.print(':');
      Serial.println(sunsetMinutes);
      break;
    }
  }
}

void getSunSetRiseJSON(String message){
  StaticJsonDocument<1000> doc2;

  DeserializationError error2 = deserializeJson(doc2, message);
  if(error2){
    Serial.println("error code zone 1");
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error2.c_str());
    return ;
  }
  String results = doc2["results"];
  delay(20);
  Serial.println(results);
  error2 = deserializeJson(doc2,results);
  if(error2){
    Serial.println("error code zone 2");
```

```
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error2.c_str());
    return ;
  }
  char str[12];
  strcpy(str,doc2["sunrise"]);
  fromCharToInt(str,'r');
  strcpy(str,doc2["sunset"]);
  fromCharToInt(str,'s');

  Serial.println();
}

void getTimeSunsetSunrise(){
  Serial.println("making request");
  HttpClient httpTimeAPI(client,URL_SUNSETSUNRISE , 80);
  httpTimeAPI.get("/json?lat="+String(Lat,2)+"&lng="+String(Lon,2));


  while(httpTimeAPI.connected()){
    if(httpTimeAPI.available()){
      int statusCode = httpTimeAPI.responseStatusCode();
      String result = httpTimeAPI.responseBody();

      Serial.print("Status code: ");
      Serial.println(statusCode);
      Serial.println(result);

      if(statusCode==200){
        getSunSetRiseJSON(result);
      }

    }
  }
  httpTimeAPI.stop();
}


void setup()
{

  Serial.begin(9600);
  while (!Serial) {
    ;
  }
  while (WiFi.status() != WL_CONNECTED){
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(SECRET_SSID);
    WiFi.begin(SECRET_SSID, SECRET_PASS);
    delay(2000);
  }

  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  rtc.begin();
  if(city!=NULL){
    getLocationAndTimeZone();
    delay(500);
    if(Lat!=0 && Lon!=0){
       getTimeSunsetSunrise();
    }
  }
  pinMode(LAMP_WARM, OUTPUT
  pinMode(LAMP_NEUTRAL,OUTPUT);
  attachInterrupt(digitalPinToInterrupt(2), zero_crosss_int, RISING);
  noInterrupts();
  if(rtc.getHours()>=sunriseHours && rtc.getHours()<=sunsetHours){
    digitalWrite(LAMP_NEUTRAL, HIGH);
```

```
    digitalWrite(LAMP_WARM, LOW);
  } else {
    digitalWrite(LAMP_WARM,HIGH);
    digitalWrite(LAMP_NEUTRAL,LOW);
  }

}

void zero_crosss_int(){
  int dimtime = (75*dimming);
  delayMicroseconds(dimtime);
  if(isChange == false){
    digitalWrite(LAMP_NEUTRAL, HIGH);
    delayMicroseconds(10);
    digitalWrite(LAMP_NEUTRAL, LOW);
  } else {
    digitalWrite(LAMP_WARM, HIGH);
    delayMicroseconds(10);
    digitalWrite(LAMP_WARM, LOW);
  }
}


void loop(){

  if((rtc.getHours() == 12 && rtc.getMinutes()==25 && rtc.getSeconds()==40) && city!=NULL){
    getTimeAPI();
  }

  if((rtc.getHours()==0 && rtc.getMinutes() ==35 &&rtc.getSeconds()==40) && city != NULL){
    getTimeSunsetSunrise();
    getTimeAPI();
  }
  if((rtc.getHours() == sunriseHours && rtc.getMinutes() == sunriseMinutes) && city !=NULL){
    interrupts();
    digitalWrite(LAMP_WARM, HIGH);
    for (int i=110; i >= 5; i--){
      dimming=i;
      delay(1000);
    }
    digitalWrite(LAMP_NEUTRAL,HIGH);
    isChange=true;
    for(int i=5;i<=110;i++){
      dimming=i;
      delay(1000);
    }
    noInterrupts();
    digitalWrite(LAMP_WARM, LOW);
  }
  if((rtc.getHours() == sunsetHours && rtc.getMinutes() == sunriseMinutes) && city != NULL){
    interrupts();
    digitalWrite(LAMP_NEUTRAL,HIGH);
    for(int i=110; i>=5;i--){
      dimming=i;
      delay(1000);
    }
    digitalWrite(LAMP_NEUTRAL, HIGH);
    isChange = false;
    for(int i=5;i<=110;i++){
      dimming = i;
      delay(1000);
    }
    noInterrupts();
    digitalWrite(LAMP_NEUTRAL, LOW);
  }

}
```