Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
ИНСТИТУТ НАНОТЕХНОЛОГИЙ В ЭЛЕКТРОНИКЕ, СПИНТРОНИКЕ И ФОТОНИКЕ

КАФЕДРА ЭЛЕКТРОНИКИ

На правах рукописи

УДК 621.38+681.51

Наумов Никита Сергеевич

ФИО полностью

Разработка устройств управления и сбора информации, входящих в состав киберфизической системы «Умный дом» с Web-интерфейсом.

тема проекта

Выпускная квалификационная работа специалиста

Направление подготовки (специальности)

14.05.04 Электроника и автоматика физических установок

Выпускная квалификационная работа защищена

« 28 » января 2021 г.

Оценка _____

Секретарь ГЭК _____

г. Москва

2021

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ДИПЛОМНОМУ ПРОЕКТУ НА ТЕМУ:

Разработка устройств управления и сбора информации, входящих в состав киберфизической системы «Умный дом» с Web-интерфейсом.

Студент-дипломник: _____ / Наумов Н.С. /
подпись ФИО

Руководитель проекта: _____ / Некрасов П.В. /
подпись ФИО

Рецензент проекта: _____ / Бакеренков А.С. /
подпись ФИО

Зам. Заведующего кафедрой №3_____ / Барбашов В.М. /
подпись ФИО

Москва 2021 г.

Оглавление

Аннотация

Данная работа посвящена созданию и внедрению компонентов киберфизической системы класса «Умный дом»: электронного устройства регулировки мощности осветительных приборов сети 220В, а также пользовательского Web-интерфейса для удаленного управления устройствами системы «Умный дом». В работе подробно описан маршрут проектирования узлов киберфизической системы. Каждая глава затрагивает отдельный этап разработки электронного устройства и пользовательского интерфейса.

В главе 1 приводятся понятия киберфизической системы и ее компонентов, а также описывается ее структура. Кроме того, в данной главе проведен обзор и анализ готовых решений, предлагаемых компаниями-производителями систем «Умный дом», с целью формирования требований к разрабатываемому устройству и интерфейсу пользователя.

Глава 2 посвящена проектированию аппаратной части электронного устройства. Разработана электрическая схема устройства, проведен подбор компонентов, промоделирована работа схемы и создан прототип печатной платы для отправки на фабрику-изготовитель.

В Главе 3 затрагиваются основные аспекты разработки программного обеспечения узлов киберфизической системы «Умный дом». Проиллюстрированы фрагменты программного кода свето-регулятора с описанием работы устройства. Также приводится структура исходного кода пользовательского Web-интерфейса. Полный код свето-регулятора и Web-интерфейса доступен по ссылкам https://gitlab.com/Writerous/dimmer и https://gitlab.com/Writerous/smart-home-2.0 , соответственно.

В главе 4 описан процесс внедрения узлов в систему «Умный дом»: подключение и конфигурация умного свето-регулятора, размещение Web-интерфейса в сети Internet.

Введение

С тех пор как в 1970-х годах Intel Corp. по заказу японской компании производителя калькуляторов Busicom Corp. выпустила в свет первый в мире микропроцессор, электронная индустрия шагнула далеко вперед. Сегодня электронные вычислительные устройства можно встретить во всех сферах человеческой деятельности, начиная от домашнего быта и заканчивая наукоемкими технологиями. С развитием вычислительной техники происходит и совершенствование программного обеспечения. Последние достижения в электронике и кибернетике объединяются для обеспечения большей энергоэффективности, производительности, безопасности, доступности и удобства пользователя. На стыке различных областей науки и техники зарождаются наиболее перспективные технологические концепции, принципиальной реализацией которых занимаются ведущие компании-производители всего мира. Среди прочих повышенное внимание специалистов и потребителей привлекают киберфизические системы. Они призваны улучшить инфраструктуру целых городов, повысить эффективность работы предприятий и сделать повседневную жизнь людей комфортнее.

Одной из типичных реализаций киберфизических систем является так называемый «Умный дом». Актуальность систем такого рода основывается на желании рядового пользователя упростить собственную жизнь, которое удается удовлетворить с развитием информационных и электронных технологий. Тем не менее в погоне за развертыванием собственных масштабных киберфизических систем крупные компании не уделяют достаточно внимания простым деталям и оставляют целый ряд нерешенных вопросов.

В рамках данной выпускной квалификационной работы были подробно изучены нюансы функционирования современных киберфизических систем, разработано собственное электронное устройство плавной регулировки яркости верхнего освещения в жилом помещении, а также создан

пользовательский Web-интерфейс для системы «Умный дом», развитием которой занимаются сотрудники и студенты кафедры №3 («Электроники») НИЯУ МИФИ.

Для достижения намеченных целей по внедрению свето-регулятора, управляемого непосредственно или через Web-интерфейс, в киберфизическую систему «Умный Дом» была поставлена задача разработать электронное устройство для киберфизической системы «Умный дом», позволяющее управлять приборами сети 220В, собирать информацию о их состоянии, а также контролировать параметры токопотребления в ручном режиме и дистанционно из личного кабинета с использованием Web-технологий, а тажке реализовать интуитивный Web-интерфейс на базе масштабируемого Web-приложения с последующей поддержкой и расширением до Progressive Web Application.

В качестве исходных данных киберфизической системы «Умный дом» были приняты следующие компоненты: сервер-шлюз, реализованный на микрокомпьютере Raspberry Pi, облачная база данных класса NoSQL Realtime Database на платформе Google Firebase.

Для написания программного кода выбраны языки программирования: C, C++, CSS, HTML, JavaScript, TypeScript, Pascal, Python.

Для проектирования электронного устройства и интерфейса пользователя сформирован список прикладного программного обеспечения, содержащий: Altium Designer, Arduino IDE, Visual Studio Code, LTSpice.

1. Основные определения и обзор киберфизических систем

1.1 Киберфизические системы. Основные определения

1.1.1 Киберфизические системы

Термин «киберфизическая система» был предложен в 2006 году доктором Хелен Гилл, руководившей тогда программой Национального научного фонда США. Однако эти системы имеют гораздо более долгую историю, восходящую к началу кибернетики, которую математик Норберт Винер определил, как науку об управлении и коммуникации между машинами и людьми. [1]

Киберфизическая система - информационно-технологическая система сбора и хранения информации об объектах окружающей среды, и осуществляющая контроль за показателями электронных устройств, взаимодействующих с физическим миром.

Использование киберфизических систем главным образом направлено на повышение эффективности работы физических систем за счет улучшения функциональности, надежности, безопасности и удобства использования таких систем. [2]

Киберфизические системы строятся на основе современных технологических концепций таких как: облачные вычисления, интернет вещей, информационная безопасность, дополненная реальность, большие данные, автономные роботы, интерактивные устройства.

По размеру киберфизические системы разделяют на:

- Киберфизические системы в масштабах города («Умные города»)
- Промышленные киберфизические системы («Умные предприятия»)
- Частные киберфизические системы («Умный дом»)

Размер киберфизической системы сильно коррелирует с ее сложностью, местом назначения и целью работы. Примерами «Умных городов» служат:

- Сингапур,
- Масдар (ОАЭ).

Примерами умных предприятий выступают:

- Toshiba Virtual Power Plant - Виртуальная электростанция Toshiba
- EWA (Electronics works Amberg) – умная фабрика по производству печатных плат Siemens [3]

Примерами умных домов служат решения таких компаний, как:

- Xiaomi
- Apple
- Redmond
- Fibaro

В рамках дипломной работы рассматривались киберфизические системы класса «Умный дом». Далее, именно киберфизические системы назначения «Умный дом» будут упоминаться в работе.

Разные производители предлагают разные подходы к реализации систем «Умный дом», кроме того конечные потребители предъявляют разные требования к комплексам в зависимости от помещений и бюджета.

Из этого следует, что все технологические комплексы «Умный дом» различаются устройствами, применяемыми в этих системах, и способах их взаимодействий как между собой, так и с конечным пользователем и окружающей средой. Тем не менее, большинство систем «Умный дом» имеют схожие принципы работы, и оснащаются умными устройствами, выполняющими похожие задачи. Значит, можно выделить необходимый набор устройств и программного обеспечения, без которого не обойдется ни одна киберфизическая система «Умный дом». Базовая структура системы «Умный дом» содержит прежде всего набор «Умных устройств», сервер-шлюз

(коммутатор), базу данных и интерфейс управления (Web или Android). Связь узлов киберфизической системы иллюстрирует рисунок 1.1.

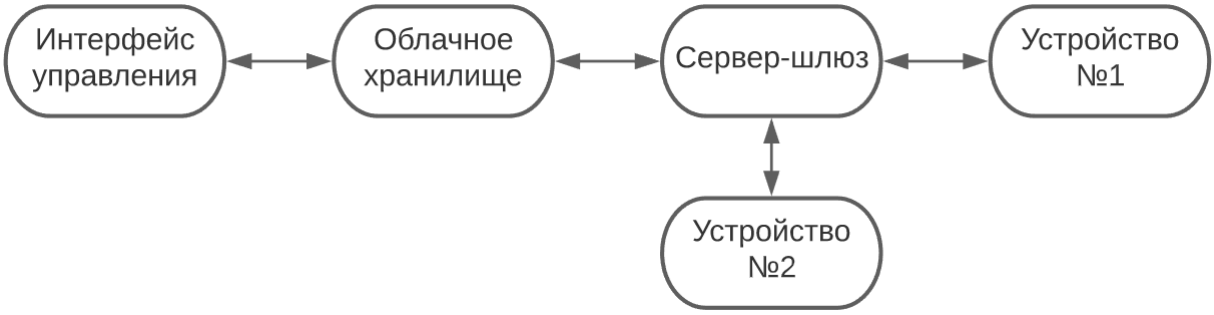

Рисунок 1.1 – Базовая структурная схема системы «Умный дом»

1.1.2 Умное устройство

Умным называют сложное вычислительное устройство, способное принимать и передавать данные на другие устройства при помощи протоколов беспроводной связи, таких как Bluetooth, NFC, Wi-Fi, 3G и других, а также работающее в некоторой степени интерактивно и автономно.

С развитием интернета вещей умными стали становится бытовые устройства, такие как: утюги, чайники, стиральные машины, розетки, выключатели, батареи, лампы и даже шторы. [4]

Интеллектуальные устройства выпускаются как для самостоятельной работы, так и для работы в системах класса «Умный дом». Такие устройства – неотъемлемая часть любой киберфизической системы. Работая в таких системах, умное устройство должно выполнять ряд обязательных функций таких как:

- функция управления, если речь идет об управляющем устройстве (контроллере), например, умная розетка, умный выключатель света
- функция измерения, если речь идет об измеряющем устройстве (датчике), например, умный датчик протечки, освещенности, температуры или другого показателя окружающей среды
- отправка данных и прием команд

1.1.3 Сервер-шлюз

Одиночное интеллектуальное устройство способно работать в системе «Умный дом», подключаясь к домашней сети Wi-Fi. В случае если в системе уже функционирует несколько устройств, подключение к домашней Wi-fi сети затруднительно, в таких случаях целесообразно использовать единый сервер-шлюз. Сервер-шлюз создает локальную беспроводную сеть, к которой уже подключаются умные устройства. Сервер-шлюз позволяет снять с устройств часть задач по работе в локальной сети. Умные устройства, подключаясь к единому серверу-шлюзу, реализуют топологию «звезда» (одна из топологий локальных сетей). [5]

Обязательные функции, которые выполняет сервер-шлюз:

- создание локальной сети для умных устройств
- сбор и агрегация данных с устройств
- опрос устройств для проверки работоспособности
- отправка команд умным устройствам, с целью внесения изменений в их работу
- запись и чтение данных из базы данных

Иногда сервер-шлюз сочетает в себе функции коммутатора локальной сети и сервера обработки данных, кроме того, нередко сервер-шлюз является интерактивным устройством. Значит, дополнительные функции, которые может выполнять сервер-шлюз:

- Приоритетный опрос устройств
- Сбор, обработка и хранение данных
- «умное» управление устройствами (создание сценариев управления устройствами)
- Резервное копирование данных
- Анализ данных с использованием искусственного интеллекта
- Выполнение иных интерактивных функций.

Для хранения данных сервер-шлюз использует базу данных. Современные киберфизические системы «Умный дом» используют базу данных расположенную не на устройстве сервере-шлюзе, а удаленно, то есть в облаке. Это открывает массу возможностей для контроля за устройствами умного дома.

1.1.4 Облачные технологии

Как отмечалось ранее, использование удаленной базы данных позволяет раскрыть потенциал киберфизической системы «Умный дом», делая ее частично распределенной. Облачные технологии включают в себя кроме облачной базы данных, некоторые серверные функции, снимая тем самым с сервера-шлюза часть задач. Облачные технологии могут включать:

- Облачные базы данных
- Сервер авторизации
- Сервер приложений
- Вычислительный сервер

Облачные технологии позволяют:

- контролировать устройства и проводить мониторинг состояния системы из любой точки мира через интернет при помощи пользовательского интерфейса
- упростить процесс интеграции киберфизической системы и снизить стоимость установки, уменьшив количество электронных устройств в помещении, оставив только самые необходимые

Для контроля за показателями системы через интернет используют различные пользовательские интерфейсы: Android-приложения, Web-приложения, компьютерные программы и другие.

1.1.5 Пользовательские интерфейсы

Для того чтобы обеспечить удаленный контроль за показателями киберфизической системы, необходимы способы наблюдения за ее показателями, а также панели управления умными устройствами. Эти инструменты реализуются в рамках пользовательского интерфейса.

Пользовательским называют интерфейс, обеспечивающий процесс передачи информации от пользователя к программно-аппаратным компонентам, входящим в состав компьютерной системы и наоборот. [6]

Наиболее часто пользовательские интерфейсы реализуются в Web-приложениях, Android-приложениях и компьютерных программах. Такие приложения получают данные о состоянии системы в реальном времени по сети и отображают их в удобном для пользователя виде.

В рамках дипломной работы особое внимание уделялось созданию Web-приложения, как одного из самых популярных решений по реализации пользовательского интерфейса.

Web-приложение - сложное распределенное приложение, состоящее из клиентской и серверной частей, взаимодействие между которыми осуществляется по сети.

В киберфизических системах «Умный дом» Web-приложение, как правило, расположено на сервере приложений, который является частью облачных технологий.

1.2 Обзор рынка киберфизических систем «Умный дом»

Рынок киберфизических систем класса «Умный дом» растет с каждым годом. Среди компаний, предлагающих свои решения по системам «Умный дом», есть как компании, хорошо зарекомендовавшие себя как производители электронных портативных устройств, например, Xiaomi, Redmond, так и компании, узко специализирующиеся на системах «Умный дом», например, Fibaro, Rubetek. [7] Далее будут рассматриваться аппаратные и программные комплексы производителей, это необходимо перед проектированием собственных умных устройств и программного обеспечения. Такой подход позволит, определить достоинства и недостатки уже имеющихся на рынке решений, чтобы повторить сильные стороны устройств одних производителей и избежать ошибок, допущенных другими производителями.

1. Xiaomi

Китайская компания, занимающаяся выпуском умных устройств. В каталоге компании можно найти умные светильники, розетки, лампочки, датчики влажности, вешалки, датчики температуры многое другое. [7]

Достоинства по аппаратной части:

- Обилие устройств
- Подключение по Wi-Fi, Bluetooth
- Приятная стоимость
- Высокое качество изделий (качество сборки показано на рисунке 1.2)

Рисунок 1.2 – Пример устройства марки Xiaomi

Недостатки по аппаратной части:

- Есть риск встретить устройства с неподходящими для России вилками
- Отсутствие у автономных устройств энергоемких интерфейсов, таких как Z-Wave

Достоинства по программной части:

- Наличие Android-приложения, iOS-приложения

Недостатки по программной части:

- Andriod-приложение и IOS-приложение частично не руссифицированы
- Отсутствует Web-приложение и Windows-приложение, что означает невозможность управления умным домом с компьютера [8]

2. Fibaro

Польская компания, занимающаяся разработкой и внедрением систем «Умный дом». В арсенале производителя набор устройств, способный укомплектовать весь дом. Специалисты компании сами подбирают устройства и занимаются установкой системы «Умный дом» преимущественно под ключ. [7] Установочный комплект системы содержит в обязательном порядке мощный вычислительный сервер, интегрированный с сервером-шлюзом. Размеры вычилительного сервера проиллюстрированы на рисунке 1.3.

Рисунок 1.3 – Вычислительный сервер Fibaro

Достоинства по аппаратной части:

- Обилие устройств
- Высокое качество изделий
- Поддержка энергоэффективного протокола Z-Wave

Недостатки по аппаратной части:

- Высокая стоимость устройств
- Отсутствие упоминаний о поддержке продуктов других изготовителей
- Большие габариты устройств

Достоинства по программной части:

- Наличие Android-приложения, iOS-приложения

Недостатки по программной части:

- Замедленное действие на команды пользователя
- Уязвимости в php-серверах системы [9]

3. Redmond

Русский производитель бытовой техники предлагает линейку умных устройств для последующей интеграции в систему «Умный дом» в партнерстве с Яндекс. Экосистема умного дома Redmond слабосвязная и не предлагает решений по автоматизации функционирования датчиков и контроллеров. [10]

Достоинства по аппаратной части:

- Невысокая стоимость устройств

Недостатки по аппаратной части:

- Использование Bluetooth Low Energy с диапазоном действия до 10 метров

- Невысокое качество изделий (качество сборки показано на рисунке 1.4)
- Использование смартфона в качестве сервера-шлюза

Рисунок 1.4 – Розетка марки Redmond

Достоинства по программной части:

- Наличие Andriod-приложения и iOS-приложения

Недостатки по программной части:

- Отсутствие таймаута на восстановление соединения с сервер-шлюзом
- Недочеты в работе приложений
- Недочеты в работе умных устройств

Таким образом, в ходе обзора киберфизических систем «Умный дом» популярных производителей были выявлены как удачные моменты и взвешенные решения, так и уязвимости, недочеты.

Выводы

Киберфизическая система – комплексная технологическая система, структура которой главным образом определяется экосистемой умных устройств и облачными сервисами. Система «Умный дом» одна из представителей малых киберфизических систем, активно развивающихся в мире современной пользовательской электроники. Крупные технологические компании предлагают свои реализаций систем «Умный дом» как для установки комплекса под ключ, так и для использования ограниченного набора умных устройств. Наиболее сбалансированное решение для российского пользователя предлагает компания Xiaomi. Качественные устройства по разумной цене и современное программное обеспечение с массой функций. К сожалению, в арсенале умных устройств Xiaomi не нашлось место умному свето-регулятору, работающему не только с умными лампами. Кроме того, отсутствует и Web-приложение для управление умным домом. В рамках разработки киберфизической системы «Умный дом», работа над которой проводится сотрудниками и студентами кафедры 3 («Электроники»), моей задачей была реализация умного свето-регулятора (Диммера) и Web-интерфейса. Конкретные этапы разработки будут освещены в следующих главах.

2. Аппаратная часть

2.1 Общая структура

Аппаратная часть любой киберфизической системы главным образом состоит из умных устройств, соединенных в единую локальную сеть. Нередко появляется необходимость в мастер-устройстве, называемом сервер-шлюз, которое берет на себя роль по управлению остальными.

К настоящему моменту киберфизическая система «Умный дом» насчитывает несколько устройств, такие как: Метео-станция, Умная розетка.

В рамках дипломной работы проводилась разработка умного устройства плавной регулировки яркости освещения в помещениях для последующей интеграции в киберфизическую систему «Умный дом». К умному свето-регулятору предъявлялись следующие требования:

- работа от сети 220В
- осуществлять близкую к линейной регулировку мощности на нагрузке
- измерять относительную освещенность комнаты для корректирования мощности на нагрузке в автоматическом режиме
- работать как с лампами накаливания, так и с диммируемыми светодиодными лампами,
- предоставлять пользователю возможность непосредственного управления устройством при помощи кнопок на лицевой панели
- предоставлять пользователю возможность удаленного управления устройством при помощи Web-интерфейса или Android-приложения в рамках системы «Умный дом»
- габариты устройства должны обеспечивать конечному потребителю возможность установить свето-регулятор в настенную монтажную коробку

Часть этих требований может быть реализована программным способом, другая же часть – аппаратно.

Процесс разработки аппаратной части свето-регулятора включает множество этапов:

- выбор технологии
- создание электрической схемы
- моделирование электрической схемы
- подбор компонентов
- разработка печатной платы
- монтаж компонентов на печатную плату
- сборка устройства в корпус

Главные этапы разработки проводились в рамках дипломной работы от выбора технологий до монтажа компонентов на печатную плату и будут освещены в этой главе.

2.2 Обзор диммеров

Диммер (свето-регулятор) – электронное устройство, используемое для регулировки яркости света, излучаемого осветительными приборами, посредством изменения потребляемой ими мощности.

2.2.1 Классификация

По встраиванию в помещении разделяют два типа диммеров

1. Диммер встраиваемый

   Это устройства, которые объединяют в себе коммутацию всей линии верхнего освещения и управления выходной мощностью, выдаваемой на эту линию. К достоинствам следует отнести возможность управления встроенным освещением, на базе ламп накаливания или диммируемых светодиодных ламп. Чаще всего устанавливаются в монтажную коробку. Недостатком таких диммеров является избирательность к типу лампы. Поэтому при выборе ламп необходимо обращать внимание на совместимость работы лампы с установленным в помещении диммером.

2. Диммеры-лампы

   Это отдельные светодиодные лампы, внутри которых находится схема управления мощностью, выдаваемой на осветительный элемент. Управление такими лампами происходит при помощи пультов, и как следствие, необходимо управлять каждой такой лампой. К достоинствам такой лампы следует отнести 100% возможность диммирования, потому что ответственность за совместимость работы схемы управления и осветительного элемента производитель берет на себя. Такие диммеры устанавливаются в люстры и светильники. К недостаткам таких диммеров следует отнести возрастающую стоимость при оснащении ими всего помещения, то есть при покупке более 3-4 ламп.

Схемотехнически выделяют главным образом три типа диммеров

- Диммер на реостате
- Автотрансформаторный диммер
- Электронный диммер [11]

Диммер на реостате – представляет из себя делитель напряжения с переменным резистором, что позволяет управлять выходным напряжением нагрузки. К достоинствам такой схемы следует отнести очевидную простоту реализации. К недостаткам относят отсутствие экономии электроэнергии, ведь энергия, не пошедшая в осветительный прибор, рассеивается на реостате, как следствие сильный нагрев устройства.

Диммер на автотрансформаторе – выдает чистый синус, но меньшей амплитуды, совместим со всеми типами ламп. Но из-за габаритов и сложности эксплуатации и стоимости не применяется.

Электронный диммер – состоит из симисторного или транзисторного ключей. Изменяет форму сигнала, но в большинстве случаев применим для осветительных элементов. Прост в эксплуатации и имеет сравнительно небольшие размеры. Таким образом, электронный диммер нашел широкое применение в бытовых и промышленных осветительных системах.

Так, для диммирования освещения в помещении использовалась схема электронного диммера. Далее под диммером будет подразумеваться именно электронный диммер.

По способу диммирования выделяют следующие схемы диммеров:

- Схемы фазового регулирования с отсечкой по переднему фронту
- Схемы фазового регулирования с отсечкой по заднему фронту (спаду)
- Схемы регулирования мощности с пропуском полуволн [12]

2.2.2 Способы диммирования

Фазовое регулирование с отсечкой по переднему фронту (или коротко: диммирование по переднему фронту)

Диммирование по переднему фронту означает, что коммутация нагрузки будет осуществляться по прохождении некоторого времени не превышающем полупериод синусоиды напряжения сети с момента пересечения синусоиды напряжения сети. [12] График, иллюстрирующий диммирование по переднему фронту представлен на рисунке 2.1

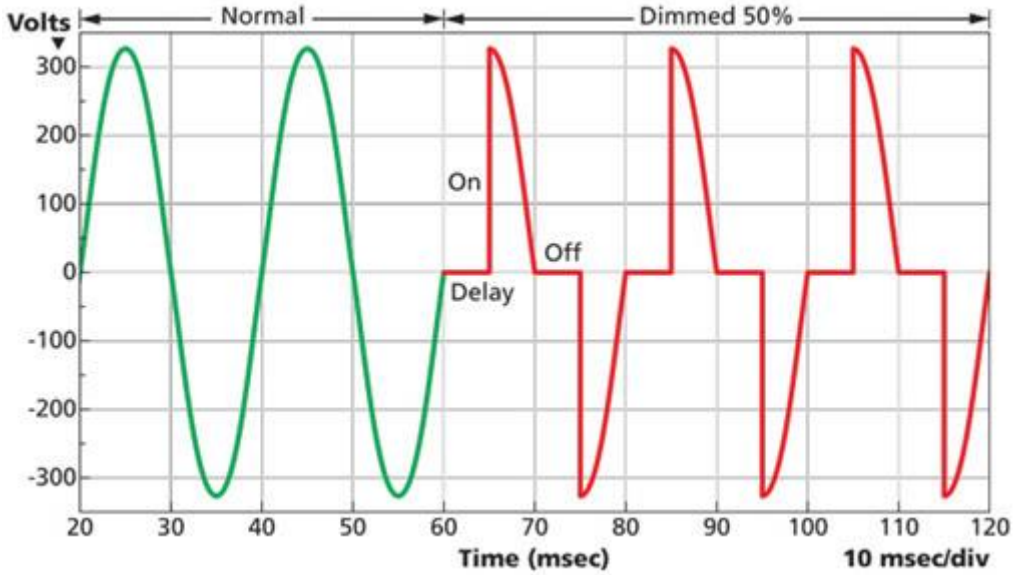

Рисунок 2.1 – Временная характеристика напряжения при диммировании по переднему фронту

Схемы с диммированием по переднему фронту выполняются на симисторном или тиристорном ключах. Являются достаточно простыми в реализации. Основаны на коммутации мощной нагрузки переменного тока. [13]

Ниже на рисунке 2.2 представлена схема диммера по переднему фронту, где управление мощностью на лампе осуществляется при помощи изменения сопротивления.

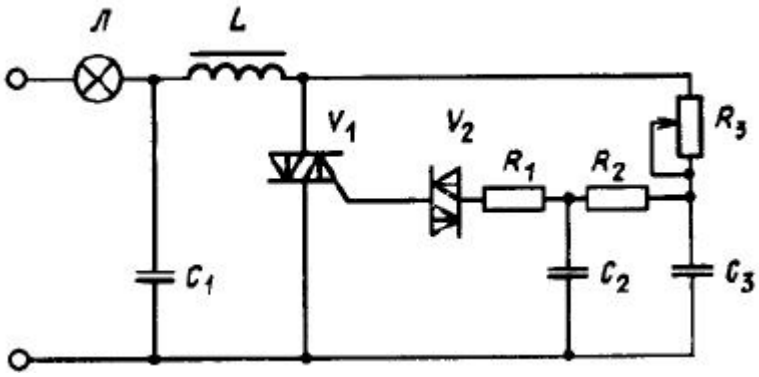

Рисунок 2.2 – Схема диммирования по переднему фронту

Напряжение на конденсаторе C3 нарастает согласно постоянной времени C3R3. Напряжение на C3 отпирает динистор V2. Динистор V2 подает импульс на открытие симистора V1. Симистор V1 остается открытым пока ток нагрузки (ток через симистор) не станет ниже тока удержания (характеристика симистора). Таким образом, чем больше сопротивление R3, тем через большее время от начала синусоиды напряжения сети произойдет коммутация цепи лампы. [14] [15]

Примечателен тот факт, что для перевода симистора в низкоомное (проводящее) состояния, необходимо подать короткий импульс тока на управляющий электрод. Симистор, переведенный в низкоомное состояние будет находится в нем, пока ток, протекающий через симистор, не станет меньше тока удержания (характеристики симистора). Резюмируя примечательную особенность поведения симистора в проводящем состоянии, можно сказать, что симистор отлично подходит для диммирования по переднему фронту, потому что будет закрываться сменой полярности напряжения сети в конце полупериода, и не подходит для реализации диммера по заднему фронту, потому что не способен закрываться, когда через него течет ток сети.

К достоинствам диммера по переднему фронту следует отнести:

1. Простоту реализации на симисторном или тиристорном ключе
2. Низкую стоимость конечного изделия
3. Способность к диммированию осветительных элементов

К недостаткам диммера по переднему фронту следует отнести:

1. Возможность использования вместе с таким устройством только ламп накаливания. Обычные светодиодные лампы не предназначены для диммирования, потому что их схема питания основана либо на базе балластного (конденсаторного) блока питания, либо на простом импульсном понижающем преобразователе. Если в лампе установлен импульсный блок питания, то драйвер в лампе компенсирует колебания напряжения до оптимального рабочего тока светодиода, в случае конденсаторного блока питания, авто коррекции тока светодиодов не произойдет. В любом случае при понижении входной мощности светодиодной лампы, она сначала замерцает, затем погаснет, то есть не будет наблюдаться участок плавной регулировки тока светодиодной цепи.

Фазовое регулирование с отсечкой по заднему фронту (или коротко: диммирование по заднему фронту (спаду)

Диммирование по заднему фронту (спаду) означает, что сначала полупериода синуса напряжения сети нагрузка будет подключена к сети, но за время, не превышающее полупериод синуса, до окончания полуволны напряжения сети произойдет отключение нагрузки от сети. На следующем полупериоде коммутация нагрузки к сети будет снова восстановлена, пока снова не наступит время для отключения нагрузки от сети. График, наглядно иллюстрирующий процесс диммирования по заднему фронту (спаду) представлен на рисунке 2.3.

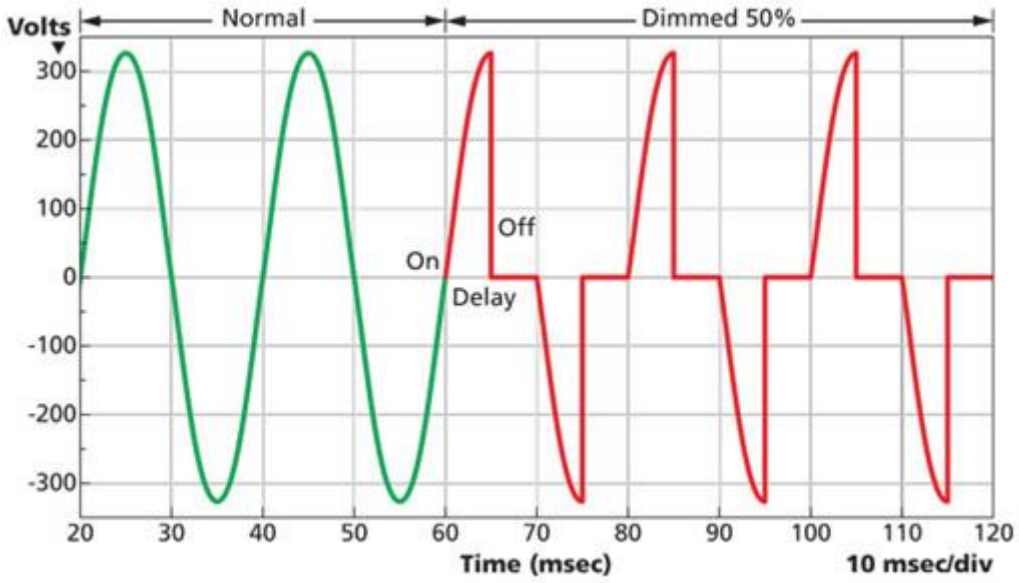

Рисунок 2.3 – Временная характеристика напряжения при диммировании по спаду

Схемы диммирования по заднему фронту (спаду) выполняются на транзисторных ключах. Являются более сложными в реализации, чем схемы диммирования по переднему фронту. На рисунке 2.4 представлена схема диммера по заднему фронту (спаду), выполненная на микросхеме TLC555 с управлением нагрузкой при помощи мощных MOSFET транзисторов, соединенных затворами и истоками. [16] [12]

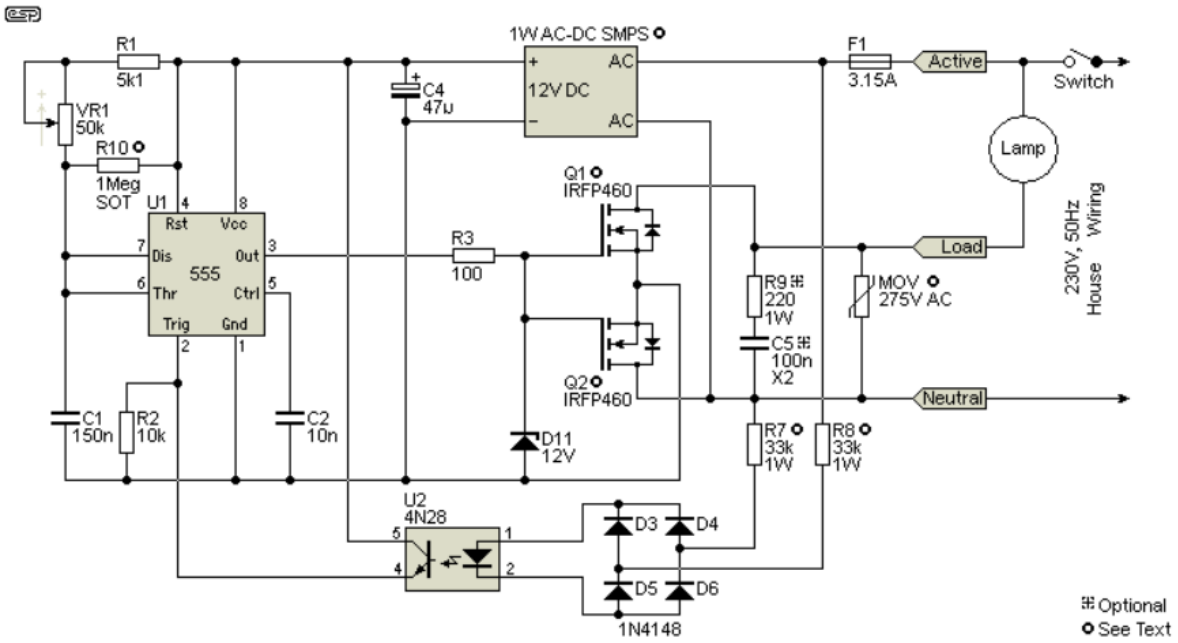

Рисунок 2.4 – Схема диммирования по спаду

Схему можно условно разделить на несколько частей, выполняющих разные функции. Так, например, диодный мост D3-D6, оптопара U2 и токоограничивающие сопротивления R7-R8 служат детектором перехода через ноль, и включают таймер U1. Таймер U1 выдает высокий уровень сигнала на выходе 3 около 12В. Высокий уровень сохраняется пока напряжение на C1 не станет около 8В, тогда произойдет переключение выхода таймера на низкий уровень. Время нарастания напряжения на конденсаторе C1 определяется постоянной времени C1VR1. Микросхема таймера U1 выходом 3 управляет двунаправленным ключом на мощных MOSFET Q1 и Q2, соединенных затворами и истоками. [17] Ключ на MOSFET транзисторах коммутирует нагрузку при помощи внутренних диодов в MOSFET транзисторах: когда ток течет от нагрузки к нейтрали, то в случае если транзисторы открыты Q1 пропускает ток через канал, а Q2 пропускает через встроенный диод. Образование внутреннего диода показано на рисунке 2.5.

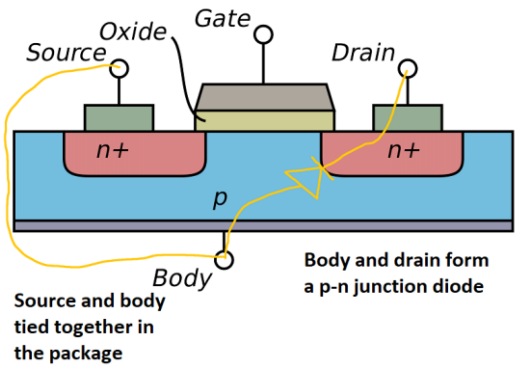

Рисунок 2.5 – Внутренний диод MOSFET

К достоинствам диммера по заднему фронту (спаду) можно отнести:

1. Низкую стоимость конечного изделия
2. Возможность управлять освещенностью как ламп накаливания, так и светодиодных ламп, предназначенных для диммирования. В отличии от диммера с фазовой регулировкой по переднему фронту диммер по заднему фронту (спаду) выдает напряжение формы, более благоприятной для блоков питания светодиодных ламп. Хотя по-прежнему обычные светодиодные лампы слабо поддаются диммированию из-за принципа работы блоков питания (оговаривалось в недостатках диммеров по переднему фронту), обычные светодиодные

лампы меньше подвержены преждевременному износу при работе через диммер по заднему фронту. С появлением схем диммирования по заднему фронту (спаду) появилась возможность диммировать специальные светодиодные лампы с пометкой «диммируемые» (предназначенные для диммирования). У блоков питания таких светодиодных ламп установлена специальная схема детектирования среднеквадратичного значения входного напряжения, которая устанавливает напряжение в цепи светодиодов внутри лампы пропорционально входному среднеквадратичному значению напряжения. [18] Таким образом, схема диммирования по заднему фронту показывает хорошие результаты как при работе с обычными лампами накаливания, так и со специальными светодиодными лампами, предназначенными для диммирования.

К недостаткам диммера по заднему фронту (спаду) стоит отнести:

1. Возросшую сложность электрической схемы по сравнению со светорегулятором по переднему фронту.

Регулирование мощности пропуском полуволн

Пропуск полупериодов – способ диммирования, при котором нагрузка коммутируется в момент перехода синуса напряжения сети через ноль, но не на каждой полуволне, а только для определенных полуволн. Реализацию алгоритма пропуска волн берет на себя схема диммирования. График, наглядно иллюстрирующий процесс регулирования выходной мощности, представлен на рисунке 2.6.

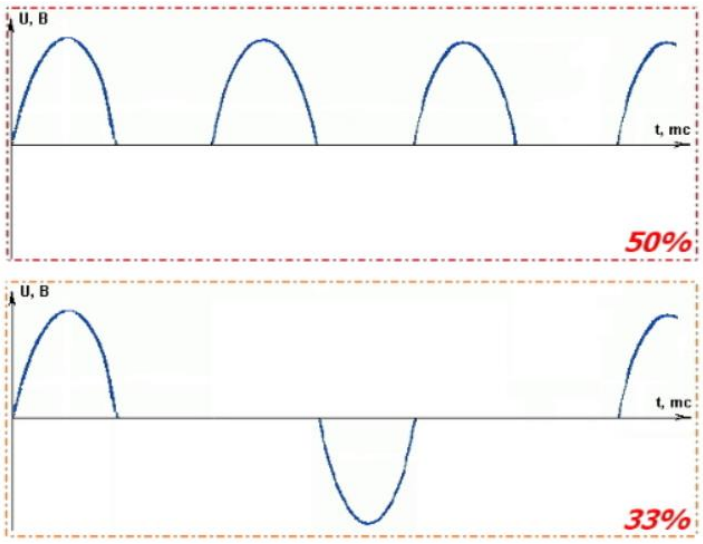

Рисунок 2.6 – Временная характеристика напряжения при пропуске полупериодов

Схемы диммирования с пропуском полуволн выполняются на симисторах или тиристорах, преимущественно с использованием микроконтроллера для реализации алгоритма пропуска полуволн. Как видно, из графика выше, полуволны должны быть равномерно распределены по времени для уменьшения колебания мощности на нагрузке. В качестве алгоритма выбора полуволн нередко применяют алгоритм из области машинной графики Брезенхэма. [19] На рисунке 2.7 приведен пример схемы управления мощности нагрузки по схеме пропуска полуволн:

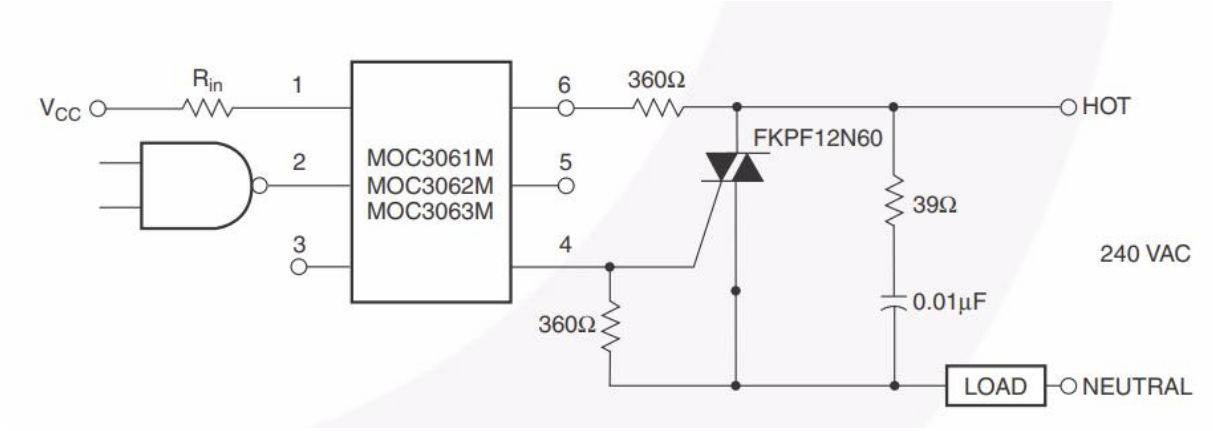

Рисунок 2.7 – Схема регулирования мощности пропуском полуволн

Под логическим элементом в схеме подразумевается порт микроконтроллера. MOC306X – это оптопара с симисторным выходом и

детектором перехода через ноль. Использование MOC3063X позволяет коммутировать цепи переменного тока только в моменты, когда питающее напряжение переходит через ноль. [20] Таким образом, для данной схемы ошибка управляющего сигнала микроконтроллера является систематической, и не вносит изменений в работу схемы.

К достоинствам диммера с пропуском полупериодов (полуволн) относят:

1. Простоту реализации
2. Стоимость устройств
3. Возможность управлять мощностью электронагревательных приборов

К недостаткам относят:

1. Невозможность управлять яркостью осветительных приборов из-за скачков напряжения даже с учетом реализации алгоритма выборочного пропуска полуволн. Даже лампы накаливания будут мерцать, светодиодные лампы придут в негодность в очень короткий срок.

Итог

Самым удачным и перспективным решением по части диммирования стоит признать схемы фазового регулирования с отсечкой по заднему фронту (спаду). Только диммеры по заднему фронту (спаду) способны стабильно работать с диммируемыми светодиодными лампами, не нагружая их блоки питания. Таким образом, в данной дипломной работе освещается процесс создания устройства управления приборами сети 220В фазовым регулированием с отсечкой по заднему фронту (спаду).

2.3 Разработка аппаратной части умного диммера

2.3.1 Электрическая схема устройства

В основу электрической схемы устройства положен принцип фазового регулирования мощности нагрузки с отсечкой по заднему фронту. Ранее подробно рассказывалось о разных принципах диммирования.

На рисунке 2.8 представлен сборочный чертеж электрической схемы диммера.

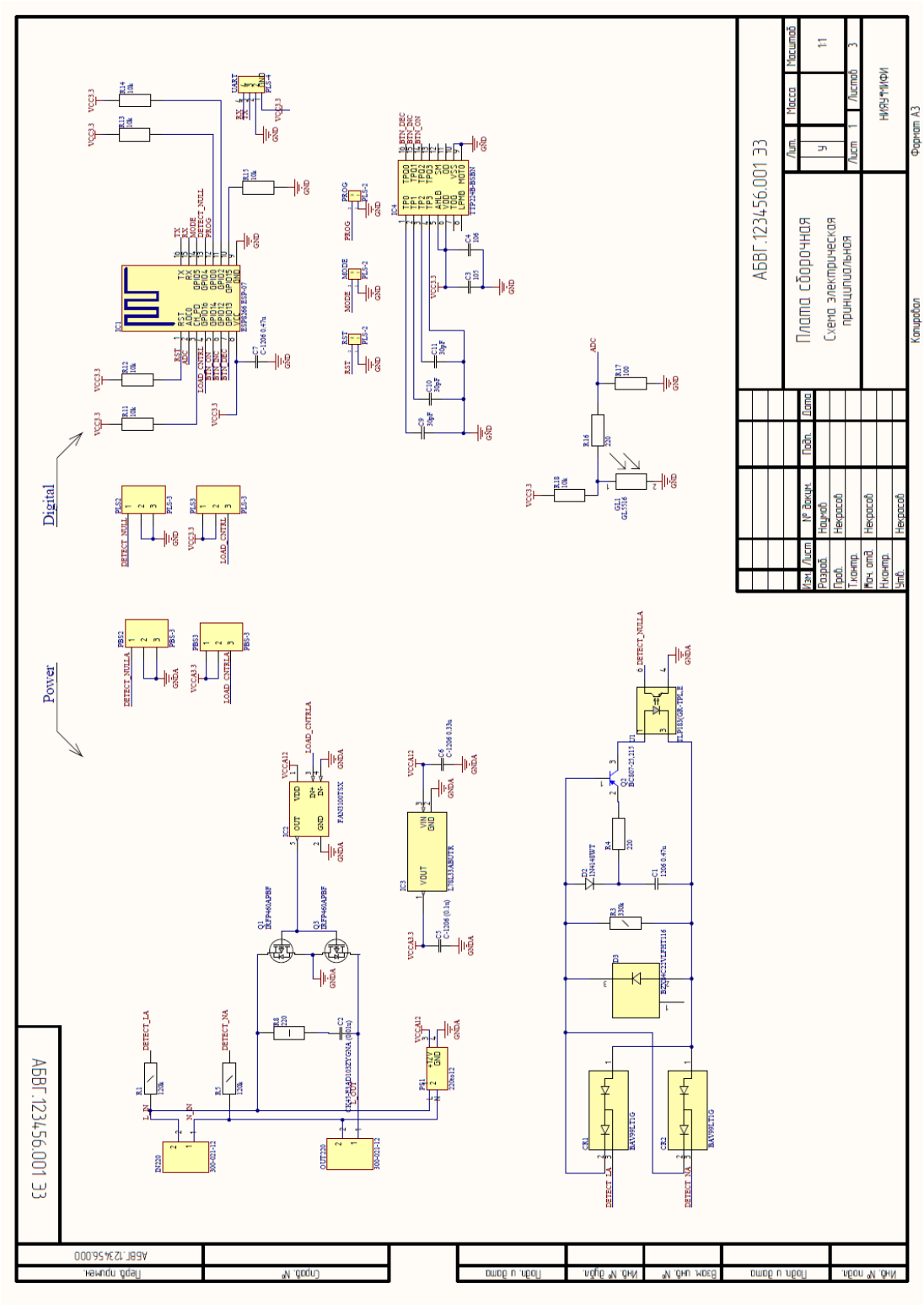

Рисунок 2.8 – Сборочный чертеж электрической схемы Диммера

Схему устройства можно главным образом разделить на силовую и цифровую части. В силовой части находятся высоковольтные элементы цепи, осуществляющие непосредственное взаимодействие с напряжением сети и с выводами на нагрузку. Цифровая часть устройства содержит элементы под

напряжением 3.3В, отвечающие за сбор, обработку, хранение и передачу информации. Стоит отметить, что силовая и цифровая части гальванически развязаны для безопасности устройства и уменьшения помех. Основными элементами силовой части являются мощные MOSFET транзисторы. Основным элементом цифровой части выступает модуль ESP-07 на основе микроконтроллера ESP8266. Ниже будут рассмотрены детально обе части электрической схемы и их взаимодействие.

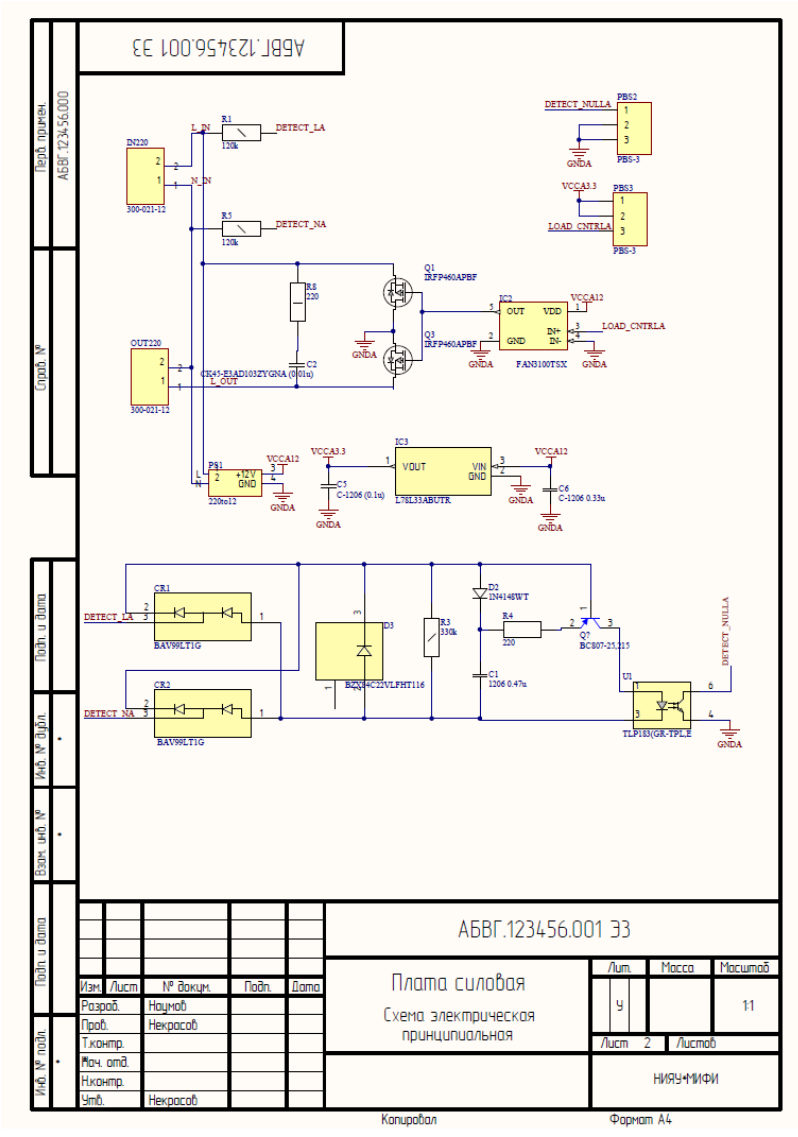

Рисунок 2.9 – Чертеж электрической схемы силовой части

Силовая часть схемы устройства представлена на рисунке 2.9 и содержит схему питания 12В и 3.3В, схему силового ключа и схему детектора перехода сетевого напряжения через ноль.

Схема детектора нуля подает импульс на выход DETECT_NULLA, когда напряжение в сети пересекает отметку в 22В. Микроконтроллер получает сигнал о начале синусоиды напряжения сети и подает сигнал на вход ключа LOAD_CNTRLA. Происходит коммутация нагрузки. Микроконтроллер держит высокий уровень сигнала на входе ключа согласно установленному уровню диммирования. Далее ключ закрывается, и нагрузка отключается от сети.

Детектор перехода через ноль работает по следующему принципу. Выпрямленное напряжение сети с выходов диодного моста обрезается стабилитроном и заряжает конденсатор C1 до напряжения стабилитрона 18-22В. Когда амплитуды напряжения сети оказывается меньше 22В, переход база-эмиттер смещается в прямом направлении и транзистор Q2 открывается. Далее, происходит разряд конденсатора C1 через токоограничивающее сопротивление R4 и транзистор Q2 на светодиод оптопары. На выходе оптопары наблюдается сглаженный импульс от 3.3 В до 0 В.

Принцип работы силового ключа основан на возможности MOSFET транзистора пропускать ток в обратном направлении. Таким образом удается построить ключ переменного тока, если соединить MOSFET транзисторы затворами и истоками. [21] Когда один транзистор пропускает ток в прямом направлении, потому что он открыт, другой транзистор пропускает ток в обратном направлении. На следующем полупериоде транзисторы поменяются ролями. Драйвер затвора обеспечивает гальваническую развязку цепей, а также большой ток, необходимый для быстрого открытия транзисторов, и напряжение на затворе 10В для низкого сопротивления канала. [22]

Цифровая часть схемы устройства представлена на рисунке 2.10 и включает в себя модуль ESP-07 на базе микроконтроллера ESP8266EX, необходимую схему для введения микроконтроллера в режим установки программного обеспечения (программации), схему подключения фоторезистора, схему сенсорных кнопок и разъемы для взаимодействия с

силовой частью устройства и подключения к ПК для установки программного обеспечения.

Схема сенсорных кнопок выполнена на базе микросхемы TTP224. И представляет собой три конденсатора C9, C10, C11а на 30 пФ подключенных к входам микросхемы и параллельно подключенные сенсорные площадки, прикасаясь к которым, человеческий палец увеличивает емкость между землей и входом микросхемы. [23] Конденсаторы C9, C10, C11 необходимы для настройки чувствительности сенсорных кнопок и выступают в роли минимальных значений емкостей между землей и входами микросхемы.

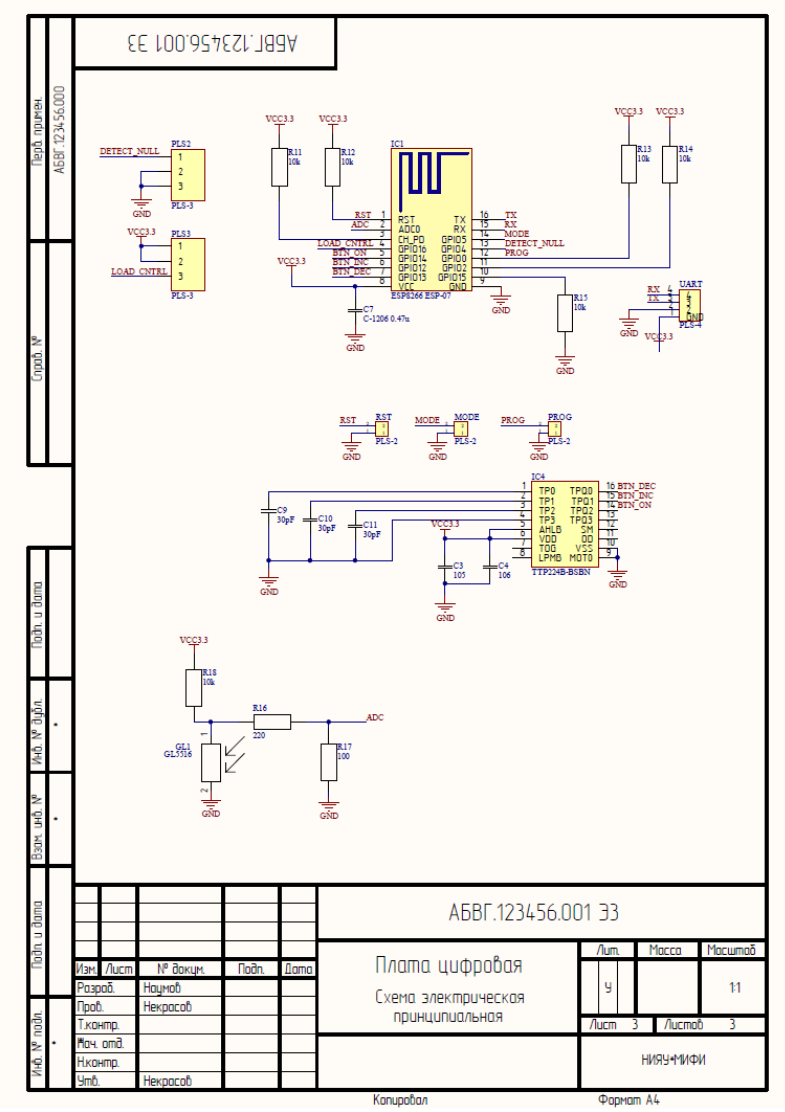

Рисунок 2.10 – Чертеж электрической схемы цифровой части

Силовая и цифровые части находятся на разных платах и соединяются про мощи разъемов.

2.3.2 Подбор компонентов

Подбор компонентов осуществлялся с учетом электрических характеристик, описанных в документациях к компонентам. Особое внимание было уделено силовой части схемы из-за наличия высоковольтных цепей. Силовая часть состоит из схемы детектора нуля, схемы питания и схемы силового ключа.

Подбор компонентов для схемы детектора нуля

Схема симуляции работы для подбора компонентов детектора нуля представлена на рисунке 2.11.

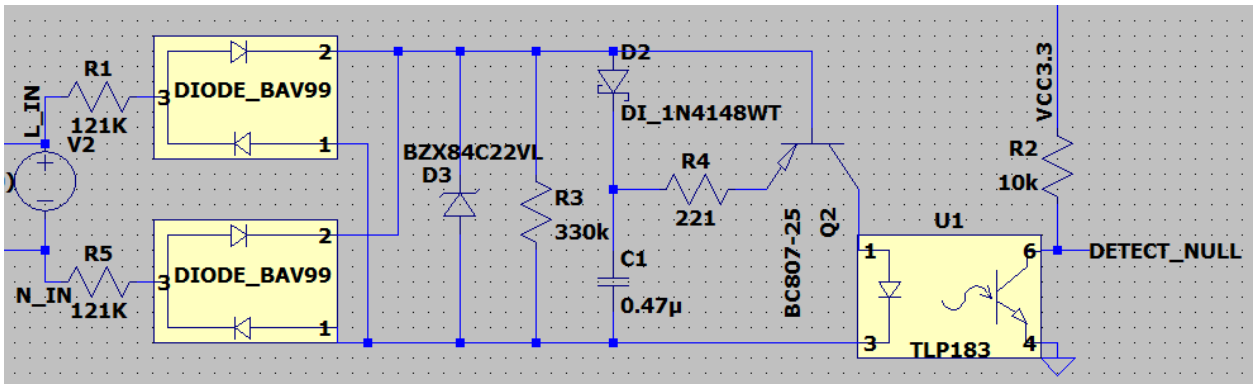

Рисунок 2.11 – Схема симуляции работы детектора нуля

Гасящие резисторы R1 и R5 на 121 кОм, на которых падает по половине сетевого напряжения должны рассеивать на себе мощность 0.1 Вт.

$$P_{R1} = \frac{U^2}{R1} = \frac{110^2}{121 \cdot 10^3} = 0.1\text{Вт}$$

Были использованы SMD резисторы типоразмера 1206 на 0.25 Вт.

Резистор R3 на 330 кОм при напряжении пробития стабилитрона 30В должен рассеивать максимальную мощность 2.7 мВт.

$$P = \frac{U^2}{R} = \frac{30^2}{330 \cdot 10^3} = 2.7\text{ мВт}$$

Был использован также резистор типоразмера 1206.

Резистор R4, использующийся для ограничения пикового тока светодиода оптопары на 221 Ом, аналогично был выбран типоразмера 1206.

Конденсатор C1 на 0.47 мкФ был взят типоразмера 1206 на 50В с запасом от напряжения пробития стабилитрона 30 В.

В качестве диода D2 использовался 1N4148WT с напряжением пробоя 80 В.

В качестве транзистора Q2 использовался BC807-25 с максимальной мощностью рассеивания 250 мВт. На рисунке 2.12 представлен график рассеиваемой мощности на транзисторе Q2 в ходе моделирования схемы. Наблюдается двукратный запас по мощности.

Рисунок 2.12 – График рассеиваемой мощности на транзисторе BC807-25

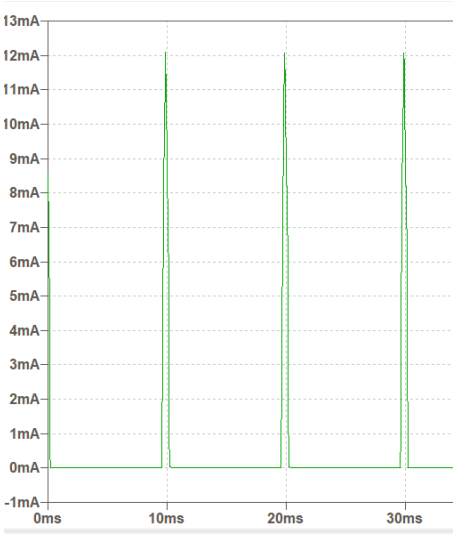

Рисунок 2.13 – График пикового тока светодиода оптопары TLP183

В качестве оптопары U1 была использована TLP183 с максимальным током светодиода 50 мА. На рисунке 2.13 представлен график пикового тока светодиода в ходе моделирования схемы. Наблюдается 4 кратный запас по току.

Подбор компонентов для схемы силового ключа

Для подбора компонентов схемы силового ключа использовалась симуляция работы силового ключа. Схема для симудяции работы представлена на рисунке 2.14.

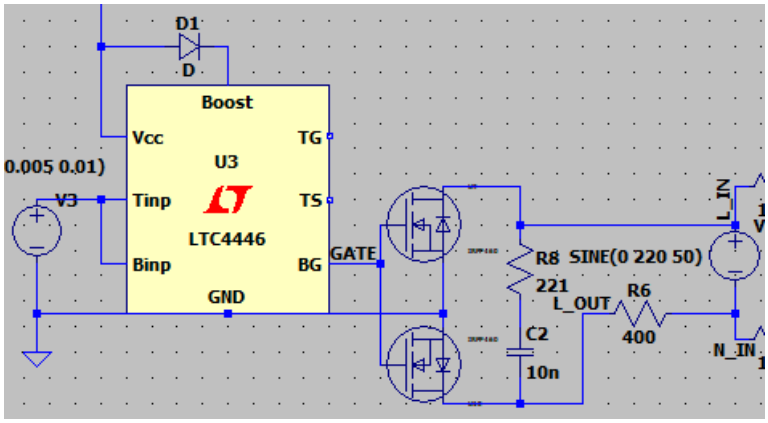

Рисунок 2.14 – Схема симуляции работы силового ключа

В качестве транзисторов Q1 и Q3 использовались MOSFET транзисторы IRFP460APBF. Рассчитаны на максимальное напряжение сток-исток 500В, затвор-исток = 20В и на рассеиваемую мощность 280 Вт. Главной характеристикой этих транзисторов можно считать низкий заряд затвора Qg = 105 нКл. Минимальное время открытия этого транзистора 55 нс.

Таким образом, чтобы открыть транзисторный ключ за время порядка 55нс использовался драйвер затвора способный дать ток порядка 1-2 А. В качестве драйвера затвора использовался FAN3100TSX, обеспечивающий ток 2А при напряжении 12 В.

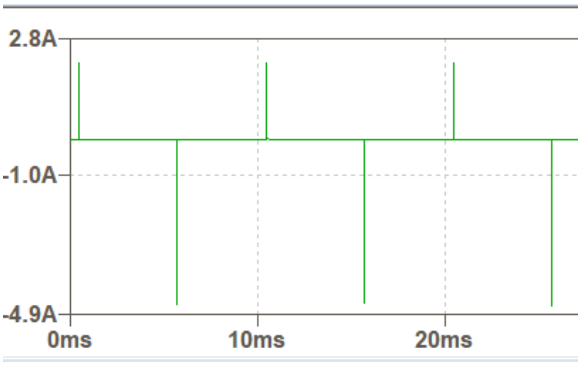

Рисунок 2.15 – Временная характеристика тока затвора транзиторов IRFP460A

Подбор компонентов для схемы цепи питания

В цепи питания использовались блок питания на 12В и стабилизатор напряжения на 3.3В. Блок питания представляет из себя отдельное устройство, встраиваемое в схему. Стабилизатором напряжения выступает

L78L33ABUTR. Напряжение 12 В необходимо для питания драйвера силового ключа, а напряжение 3.3 В для цифровой части устройства.

Подбор микроконтроллера для цифровой части устройства

В качестве усправляющего устройства был выбран модуль ESP-07. Внешний вид модуля показан на рисунке 2.16. Модуль включает в себя систему на кристале ESP8266EX на базе 32-битного миркопроцессора Tensilica L106 32, SPI flash-память 1МБ и керамическую антенну. Стоит выделить следующие особенности микроконтроллера ESP8266

Рисунок 2.16 – Модуль ESP-07

- Поддержка беспроводного стандарта 802.11 b/g/n
- Поддержка 2 режима работы Wi-Fi Direct (P2P), soft-AP
- Интегрирован стек протокол TCP/IP
- Интерфейсы SDIO 1.1/2.0, SPI, UART
- Энергопотребление в режиме ожидания <1.0мВт.
- Тактовая частота — 80 МГц
- Возможность программации по Wi-Fi
- Файловая система SPIFFS [24]

Модуль ESP-07 на базе микроконтроллера ESP8266EX полностью подходит для реализации функционала умного диммера.

2.3.3 Моделирование схемы

Симуляция работы схемы проходила в программе LTSpice XVII. Помимо spice-моделей из стандартной библиотеки LTSpice XVII в схеме использовались spice-модели подобранных компонентов либо их ближайших аналогов. Цифровой сигнал с микроконтроллера замещен источником прямоугольных импульсов амплитудой 3.3 В. В качестве нагрузки был выбран резистор 400 Ом. Особое внимание было уделено силовой части схемы из-за наличия высоковольтных цепей. Способность работы силовой части схемы оценивалась из возможности диммирования нагрузки с помощью схемы силового ключа и наличия выходного сигнала от схемы детектора перехода напряжения сети через ноль. На рисунке 2.17 представлена схема для симуляции силовой части устройства.

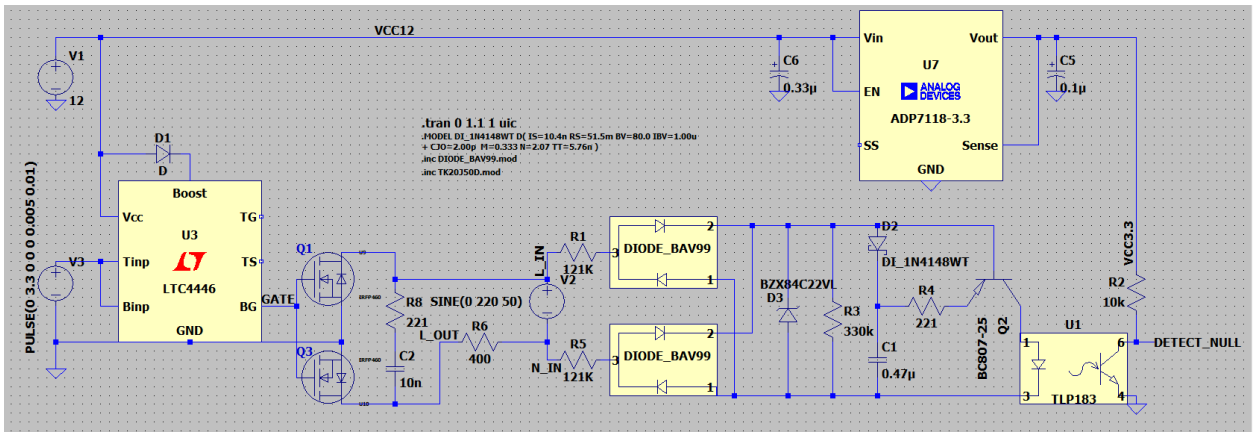

Рисунок 2.17 – Схема симуляции силовой части

Временная характеристика напряжения на выходе оптопары сетевого напряжения представлена на рисунке 2.18.

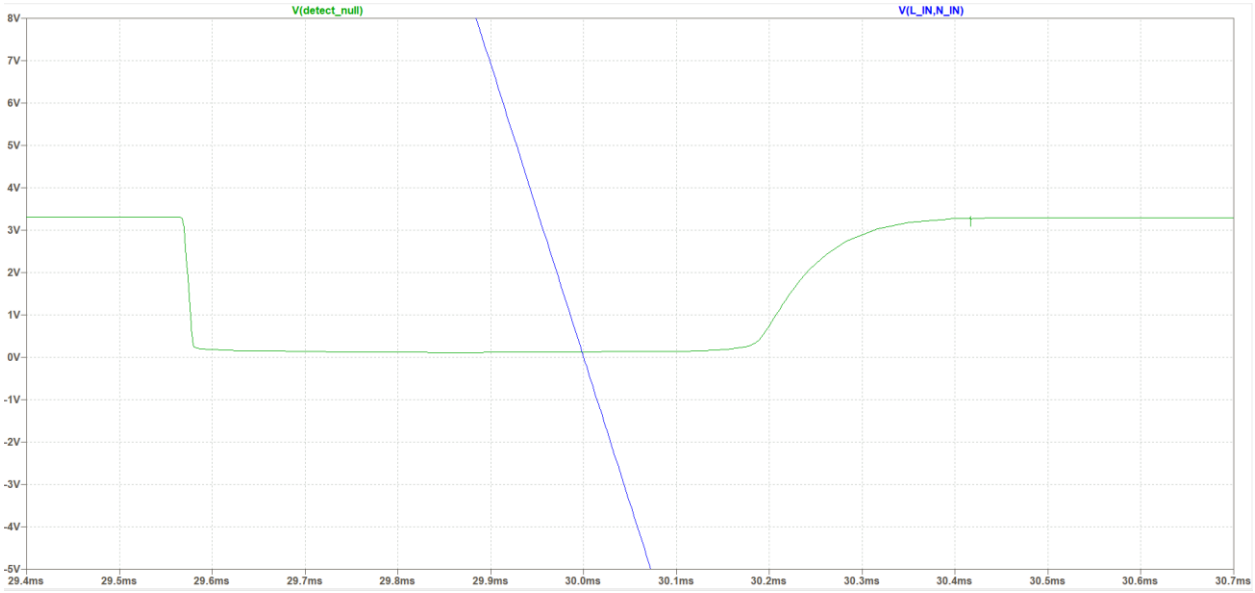

Рисунок 2.18 – Выходная временная характеристика напряжения на выходе детектора перехода через ноль

Зеленым цветом обозначен импульс с детектора перехода через ноль. Синим – сетевое напряжение. По характеристике видно, что импульс опережает переход синусоиды через ноль. Как и должно быть детектор срабатывает на напряжение около 18-22В (напряжение на стабилитроне). Оценка времени упреждающего срабатывания детектора нуля показана на рисунке 2.19.

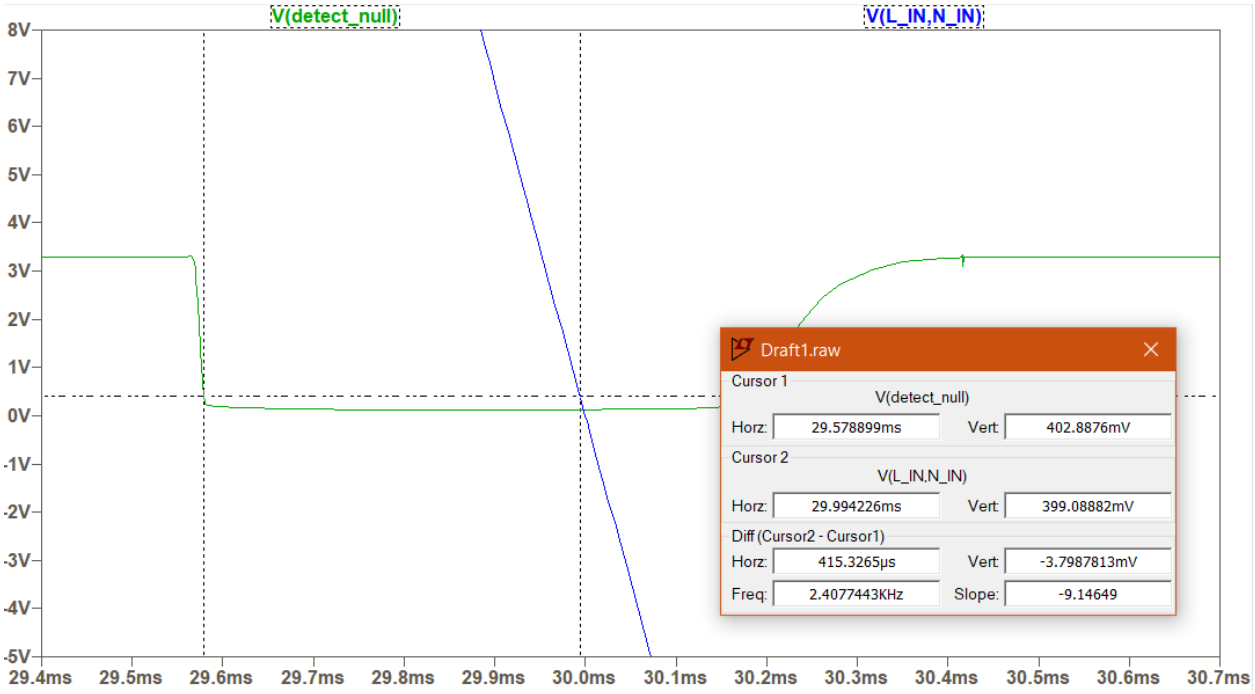

Рисунок 2.19 – Оценка времени упреждающего срабатывания детектора перехода через ноль

Время упреждающего срабатывания детектора $t \approx 0.4$ мс.

Временные характеристики напряжения на нагрузке при разных уровнях диммирования представлены на рисунках 2.20, 2.21, 2.22.

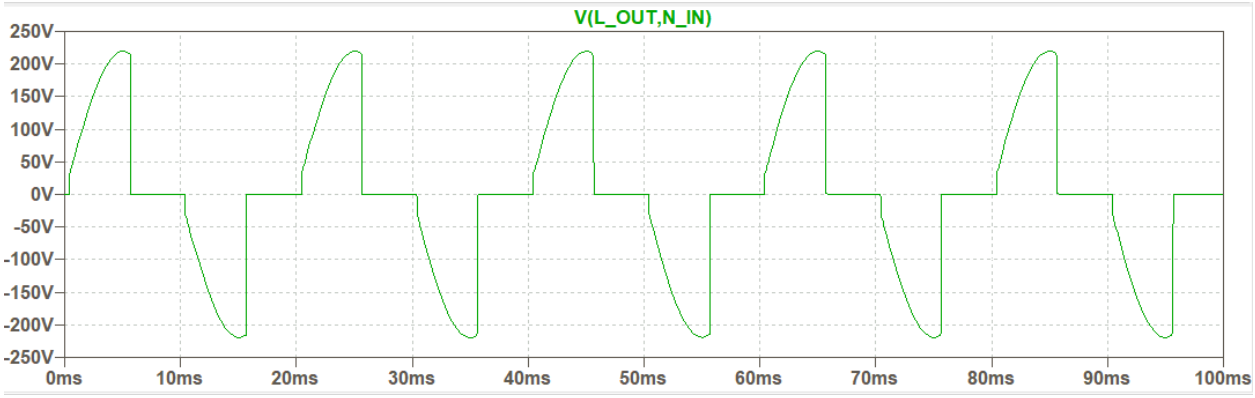

Рисунок 2.20 – Временная характеристика напряжения на нагрузке при уровне диммирования 50%

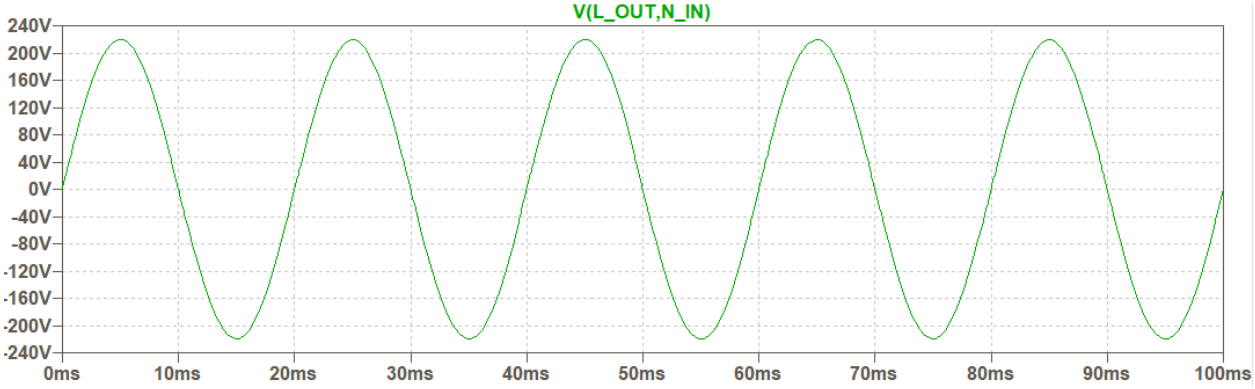

Рисунок 2.21 – Временная характеристика напряжения на нагрузке при уровне диммирования 100%

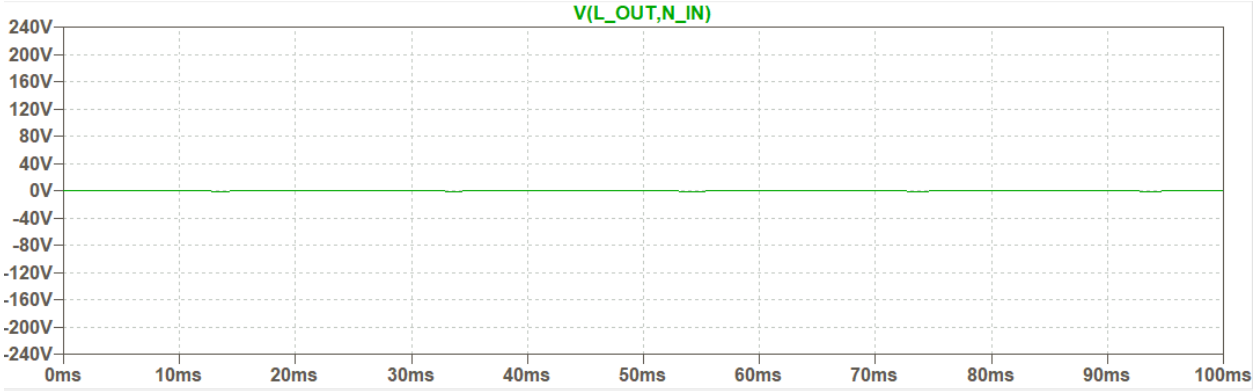

Рисунок 2.22 – Временная характеристика напряжения на нагрузке при уровне диммирования 0%

Характеристики полностью соответствуют фазовому регулированию по заднему фронту (спаду). Таким образом, можно сделать вывод о работоспособности предложенной электрической схемы силового ключа для управления нагрузкой переменного тока.

2.3.4 Разработка печатной платы

Проектирование печатной платы осуществлялось при помощи современного САПР Altium Designer. Разработка состояла из нескольких этапов:

- Сборка электрической схемы
- Создание правил проектирования design rules
- Размещение компонентов
- Размещение проводников
- Проверка соответствия правилам Design Rule Check
- Создание GERBER

Самым важным этапом является создание правил проектирования. Для грамотного задания правил все цепи были разделены на три класса: класс цифровых цепей, класс силовых цепей, класс силовых цепей под высоким напряжением (высоковольтных цепей). Для проводников трех классов цепей созданы соответствующие правила по толщине, длине. Для соблюдения норм по гальваническим и электрическим зазорам между высоковольтными и низковольтными цепями были добавлены соответствующие правила проектирования, касающиеся зазоров. Цепи питания и земли были разведены с помощью полигонов для уменьшения сопротивления проводников. Некоторые высоковольтные цепи также были разведены при помощи полигонов для уменьшения нагрева.

Исходя из соображений места установки устройства, например, стандартную квартирную монтажную коробку, было принято решение разделить функциональность устройства между двумя печатными платами верхнего и нижнего уровней. Плата нижнего уровня была отведена под силовую часть устройства, в то же время плата верхнего уровня включила в себя цифровую часть устройства. В предыдущих разделах описывались схемы, из которых состоит силовая и цифровая части устройства. Плата верхнего уровня также включает элементы сенсорного управления устройства. Это

значит, что необходимо располагать все элементы за исключением сенсорных площадок на нижнем слое печатной платы с целью облегчения контакта пользователя с элементами непосредственного управления устройством. Обе платы разрабатывались в едином технологическом процессе для обеспечения точного расположения разъемов, служащих для последующего соединения плат параллельно. Послойные чертежи плат представлены на рисунках 2.23 – 2.27.

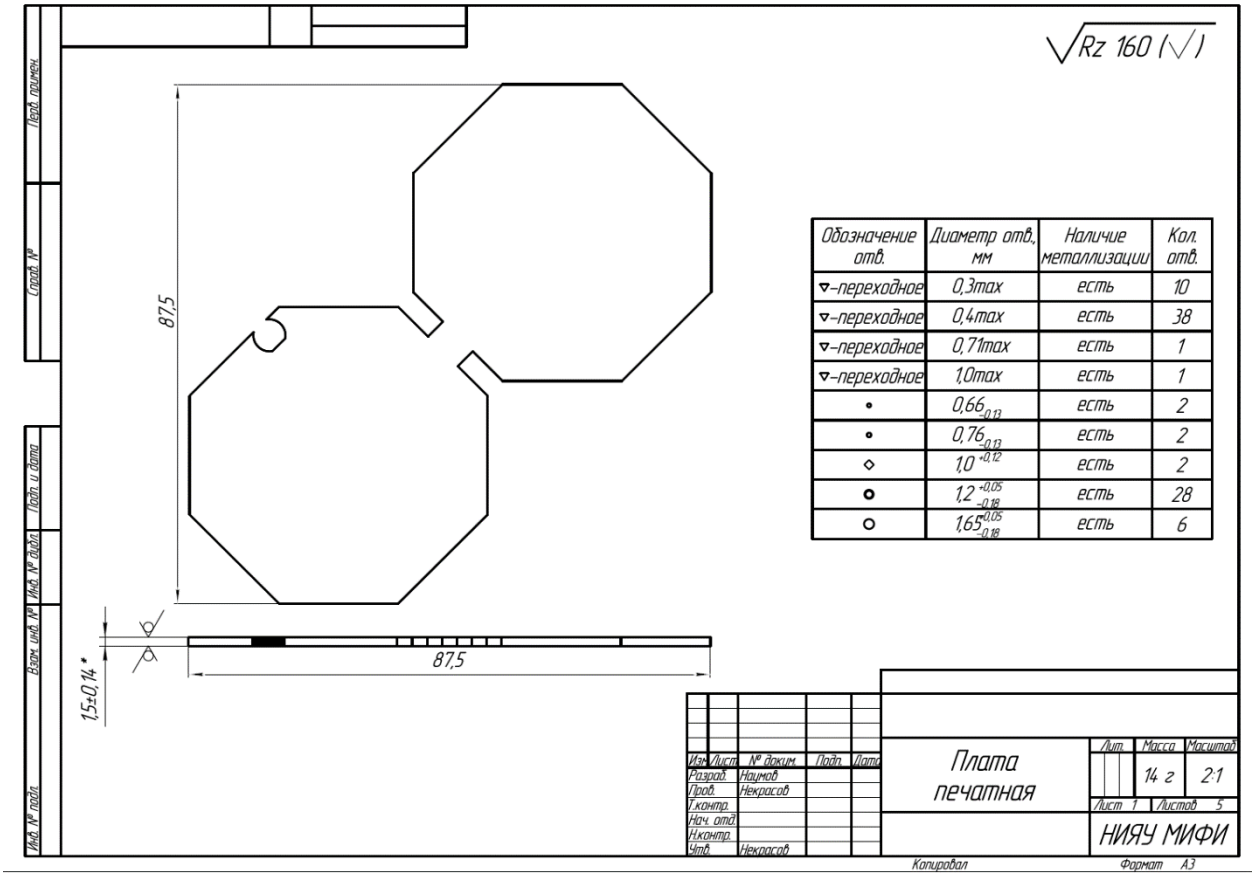

Рисунок № 2.23 – Основной чертеж печатной платы устройства

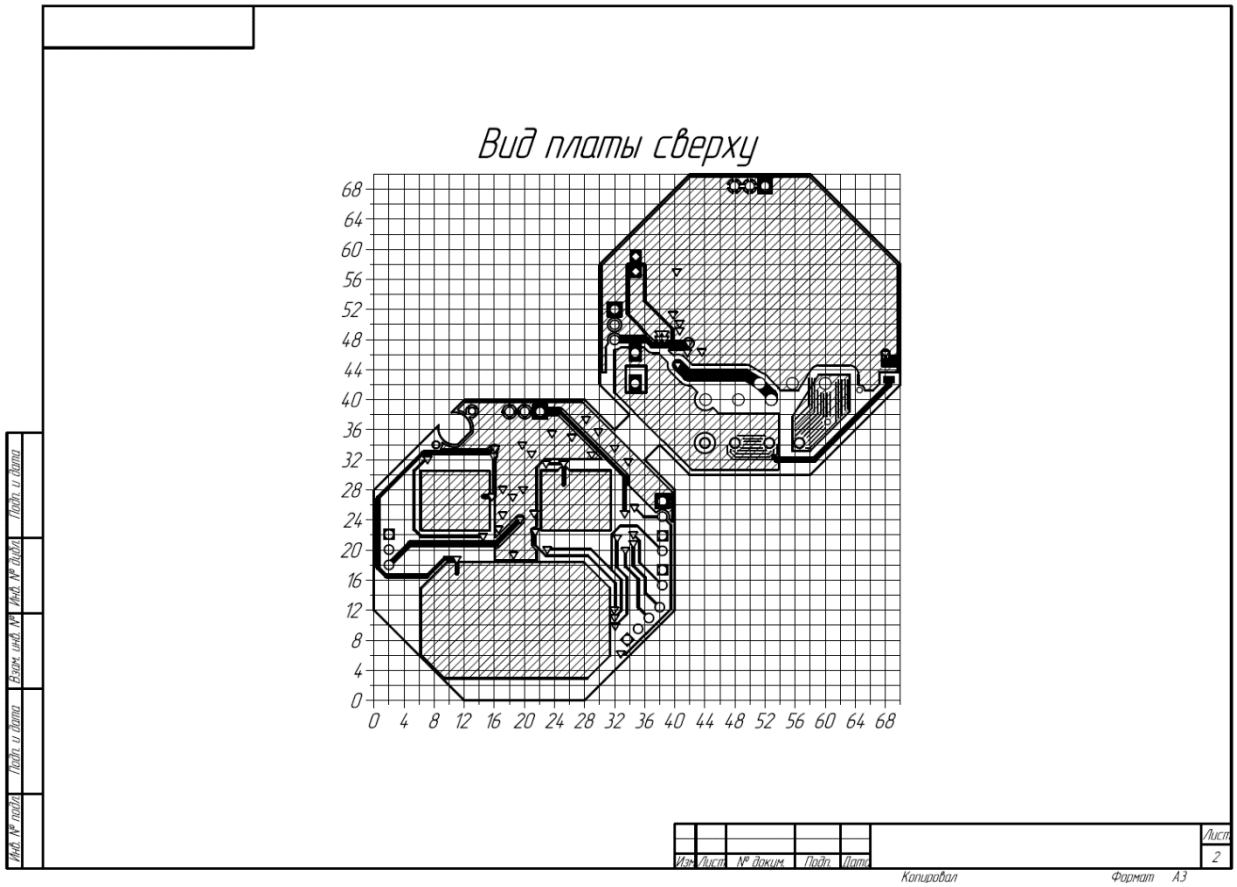

Рисунок № 2.24 – Вид печатной платы устройства сверху

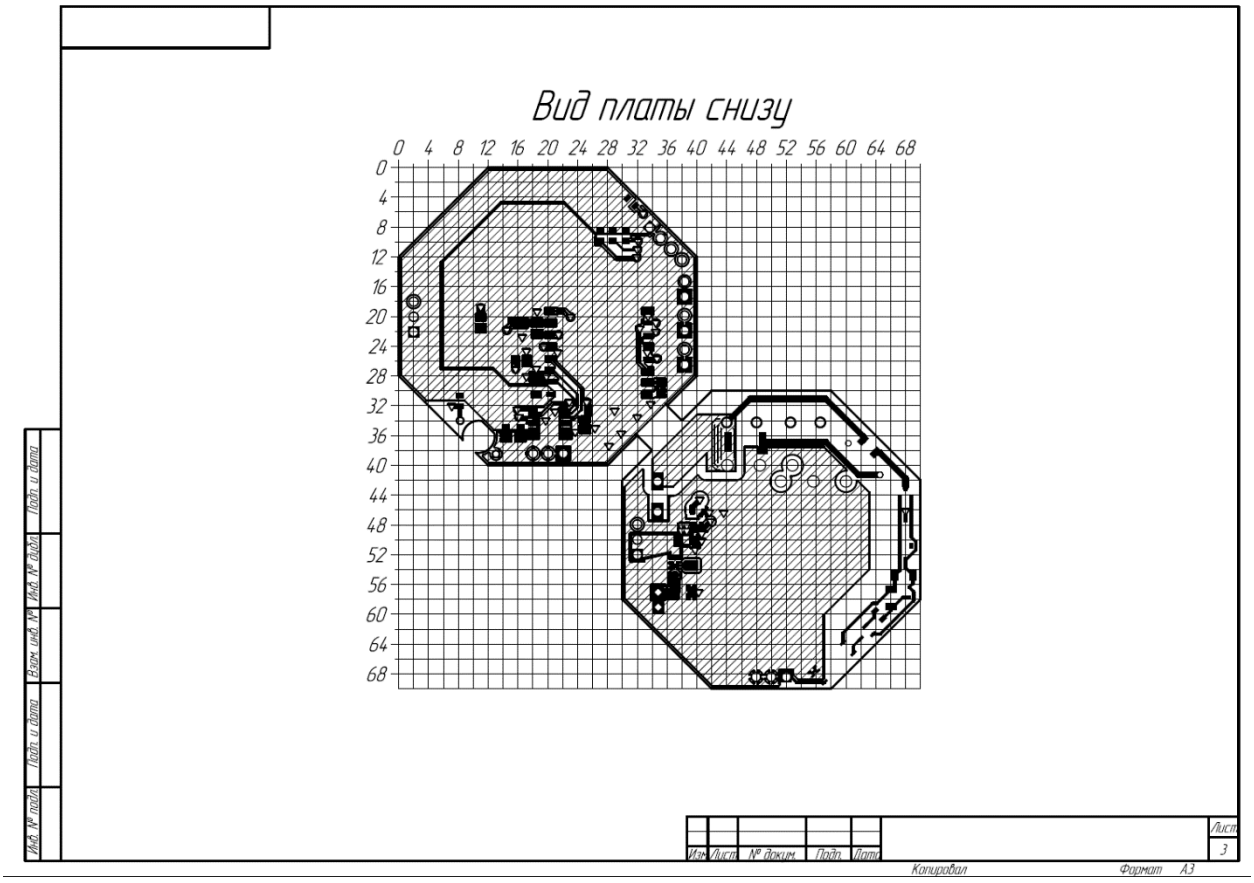

Рисунок № 2.25 – Вид печатной платы устройства снизу

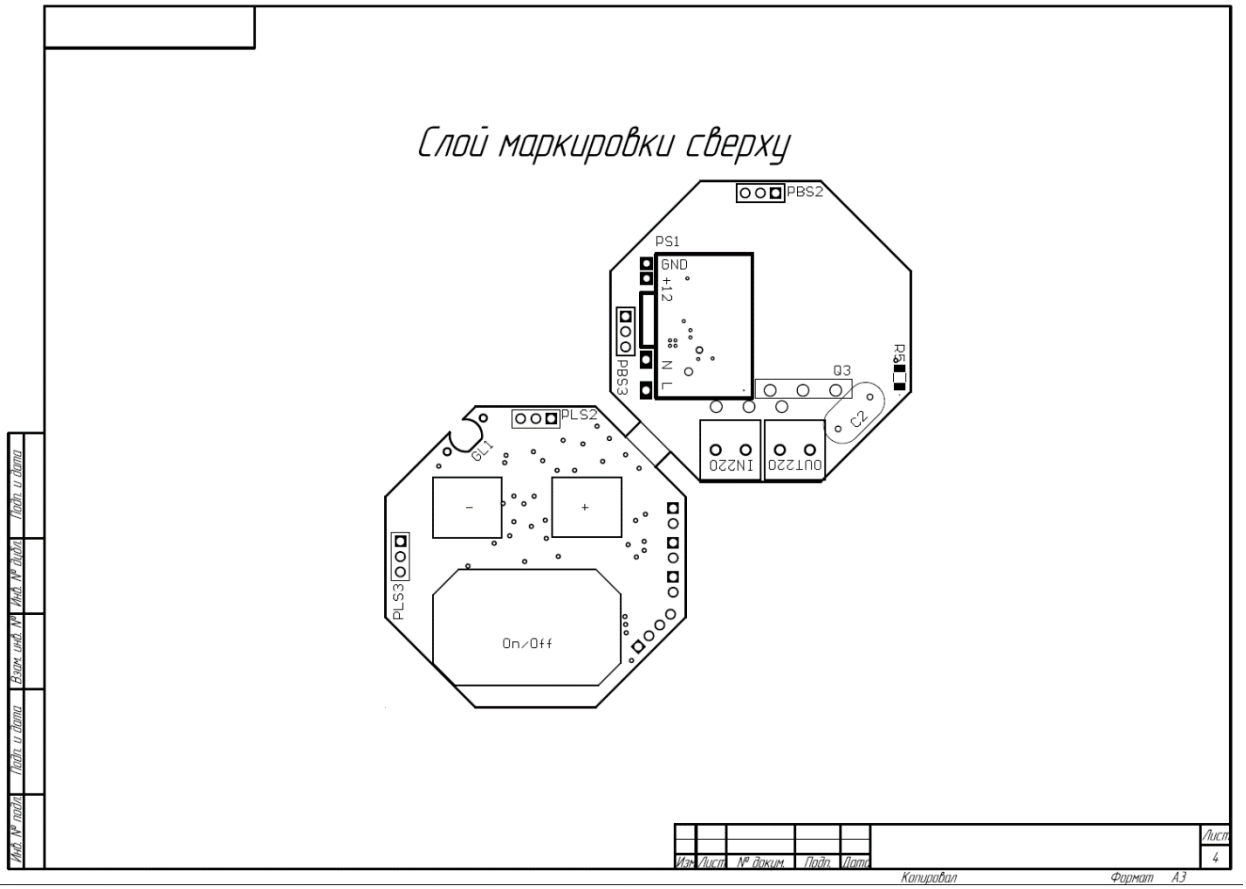

Рисунок № 2.26 – Слой маркировки печатной платы устройства сверху

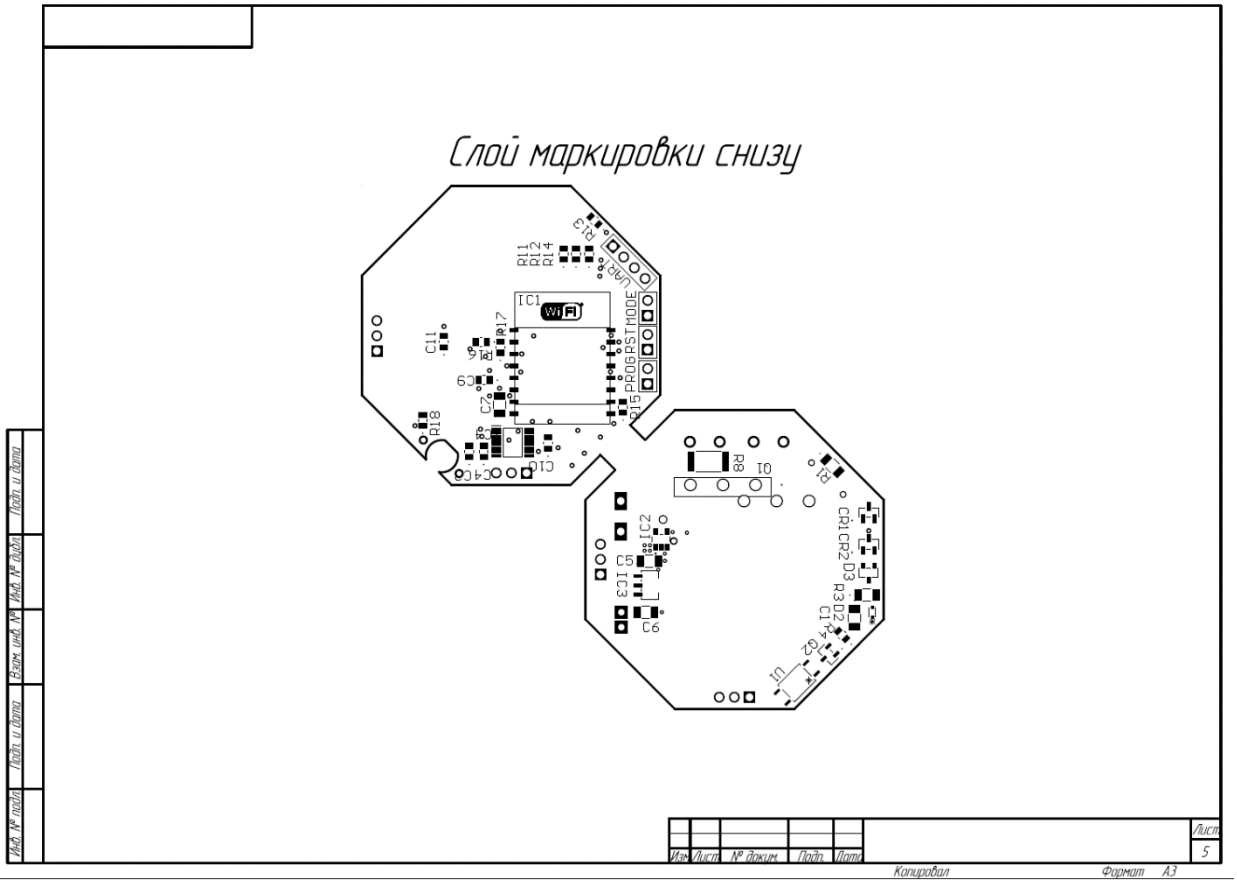

Рисунок № 2.27 – Слой маркировки печатной платы устройства снизу

Учитывая размеры гнезд на платах для их соединения и чертежи плат можно сделать вывод, что устройство не превышает по габаритам стандартный квартирный подрозетник. Это значит, что удалось компактно и в то же время технологочески верно провести трассировку сборочной печатной платы устройства.

Выводы

Разработка аппаратной части умного свето-регулятора (диммера) прошла через много этапов: выбор технологии, создание электрической схемы, симуляция ее работы, проектирование печатной платы. Таким образом, удалось создать первый прототип устройства, обладающего очевидными достоинствами, такими как: удачный подбор компонентов, возможность регулярно обновлять програмное обспечение, малый нагрев, гальваническая развязка цифровой и силовой частей схемы, малые габариты, удачный выбор расположение элементов управления, возможность использовать с разными типами ламп, в том числе со светодиодными диммируемыми лампами. В общем, результаты работы над аппаратной частью умного диммера завершились успехом.

3. Программная часть

3.1 Общая структура

Программное обеспечение киберфизической системы «Умный дом» строится на основе программного обеспечения ее отдельных узлов с использованием разных технологических решений. Узлы киберфизической системы «Умный дом» определяются согласно структуре, предложенной в Главе №1.

Для написания программного кода узлов киберфизической системы используются следующие языки программирования. Для сервера-шлюза, удаленных серверов авторизации и приложений, вычислительного сервера и прочих используются соответственно серверные языки программирования, такие как C#, Python, Java, PHP, Ruby, Perl, GoLang. Для разработки программного обеспечения для умного устройства традиционно используются языки C, C++, Python. Для разработки пользовательских интерфейсов используются C#, Java, Kotlin, Swift, JavaScript.

В рамках реализации комплексной киберфизической системы «Умный дом» сотрудниками и студентами кафедры №3 был утвержден следующий набор технологий:

- Для умных устройств использовать языки C, C++ с набором библиотек, подходящих для реализации общения устройств по Wi-Fi
- Для сервера-шлюза использовать Python, а также библиотеки для реализации серверного функционала и общения с умными устройствами по Wi-Fi
- В качестве сервера авторизации, сервера приложений и вычислительного сервера использовать сторонние облачные решения платформы Firebase компании Google такие, как: Firebase Auth, Firebase Hosting, Realtime Database.

- Для пользовательского Web-интерфейса использовать язык JavaScript и набор библиотек, ускоряющих процесс разработки.
- Для Andriod приложения использовать язык Java или Kotlin с набором библиотек для экосистемы Android

Связь конкретных узлов разрабатываемой системы показан на рисунке 3.1.

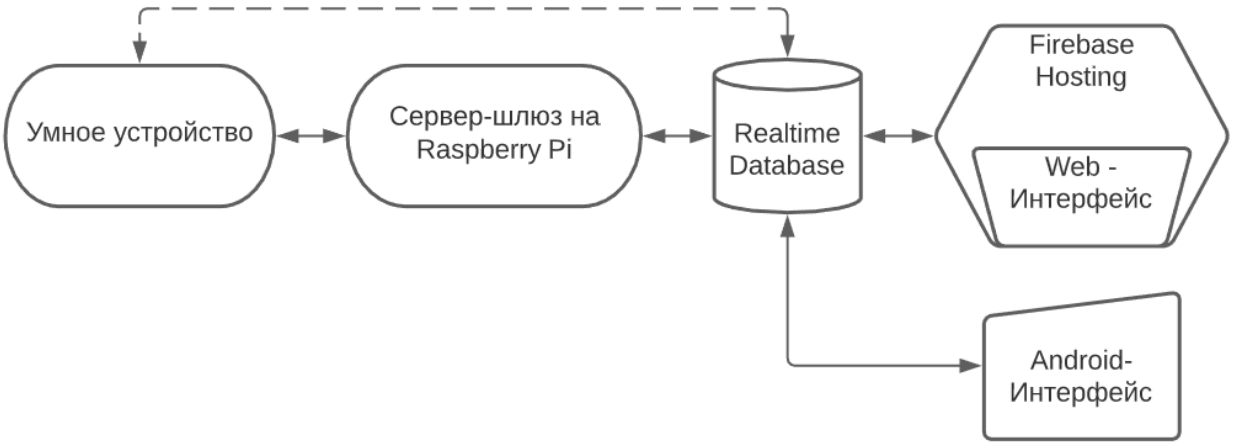

Рисунок 3.1 – Структурная схема «Умный дом», реализуемая на кафедре №3

В рамках дипломной работы реализовывалась часть программного обеспечения киберфизической системы «Умный дом», а именно программное обеспечение для умного диммера (свето-регулятора) и Web-приложение в качестве пользовательского интерфейса. Далее будут подробно изложены этапы написания программного кода проектов.

3.2 Программное обеспечение умного диммера

Для осуществления задач управления нагрузкой переменного тока, обмена данными по Wi-Fi и прочего функционала умного диммера использовался модуль ESP-07, построенный на базе микроконтроллера ESP8266EX, особенности которого были отмечены в разделе 2.3.2 «Подбор компонентов». Код программного обеспечения для микроконтроллера ESP8266 и будет представлять собой программную часть умного диммера.

Для решения поставленных задач программный код для диммера был написан на C++11. Программирование микроконтроллера осуществлялось в среде Arduino IDE с использованием библиотек Arduino для ESP8266. Использование высокоуровневого языка программирования C++ позволило следовать классическим парадигмам объектно-ориентированного программирования: инкапсуляция, наследование, полиморфизм. Код был написан на основе следующих принципов:

- разделение функционала на классы
- создание библиотек на классах
- методы разных классов, выполняющие схожие задачи, называются одинаково
- передача функционала между экземплярами класса осуществляется при помощи функций обратного вызова или ссылок на экземпляры классов.

Согласно требованиям, предъявляемым к умному диммеру и изложенным в разделе № 2.1 «Общая структура», следующий функционал должен быть реализован программным способом:

- осуществлять близкую к линейной регулировку мощности на нагрузке
- измерять относительную освещенность комнаты для корректирования мощности на нагрузке в автоматическом режиме
- предоставлять пользователю управление непосредственно при помощи кнопок на лицевой панели.
- конфигурация и работа в системе «Умный дом»

Далее детально будет рассмотрена программная реализация предъявляемых требований к функционалу устройства.

Регулировка мощности на нагрузке

Для программной реализации регулировки мощности на нагрузке был создан класс Dimmer со интерфейсом, представленным на рисунке 3.2.

```
class Dimmer {
    public:
        Dimmer(int inLoadPin, int inNullDetectedPin);
        int loadPin;                    //пин выхода на нагрузку
        int nullDetectedPin;            //пин входа с детектора перехода через ноль
        void interrupt();               //установка флага перехода через ноль
        void watch();                   //управление сигналом на выходе
        void setLevel(int inLevel);     //установка уровня диммирования
        void increase();                //увелечинение уровня диммирования
        void decrease();                //уменьшение уровня диммирования
        void on();                      //включение диммера
        void off();                     //выключение диммера
        void toggle();                  //переключение диммера
        bool isOn();                    //проверка включен ли диммер
        void setup(void (*dtct)());     //конфиграция диммера
        int getLevel();                 //получение уровня диммера
        unsigned long getMicrosLevel(); //получение времени обрезания синусоиды
        void resume();                  //возобновить детекцию нуля
        void pause();                   //приостановить детекцию нуля

    private:
        bool _nullDetected;             //флаг перехода через ноль
        bool _state;                    //состояние диммера (вкл/выкл)
        unsigned long _microsLevel;     //время обрезания синусоиды
        int _level;                     //уровень диммирования 0 - 100
        unsigned long _startMicros;     //время перехода через ноль
        void (*_detect)();              //прерывание при переходе через ноль
        static unsigned long _getMicrosByLevel(int level); //вычисление времени
                                                           //обрезания синусоиды
                                                           //по уровню диммирования
};
```

Рисунок 3.2 – Интерфейс класса Dimmer

Основными методами класса Dimmer, осуществляющего диммирование, являются методы watch, setLevel и _getMicrosByLevel. Метод watch вызывается в главном цикле микроконтроллера (как и все методы других классов с таким же названием). Программный код метода watch и Блок схема, иллюстрирующая работу метода, представлены на рисунках 3.3, 3.4. При переходе через ноль устанавливается высокий уровень сигнала на выходе микроконтроллера. По прошествии некоторого времени, называемого временем обрезания синусоиды, на выходе устанавливается низкий уровень сигнала, и коммутация нагрузки прекращается.

```
void Dimmer::watch() {
    int loadState = HIGH;

    if (!isOn()) {
        digitalWrite(loadPin, LOW);
        _nullDetected = false;
        return;
    }

    if (_nullDetected) {
        _startMicros = micros();
        _nullDetected = false;
    }

    const unsigned long currMicros = micros();

    if (currMicros - _startMicros >= _microsLevel) {
        loadState = LOW;
    }

    digitalWrite(loadPin, loadState);
}
```

Рисунок 3.3 – Программный код метода watch класса Dimmer

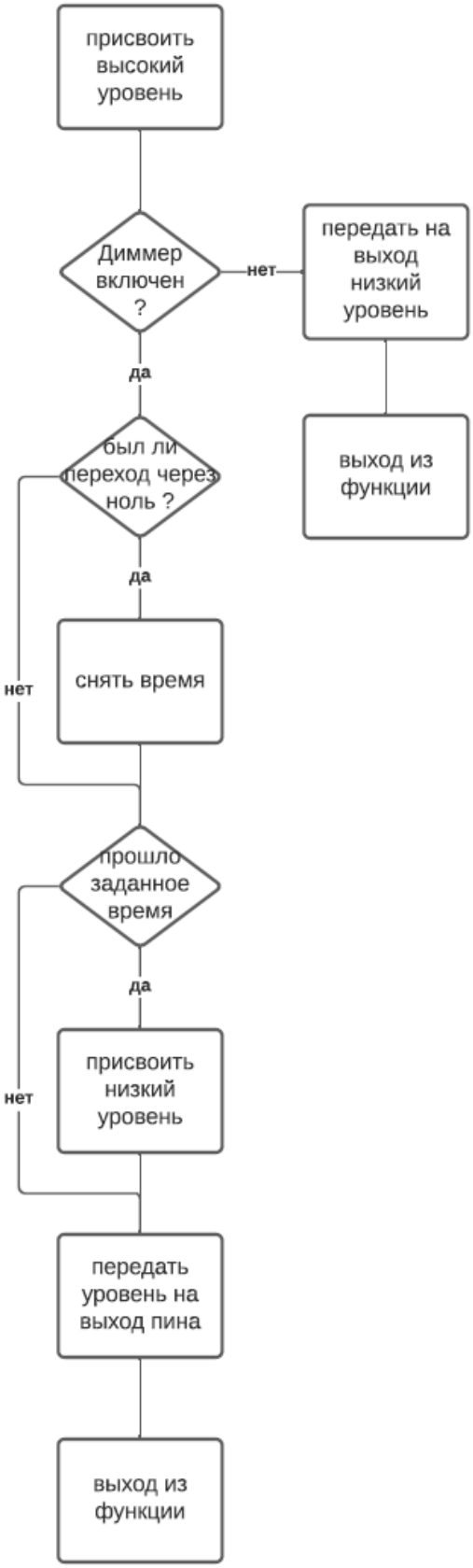

Рисунок 3.4 – Блок-схема метода watch класса Dimmer

Метод setLevel осуществляет задание уровня диммирования и времени до обрезания синусоиды. Программный код метода и блок схема изображены на рисунках 3.5, 3.6.

```
void Dimmer::setLevel(int inLevel) {
    if (!isOn()) {
        return;
    }

    _level = inLevel;
    _microsLevel = _getMicrosByLevel(_level);
}
```

Рисунок 3.5 – Программный код метода setLevel класса Dimmer

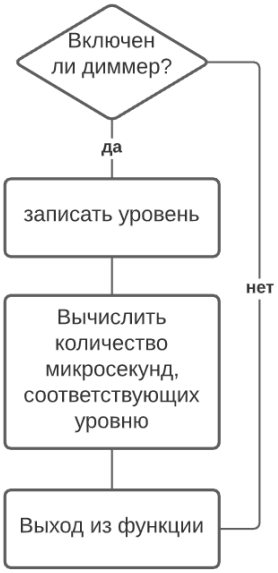

Рисунок 3.6 – Блок-схема метода setLevel класса Dimmer

Метод _getMicrosByLevel вычисляет время до обрезания синусоиды по уровню диммирования. Это значение времени используется в функции watch. Значения этого времени подобраны по следующей теории.

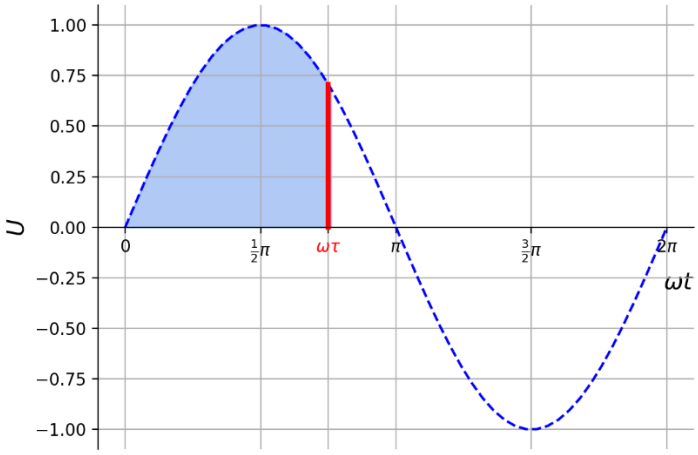

Рисунок 3.7 – Временная характеристика напряжения при диммировании по спаду

Обозначим за τ – время до обрезания синусоиды напряжения, а за L – относительный уровень диммирования и найдем функцию L(τ). В первом приближении положим, что относительный уровень диммирования L определяется отношением:

$$L(\tau) = \frac{S(\tau)}{S(T/2)},$$

где S – площадь под графиком функции напряжения от времени, T/2 – полупериод. S как функция времени определяется следующим выражением:

$$S(t) = U_m \int_0^{wt} \sin\varphi \ d\varphi,$$

Где w – циклическая частота напряжения сети и равна $2\pi\nu \approx 314{,}1593$ рад/с. Тогда

$$L(\tau) = \frac{\int_0^{w\tau} \sin\varphi \ d\varphi}{\int_0^{wT/2} sin\varphi \ d\varphi} = \frac{\int_0^{w\tau} \sin\varphi \ d\varphi}{\int_0^{\pi} sin\varphi \ d\varphi} = \frac{1-\cos w\tau}{\cos 0 - \cos \pi} = \frac{1-\cos w\tau}{2}.$$

L принадлежит отрезку [0, 1]. Для того чтобы расположить уровни диммирования линейно необходимо найти обратную функцию τ(L).

$$\tau(L) = \frac{\arccos(1-2L)}{w}.$$

С целью увеличения производительности были предварительно посчитаны значения τ для двадцати уровней диммирования L от нуля до единицы с шагом 0,05 согласно теории. Значения приведены в таблице 3.1.

Уровень диммирования (L`), %	**Уровень диммирования (L), отн. ед.**	**Время удержания синусоиды (τ), мкс.**
0	0	0
5	0,05	1436
10	0,1	2048
15	0,15	2532
20	0,2	2952
25	0,25	3333
30	0,3	3690
35	0,35	4030
40	0,4	4359
45	0,45	4681
50	0,5	5000

55	0,55	5319
60	0,6	5641
65	0,65	5970
70	0,7	6310
75	0,75	6667
80	0,8	7048
85	0,85	7468
90	0,9	7952
95	0,95	8564
100	1	10000

Таблица 3.1 – Время удержания синусоиды для 20-ти уровней диммирования

Значения времен удержания синусоиды были внесены в программный код. Таким образом, удалось реализовать двадцать уровней диммирования и покрыть весь диапазон. На рисунках 3.8, 3.9 представлена блок схема и программный код метода.

```
unsigned long Dimmer::_getMicrosByLevel(int level) {
    short convertedLevel = level / 5;
    short micros[] = {
        0,
        1436,
        2048,
        2532,
        2952,
        3333,
        3690,
        4030,
        4359,
        4681,
        5000,
        5319,
        5641,
        5970,
        6310,
        6667,
        7048,
        7468,
        7952,
        8564,
        10000};
    short result = micros[convertedLevel];
    return (unsigned long)result;
}
```

Рисунок 3.8 – Программный код метода _getMicrosByLevel класса Dimmer

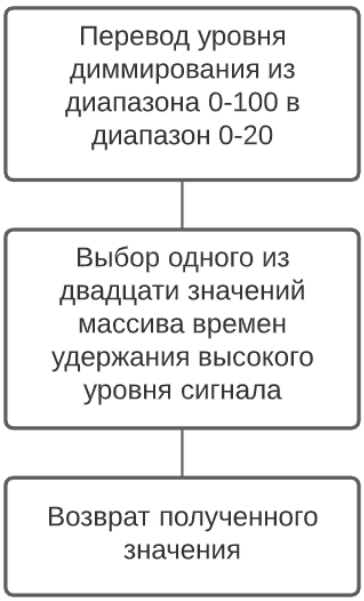

Рисунок 3.9 – Блок-схема метода _getMicrosByLevel класса Dimmer

Измерение относительной освещенности комнаты

Для программной реализации функционала измерения относительной освещенности комнаты был создан класс Photo. Интерфейс этого класса представлен на рисунке 3.10.

```
class Photo {
    public:
        Photo(int inPin, void (*inIncreaseLevel)(), void (*inDecreaseLevel)());
        int pin;                        //пин входа с фотодетектора
        static unsigned long interval;  //интервал считывания
        int requiredValue;              //требуемый уровень освещенности
        void watch();                   //накполение значений и
                                        //корректировка уровня при необходимости
        void setAutoMode();             //включение авто-коррекции
        void resetAutoMode();           //выключение авто-коррекции
        bool isAutoMode();              //включена ли авто-коррекции

    private:
        unsigned long _startMillis;         //время считывания
        bool _autoMode;                     //состояние авто-коррекции
        bool _settingAutoMode;              //флаг процесса включения авто-коррекции
        int _prevOffset;                    //отклонение при предыдущем считывании
        int _read();                        //чтение с фотодетектора
        void _adjustLevel(int readedFromBuffer);//слежение за показаниями для коррекции
        bool _needToAdjust(int offset);     //нужна ли коррекция уровня
        int _values[20];                    //буффер накопленных значений
        int _counter;                       //счетчик для буффера накопленных значений
        void (*_increaseLevel)();           //функция обратного вызова
                                            //при необходимости увеличить освещенность
        void (*_decreaseLevel)();           //функция обратного вызова
                                            //при необходимости уменьшить освещенность
        void _adjustByOffset(int offset);   //коррекция уровня
        void _writeToBuffer(int value);     //запись в буффер данных
        int _readFromBuffer();              //чтение из буфера среднего по всем значениям
        void _setRequiredValue(int readedFromBuffer);//установка требуемого уровня освещенности

};
```

Рисунок 3.10 – Интерфейс класса Photo

Основными методами класса Photo являются watch, _adjustLevel, _needToAdjust. Метод watch вызывается в теле главного цикла микроконтроллера. Блок-схема и код этого метода представлены на рисунках 3.11, 3.12.

```
void Photo::watch() {
    if (!_autoMode) {
        return;
    }

    unsigned long currMillis = millis();

    if (currMillis - _startMillis >= interval) {
        _startMillis = currMillis;

        _writeToBuffer(_read());
        int readedFromBuffer = _readFromBuffer();

        if (_settingAutoMode) {
            _setRequiredValue(readedFromBuffer);
        } else {
            _adjustLevel(readedFromBuffer);
        }

        _counter++;
    }
}
```

Рисунок 3.11 – Программный код метода watch класса Photo

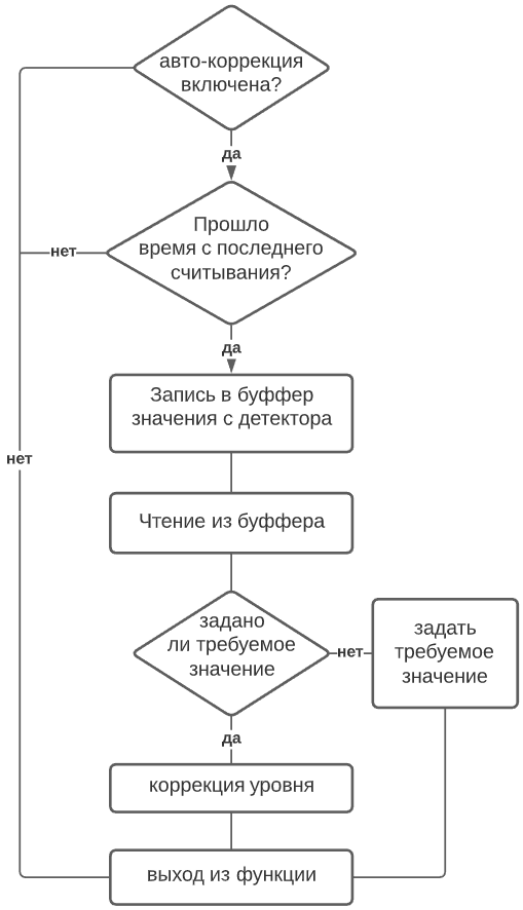

Рисунок 3.12 – Блок-схема метода watch класса Photo

Метод _adjustLevel осуществляет коррекцию уровня и вызывается в методе watch. Он вычисляет текущее смещение среднего измеренного значения от требуемого значения освещенности и если корректировка необходима, выполняет ее на основе текущего смещения. Код метода и блок-схема представлены на рисунках 3.13, 3.14.

```
void Photo::_adjustLevel(int readedFromBuffer) {
    if (readedFromBuffer == -1) {
        return;
    }

    int currOffset = readedFromBuffer - requiredValue;

    if (_needToAdjust(currOffset)) {
        _adjustByOffset(currOffset);
    }

    _prevOffset = currOffset;
}
```

Рисунок 3.13 – Программный код метода _adjustLevel класса Photo

Рисунок 3.14 – Блок-схема метода _adjustLevel класса Photo

Метод _needToAdjust возвращает истину, если требуется коррекция и ложь, если коррекция не требуется. Код и блок схема метода представлены на рисунках 3.15, 3.16.

```
bool Photo::_needToAdjust(int offset) {
    return (_prevOffset * offset > 0) || ((abs(_prevOffset) + 1) < abs(offset));
}
```

Рисунок 3.15 – Программный код метода _needToAdjust класса Photo

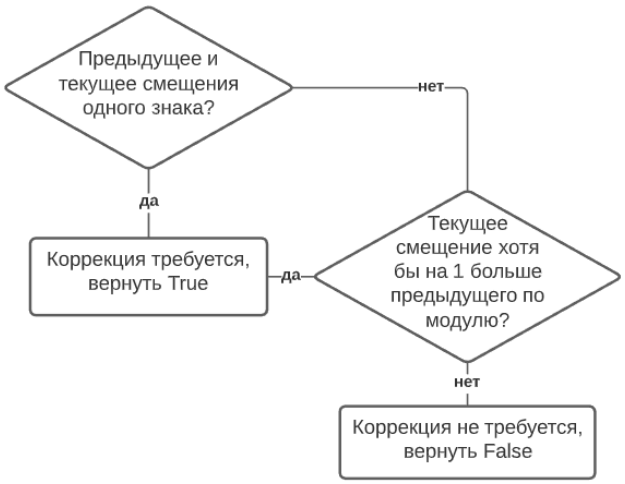

Рисунок 3.16 – Блок-схема метода _needToAdjust
класса Photo

Управление устройством при помощи кнопок на лицевой панели

На лицевой панели расположено три сенсорные кнопки: включение/выключение, кнопка «Плюс», кнопка «Минус». Помимо базовых возможностей по включению/выключению устройства и регулировке уровня диммирования, кнопки должны иметь возможность включать/выключать режим авто-коррекции освещения, а также включать/выключать режим точки доступа для ввода учетных данных. Расширение функциональности кнопок произошло за счет отслеживания долгого нажатия.

Ввод и вывод из режима точки доступа осуществляется посредством долгого нажатия на кнопку «Минус». Включение авто-коррекции уровня освещения выполняется при помощи долгого нажатия на кнопку «Плюс», а выключение при попытке самостоятельно изменить освещенность. Такой функционал реализован программным способом при помощи класса Button. Интерфейс класса представлен на рисунке 3.17.

```
class Button {
    public:
        Button(int inPin, void (*inOnClick)(), void (*inOnLongClick)());
        int pin;                          //пин входа кнопки
        static unsigned long interval;    //интервал времени для
                                          //отслеживания долгого нажатия
        int read();                       //считать уровень сигнала с пина
        void watch();                     //отслеживание прерывания
        void setup(void (*dtct)());       //конфигурирование пина для кнопки
        void interrupt();                 //выставление флага прерывания

    private:
        void _process();                  //обработка нажатий
        bool _detected;                   //флаг прерывания
        unsigned long _startMillis;       //время старта нажатия
        void (*_onClick)();               //функция обратного вызова
                                          //для короткого нажатия
        void (*_onLongClick)();           //функция обратного вызова
                                          //для длительного нажатия
};
```

Рисунок 3.17 – Интерфейс класса Button

Основным методом класса Button является метод _process. Он выполняется по прерыванию изменения уровня входного сигнала, осуществляет разделение долгого и короткого нажатия и вызывает соответствующие функции обратного вызова. Код и блок схема работы метода представлены на рисунках 3.18, 3.19.

```
void Button::_process() {
    const int btnState = read();

    if (btnState == LOW) {
        _startMillis = millis();
        return;
    }

    const unsigned long currMillis = millis();
    const bool longClick = (currMillis - _startMillis >= interval);

    if (longClick) {
        (*_onLongClick)();
    } else {
        (*_onClick)();
    }

    _startMillis = currMillis;
    return;
}
```

Рисунок 3.18 – Программный код метода _process класса Button

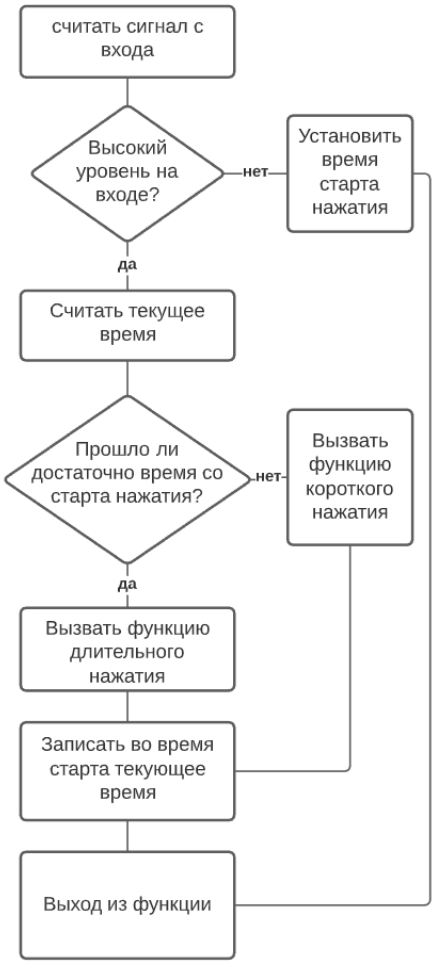

Рисунок 3.19 – Блок-схема метода _process класса Button

Конфигурация в системе «Умный дом»

Для конфигурации в системе «Умный дом» создан класс SoftAp. Основными задачами, которые выполняет эта библиотека, являются:

- переключение микроконтроллера в режим точки доступа
- запуск web-сервера для отправки Web-страниц пользователю
- сбор учетных данных домашней сети Wi-Fi и пользователя системы «Умный дом»

Интерфейс класса SoftAp представлен на рисунке 3.20.

```
class SoftAp {
    public:
        SoftAp(ESP8266WebServer &inServer, Credentials &credentials);
        ESP8266WebServer *server;       //ссылка на экземпляр класса сервера
        Credentials *credentials;       //ссылка на экземпляр класса учетных данных
        String mac;                     //mac-адресс устройства
        void begin();                   //запуск точки доступа
        void end();                     //выключение точки доступа
        bool isSoftAp();                //проверка включена ли точка доступа
        void watch();                   //обработка запросов к запущенному серверу
        void setup(void (*inHandleRoot)(), void (*inHandleCredentials)(), void (*inHandleNotFound)());
                                        //конфигурация точки доступа
        void handleCredentials();       //обработка запроса введенных учетных данных на сервер
        void handleRoot();              //обработка корневого запроса на сервер
        void handleNotFound();          //обработка неннайденной страницы

    private:
        unsigned long _startMillis;     //время последней обработки запросов клиента
        String _makePage(String body);  //создание web-страницы из каркаса
};
```

Рисунок 3.20 – Интерфейс класса SoftAp

Основными методами класса SoftAp являются begin, setup и hadleCredentials. Метод begin переводит микроконтроллер в режим точки доступа, сохраняет mac-адрес и запускает сервер на микроконтроллере.

```
void SoftAp::begin() {
    WiFi.mode(WIFI_AP);

    WiFi.softAP("Dimmer Soft Ap");

    mac = WiFi.softAPmacAddress();

    server->begin();
};
```

Рисунок 3.21 – Программный код метода begin класса SoftAp

Метод setup конфигурирует сервер, указывая обработчики запросов, посылаемых пользователем, когда тот открывает web-страницу диммера для ввода учетных данных.

```
void SoftAp::setup(void (*inHandleRoot)(), void (*inHandleCredentials)(), void (*inHandleNotFound)()) {
    server->on("/", inHandleRoot);
    server->on("/credentials", inHandleCredentials);
    server->onNotFound(inHandleNotFound);
}
```

Рисунок 3.22 – Программный код метода setup класса SoftAp

Метод handleCredentials является обработчиком http POST-запроса клиента, в котором тот отправляет учетные данные. HandleCredentials проверяет корректность данных, сохраняет их в полях экземпляра класса Credentials и отправляет пользователю web-страницу с успешным или неуспешным сохранением данных.

```
void SoftAp::handleCredentials() {
    String ssid = server->arg("ssid");
    String pwd = server->arg("pwd");
    String email = server->arg("email");
    String upwd = server->arg("upwd");

    String body = "<div class=\"bar light\">\
        <span>Данные сохранены! </span>\
        <a href=\"/\"> Назад</a>\
    </div>";

    if (ssid.length() * pwd.length() * email.length() * upwd.length() == 0) {

        body = "<div class=\"bar light\">\
            <span>Данные введены некорректно :( </span>\
            <a href=\"/\"> Назад</a>\
        </div>";

    } else {
        credentials->ssid = ssid;
        credentials->pwd = pwd;
        credentials->email = email;
        credentials->upwd = upwd;
    }

    server->send(200, "text/html", _makePage(body));
}
```

Рисунок 3.23 – Програмный код метода handleCredentials класса SoftAp

Работа в системе «Умный дом».

Для программной реализации работы диммера в рамках системы «Умный дом» был создан класс FbClient, способный выполнять следующие задачи:

- введение микроконтроллера в режим Wi-Fi станции и подключение к домашней сети при помощи учетных данных
- Запуск http-клиента
- Авторизация в сервисах Firebase при помощи учетных данных пользователя

- Получение информации об изменении настроек диммера
- Отправка информации об изменении настроек диммера

Для интеграции сервисов Firebase с http-клиентом умного диммера использовалась библиотека FirebaseESP8266, предоставляющая возможность удобной работы с Realtime Database и Firebase Auth. Интерфейс класса FbClient представлен на рисунке 3.24.

```
class FbClient {
    public:
        FbClient(Credentials &credentials);
        Credentials *credentials;                   //ссылка на экземпляр класса учетных данных
        FirebaseData data;                          //объект полученных данных
        FirebaseData postData;                      //объект отправленных данных
        FirebaseAuth auth;                          //объект авторизации Firebase
        FirebaseConfig config;                      //объект конфигурации Firebase
        String path;                                //путь в бд до диммера
                                                    //кофигурация клиента
        void setup(void (*inValueReceived)(MultiPathStreamData data), void (*inDeviceJson)(FirebaseJson &json), Stri
        void begin();                               //старт клиента
        void watch();                               //отслеживание авторизации и инициализации
        bool isClient();                            //проверка, что включен режим станции (клиента)
        bool reconnect();                           //переподключение к точке доступа Wi-Fi
        void postBool(String childPath, bool value);//записать булевы данные в бд
        void postInt(String childPath, int value);  //записать целочисленные данные в бд

    private:
        unsigned long _startMillis;                 //время с поледней проверки состояния авторизации
        void (*_onValueReceived)(MultiPathStreamData data);//функция обратного вызова, когда данные в бд изменилась
        void (*_populateDeviceJson)(FirebaseJson &json);//функция обратного вызова на заполнение json
        void _errorDataHandler();                   //обработка ошибок при отправке или получении данных
        void _createPath();                         //определение пути до диммера
        void _createDevice();                       //создание объекта диммера в бд, если его нет
        unsigned int _connectTryCounter;            //счетчик попыток переподключения
        bool _initializing;                         //флаг инициализации
        void _beginStream();                        //включение потока передачи данных от бд в диммер
};
```

Рисунок 3.24 – Интерфейс класса FbClient

Основным методом класса FbClient является watch. Метод осуществляет проверку состояния авторизации пользователя, проводит инициализацию устройства в сервисах Firebase и запускает поток получения новых данных Код данного метода и блок-схема работы представлены на рисунках 3.25, 3.26.

```
void FbClient::watch() {
    if (!isClient()) {
        return;
    }

    unsigned long currentMillis = millis();

    if (currentMillis - _startMillis < 3000) {
        return;
    }
    _startMillis = currentMillis;

    if (auth.token.uid.length() == 0) {
        Firebase.getShallowData(data, "/UsersData2/");
        return;
    }

    if (_initializing) {
        _createPath();
        _createDevice();
        _beginStream();
        _initializing = false;
        return;
    }

}
```

Рисунок 3.25 – Программный код метода watch класса FbClient

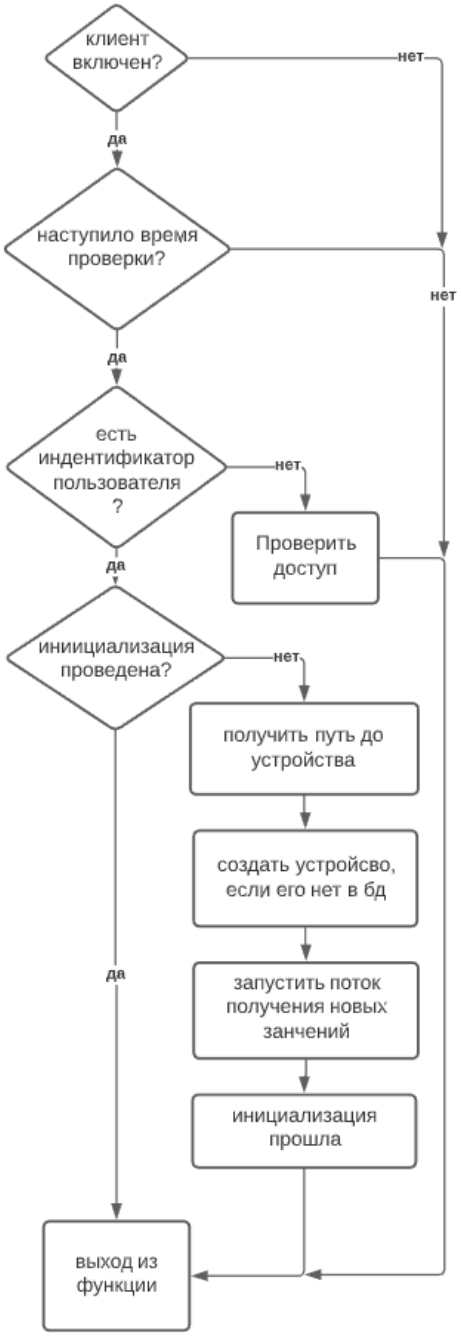

Рисунок 3.26 – Блок-схема метода watch класса FbClient

3.3 Программное обеспечение Web-интерфейса

Web-приложение системы «Умный дом» имеет бессерверную архитектуру и представляет собой совокупность Web-интерфейса, исполняемого в браузере пользователя и облачных технологий Firebase, предоставляющих сервисы для авторизации, хранения и обработки данных. Структура приложения «Умный дом» представлена на рисунке 3.27.

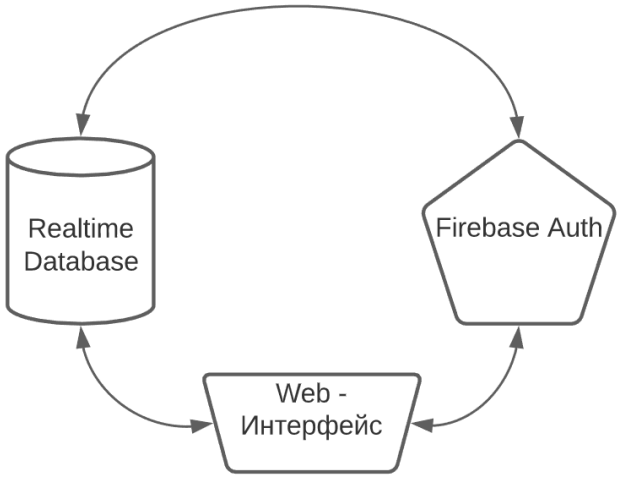

Рисунок 3.27 – Структура Web-приложения «Умный дом»

Для разработки Web-интерфейса использовались традиционные языки клиентской Web-разработки: HTML, CSS, JavaScript. Работа велась в программе Web-Storm IDE. К Web-интерфейсу предъявлялись следующие требования. Web-интерфейс должен:

- быть простым и понятным для пользователя
- проводить авторизацию пользователя
- работать без перезагрузки страницы
- предоставлять возможность управления устройствами
- иллюстрировать расположение устройств по комнатам
- позволять изменять имена комнат, устройств
- создавать, редактировать и удалять сценарии взаимодействия устройств

Для выполнения этих требований при написании программного кода использовался комплексный фреймворк от компании Google под названием

Angular, позволяющий строить модульную структуру. Программный код состоит из трех модулей:

- модуль комнат (операции с комнатами)
- модуль сценариев (операции со сценариями)
- модуль авторизации

Модули комнат и сценариев работают по следующей схеме, представленной на рисунке 3.28.

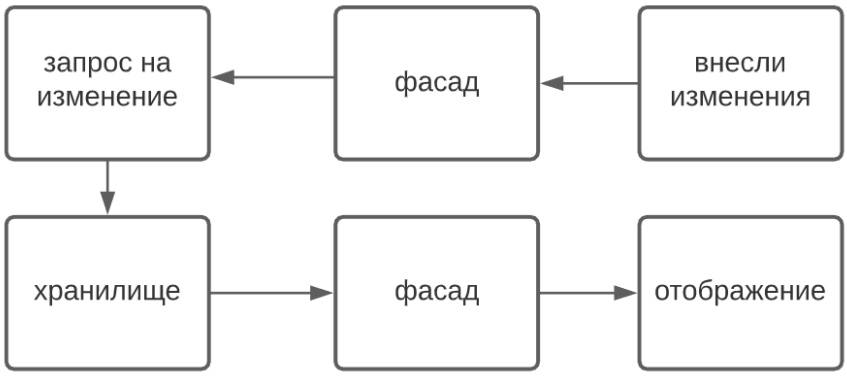

Рисунок 3.28 – Схема работы модулей комнат и сценариев Web-приложения

Основными блоками являются «запрос на изменение», «фасад», «хранилище». Блоки - это классы, экземпляры которых создает Angular в момент исполнения кода браузером. Ниже будут представлены описания классов для модуля комнат.

Класс http-запросов содержит основные методы запросов с предварительным форматированием. Класс http-запросов называется HttpRoomsService и представлен на рисунке 3.29.

```
@Injectable()
export class HttpRoomsService {
    constructor(private readonly http: HttpClient, private authService: AuthService) {}
    //первичная загрузка комнат
    public loadRoomList(): Observable<RoomList> {...}
    //запрос на обновление устройства
    public patchHardware(hardware: Hardware, roomId: Room['id']): Observable<Hardware> {...}
    //запрос на обновление части устройства
    public patchEquipment(
        equipment: Equipment,
        hardwareId: Hardware['id'],
        roomId: Room['id'],
    ): Observable<Equipment> {...}
    //запрос на обновление комнат
    public patchRoom(room: Room): Observable<Room> {...}
    //запрос на создание комнаты
    public postRoom(room: Room): Observable<Room> {...}
    //запрос на удаление комнаты
    public deleteRoom(room: Room): Observable<{ room: Room; response: null }> {...}
    //запрос на обновление списка комнат
    public patchRoomList(roomList: RoomList): Observable<RoomList> {...}
}
```

Рисунок 3.29 – Класс http-запросов HttpRoomService

Класс фасад содержит методы промежуточного слоя для отправки запросов и методы получения данных для отображения. Класс фасад называется RoomListFacade и представлен на рисунке 3.30.

```
@Injectable()
export class RoomListFacade extends LoadableFacade<RoomListState> {
    //комнаты
    public readonly rooms$: Observable<Room[]>;
    //объект для работы над списком комнат
    public readonly roomListEntities$: Observable<Dictionary<Room>>;
    //список комнат
    public readonly roomList$: Observable<RoomList>;
    //комната
    public readonly room$: Observable<Room | undefined>;
    //устройство
    public readonly hardware$: Observable<Hardware | undefined>;

    constructor(store: Store<RoomListState>) {...}
    //получение комнаты по id
    public roomById$(id: Room['id']): Observable<Room> {...}
    //получение устройства по id
    public hardwareById$(roomId: Room['id'], hardwareId: Hardware['id']): Observable<Hardware> {...}
    //получение части устройства по id
    public equipmentById$(
        roomId: Room['id'],
        hardwareId: Hardware['id'],
        equipmentId: Equipment['id'],
    ): Observable<Equipment> {...}
    //обновить комнату
    public updateRoom(room: Room): void {...}
    //добавить комнату
    public addRoom(room: Room): void {...}
    //удалить комнату
    public deleteRoom(room: Room): void {...}
    //загрузить комнаты
    public loadRooms(): void {...}
}
```

Рисунок 3.30 – Класс фасад RoomListFacade

Хранилище представляет собой функцию, которая определяет, как нужно изменить комнаты и/или устройства на основе произошедшего события. Функция хранилища называется roomsReducer и представлена на рисунке 3.31.

```
export function roomsReducer(state :RoomListState = initialState, action: RoomListActions): RoomListState {
    switch (action.type) {
        case RoomListActionsTypes.loadRoomList:
        case RoomListActionsTypes.moveHardware:
        case RoomListActionsTypes.updateOneHardware:
        case RoomListActionsTypes.updateRoom:
        case RoomListActionsTypes.addRoom:
        case RoomListActionsTypes.UpdateOneEquipment:
        case RoomListActionsTypes.DeleteRoom:...
        case RoomListActionsTypes.loadRoomListSuccess:
        case RoomListActionsTypes.moveHardwareSuccess:...

        case RoomListActionsTypes.loadRoomListError:
        case RoomListActionsTypes.moveHardwareError:
        case RoomListActionsTypes.updateRoomFailure:
        case RoomListActionsTypes.addRoomFailure:
        case RoomListActionsTypes.DeleteRoomFailure:...

        case RoomListActionsTypes.updateRoomSuccess:
        case RoomListActionsTypes.upsertRoomWhenLeft:
        case RoomListActionsTypes.UpdateOneEquipmentSuccess:
        case RoomListActionsTypes.updateOneHardwareSuccess:...

        case RoomListActionsTypes.addRoomSuccess:...

        case RoomListActionsTypes.upsertRoomListWhenLeft:...

        case RoomListActionsTypes.DeleteRoomSuccess:...

        default:...
    }
}
```

Рисунок 3.31 – Функция хранилища roomsReducer

Аналогично написан код для модуля сценариев.

Программный код модуля авторизации выглядит иначе и содержит главным образом класс сервиса авторизации. На рисунке 3.32 представлен код сервиса.

```
@Injectable()
export class AuthService {
    constructor(
        private fireAuth: AngularFireAuth,
        private router: Router,
        private readonly snackBar: MatSnackBar,
    ) {...}
    //набор ошибок
    static ErrorMessage = {'auth/invalid-email': 'Не правильный логин и (или) пароль'...};
    private _authState = new BehaviorSubject<AuthState>({redirectUrl: ''...});
    public user$ = this._authState.pipe(map( project: (state :AuthState ) => state.user));
    public isAuthorized$ = this._authState.pipe(
        filter( predicate: (state :AuthState ) => state.callState === LoadingState.LOADED),
        map( project: (state :A ) => !!state.user),
    );
    //состояние авторизации
    get authState(): AuthState {...}
    //произвести вход в аккаунт
    login(email: UserLogIn['email'], password: UserLogIn['password']): void {...}
    //задать состояние
    setState(authState: Partial<AuthState>): void {...}
    //выйти из аккаунта
    logout(): void {...}
    //начать сессию
    startSession(): void {...}
    //запомнить какая была страница перед переходом на страницу входа
    public manageRoute(prevUrl: string, nextUrl: string): Observable<boolean | UrlTree> {...}
    //вернуться на страницу с которой произошел уход после успешной авторизации
    public loginNavigate(prevUrl: string): Observable<boolean | UrlTree> {...}
    //показать сообщение ошибки
    private openSnackBar(message: string, action: string): void {...}
}
```

Рисунок 3.32 – Класс сервис авторзации AuthService

Особенность этого класса заключается в использовании AngularFire, библиотеки сервисов Firebase для Angular приложений. Использование этой библиотеки существенно упрощает интеграцию приложения с сервисами Firebase.

Выводы

Изначально к программному обеспечению узлов системы «Умный дом» предъявлялись серьезные требования по реализации сложного функционала. Однако следуя принципам построения крупных проектов, а также руководствуясь лучшими примерами использования задействованных языков программирования, удалось реализовать заложенный на этапе планирования функционал. Кроме того, написанное программное обеспечение обладает гибкостью и масштабируемостью и пригодно для переиспользования в новых проектах. Следующими закономерными шагами будут являться публикация Web-приложения в сети Internet для доступа из любой точки земного шара, а также отладка программного обеспечения на рабочем прототипе умного диммера и подключение готового устройства к киберфизической системе «Умный дом».

4. Экспериментальная часть

4.1 Внедрение умного диммера

4.1.1 Обзор прототипа устройства

Помимо входа для питания от сети 220В и выхода на нагрузку умный диммер обладает тремя сенсорными кнопками, различающими долгое нажатие, вход для установки программного обеспечения по протоколу UART, вход переключения режимов работы Wi-Fi (MODE) и вход для перезапуска (RST). Также на передней панели расположен датчик освещенности. Размеры прототипа устройства без корпуса 87,5×87,5×27,5 (Ш×В×Г). Изображение прототипа показано на рисунке 4.1.

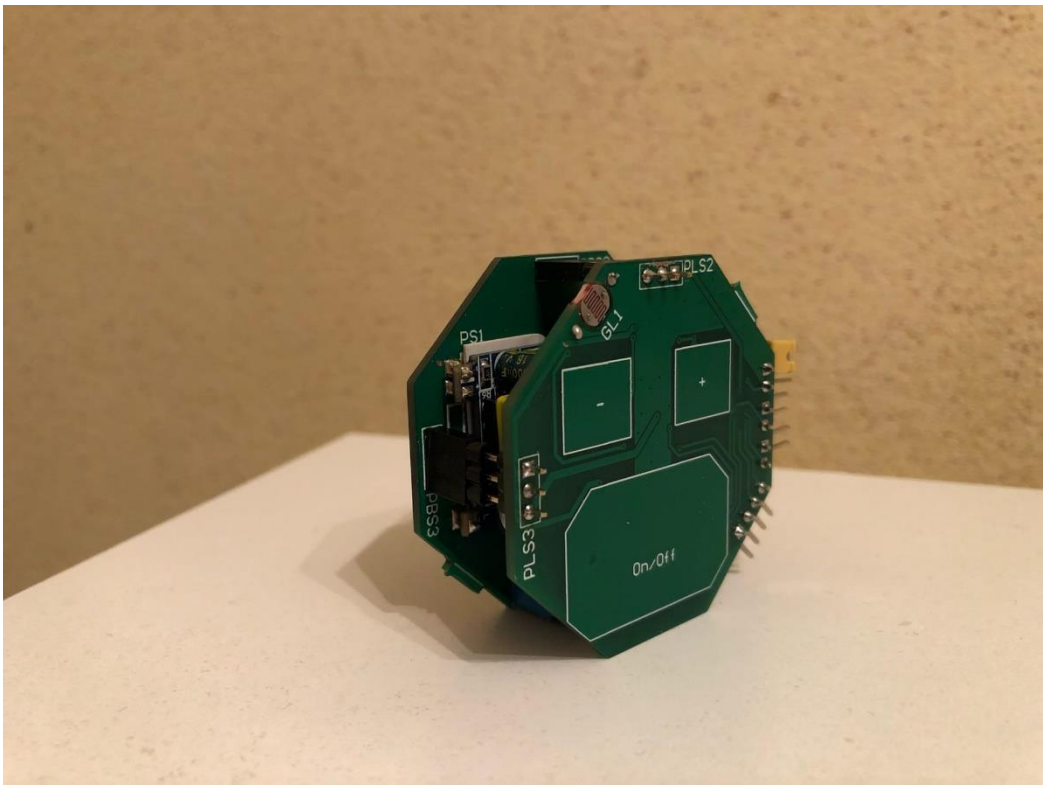

Рисунок 4.1 – Прототип устройства

4.1.2 Включение прототипа устройства

При включении умный диммер пытается подключится к последней сохраненной в flash-памяти Wi-Fi сети. Если после 10 попыток с интервалом в 1 секунду этого сделать не удается, диммер переходит в режим точки доступа для ввода актуальных учетных данных Wi-Fi сети и учетной записи

пользователя системы «Умный дом». Название точки доступа Dimmer Soft Ap. Процесс подключения к точке доступа устройства представлен на рисунке 4.2.

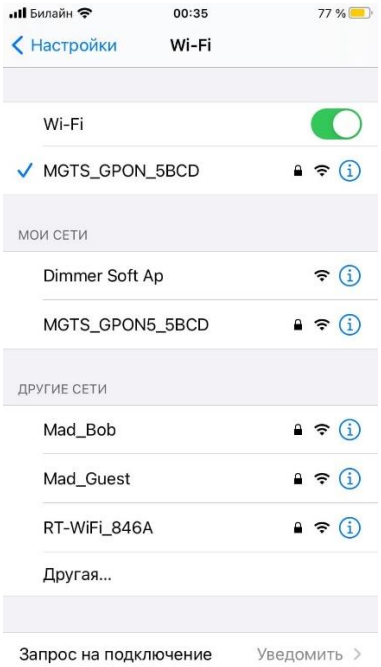

Рисунок 4.2 – Подключение к точке доступа устройства

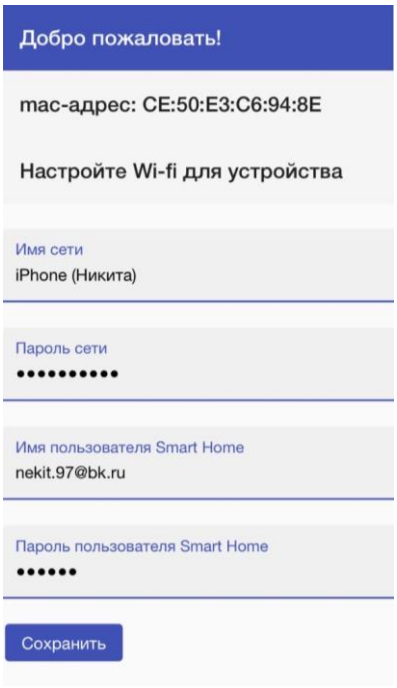

Рисунок 4.3 – Web-страница ввода учетных данных

Для доступа к форме учетных данных необходимо подключится к сети Dimmer Soft Ap. И открыть в браузере web-страницу, которую отдает сервер по ip-адресу 192.168.4.1 . Изображение страницы показано на рисунке 4.3

После получения новых учетных данных диммер сохранит их в Flash-памяти, для того чтобы при перезагрузке вновь подключатся к прошлой сети Wi-Fi и к прошлому личному кабинету системы «Умный дом». Если необходимо снова ввести димммер в режим точки доступа можно использовать длительное нажатие на кнопку «минус» (более 5 секунд).

Назначение сенсорных клавиш очевидно из их названий, однако кнопки обладают дополнительными функциями. Как отмечалось выше, длительное нажатие на клавишу «минус» включает точку доступа и отключается от текущей сети Wi-Fi. Длительное нажатие на клавишу «плюс» включает автоматический режим подстройки яркости. Для выхода из автоматического режима необходимо кратковременно нажать на клавиши «плюс» или «минус».

Для проверки работоспособности устройства использовались таршер переносной, каркас розетки для подключения таршера к диммеру, а также вилка с проводом. Включенное устройство диммер показано на рисунке 4.4.

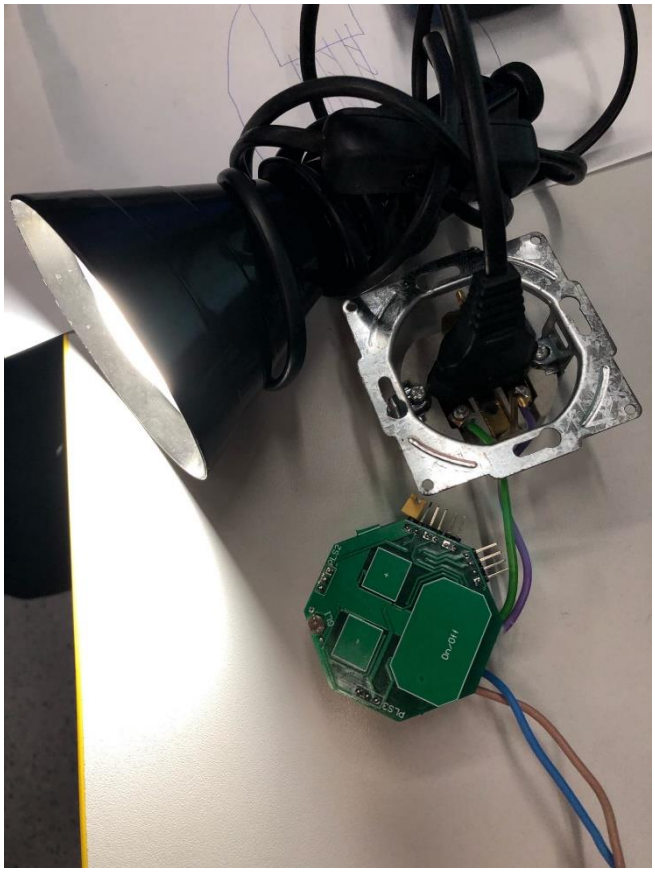

Рисунок 4.4 – Включенное устройство диммер

4.2 Внедрение Web-интерфейса

Для внедрения пользовательского интерфейса использовался еще один облачный сервис платформы Firebase компании Google, называемый Firebase Hosting. Суть сервиса заключается в предоставлении возможности разместить Web-приложение на серверах Firebase и обеспечить доступ к нему по домену 3-уровня в сети Internet. В результате развертывания Web-приложения пользователю стал доступен Web-интерфейс по адресам: https://home-for-u.web.app/ , https://home-for-u.firebaseapp.com/ .

4.2.1 Авторизация

При первом переходе на Web-страницу умного дома пользователю показывается страница входа. На рисунке 4.5 показана страница входа.

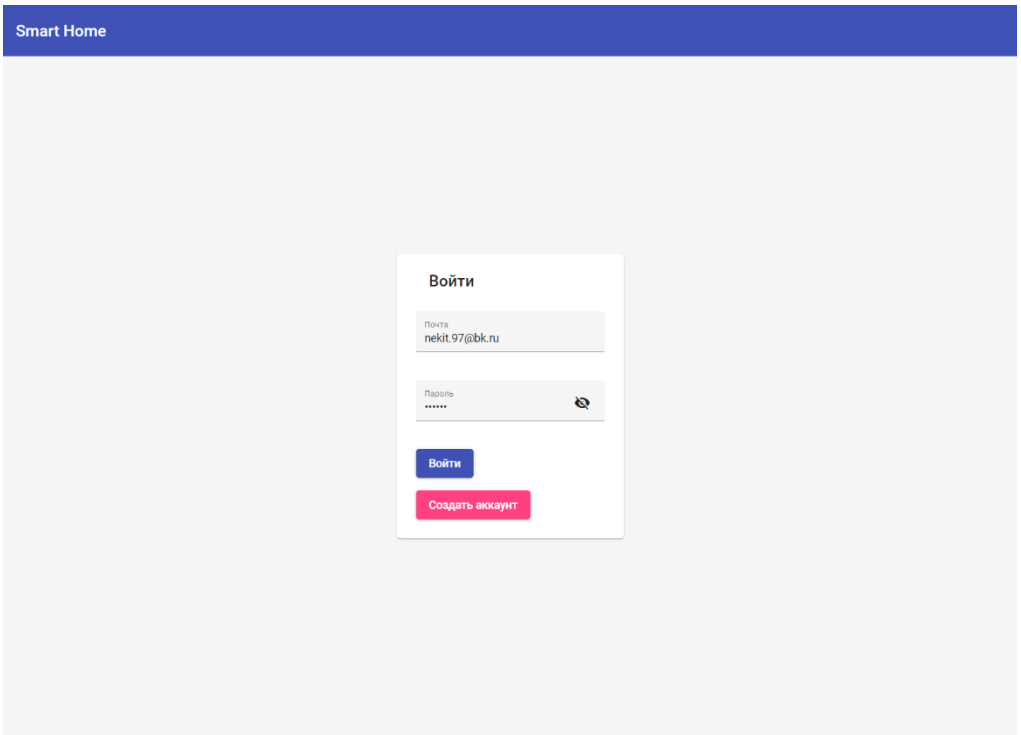

Рисунок 4.5 – Страница входа в личный кабинет

Пользователь может либо войти в личный кабинет, либо зарегистрироваться в системе. Не авторизованному лицу доступ к системе «Умный дом» не предоставляется.

4.2.2 Личный кабинет

При успешном входе в личный кабинет, пользователю показывается страница устройств сгруппированных по комнатам. На рисунке 4.6 показана страница комнат.

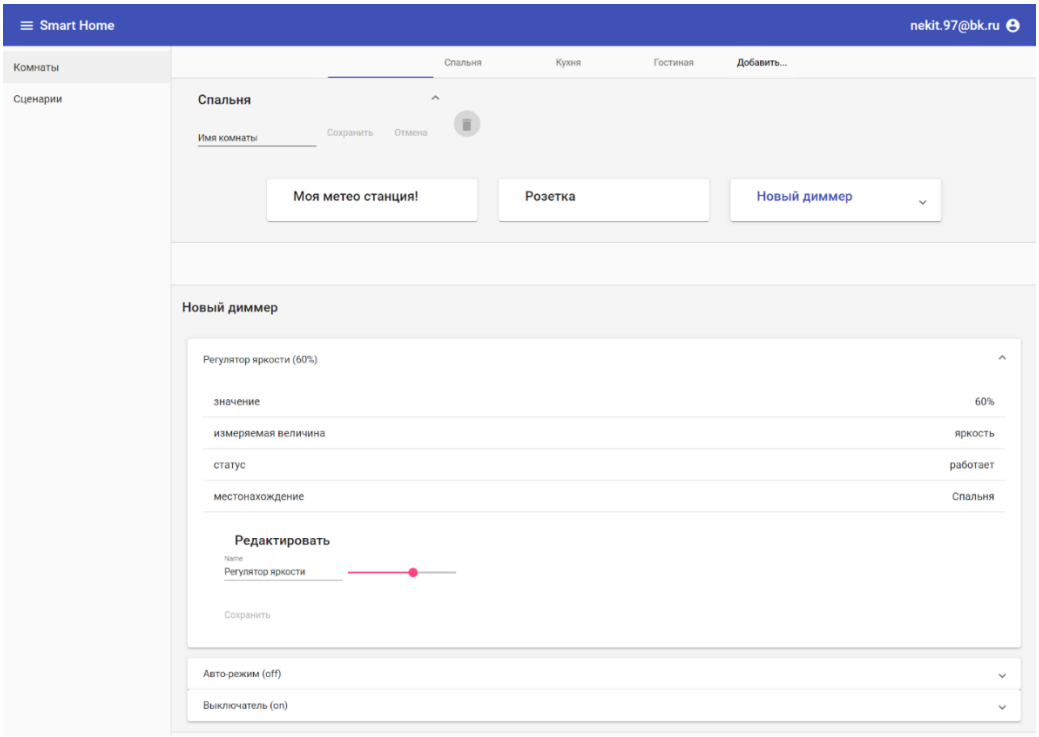

Рисунок 4.6 – Страница комнат с устройствами

Раздел комнат позволяет:

- менять названия комнат и устройств
- переносить устройства из одной комнаты в другую
- создавать новые комнаты
- регулировать доступные параметры приборов.

На рисунке 4.6 показано окно регулировки яркости разработанного диммера. На примере умного диммера интерфейс позволяет регулировать следующие параметры:

- Яркость освещения
- Режим работы (автоматический, ручной)
- Рабочее состояние (включен, выключен)

Для перехода между разделами используется боковое меню. При переходе в раздел сценариев открывается следующая страница.

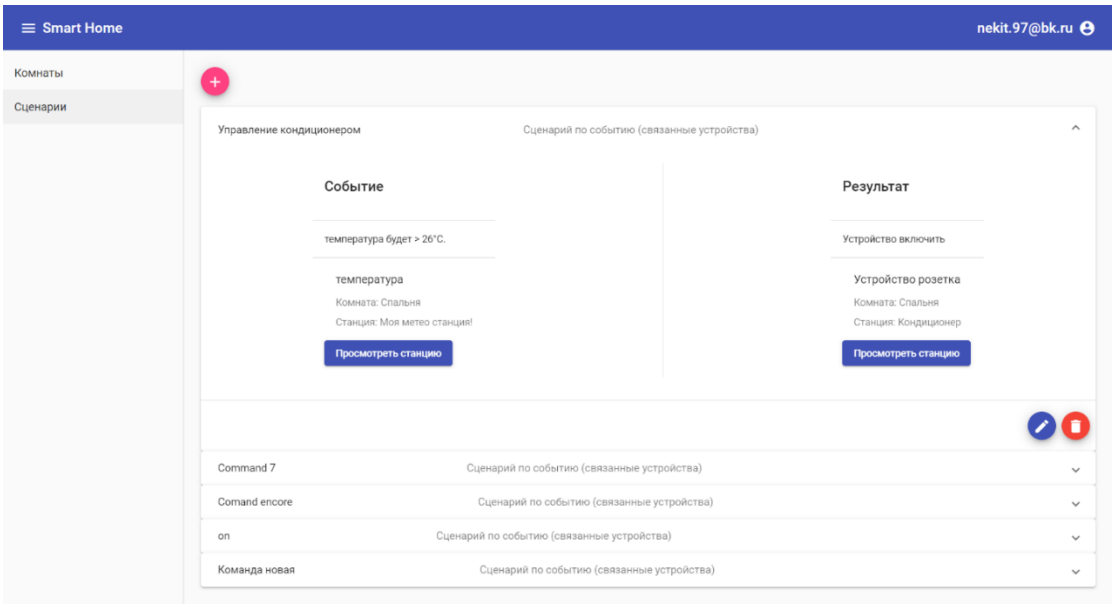

Рисунок 4.7 Страница раздела со сценариями

Раздел сценарии позволяет создавать команды для автоматизированого управления умным домом. Сценарии описывают причино-следственную связь в работе устройств. Для создания команды необходимо указать событие, которое должно произойти и результат, с которым должна отработать система. Ответственность за выполнение команд берет на себя сервер-шлюз. Интерфейс пользователя позволяет создавать, редактировать и удалять команды.

Выводы

В рамках экспериментальной части проведены основные этапы по внедрению прототипа диммера и пользовательского интерфейса в систему «Умный дом». Процесс внедрения прототипа диммера включал:

- сборку прототипа устройства,
- включение прототипа устройства,
- проверку работоспособности усройства в ручном режиме,
- подключение устройства к сети Wi-Fi и к аккаунту пользователя системы «Умный дом»

Процесс внедрения пользовательского интерфейса включал:

- размещение на серверах Firebase Hosting
- вход в личный кабинет
- проверка удаленного управления подключенным умным диммером

Таким образом, удалось воспроизвести полный цикл работы системы «Умный дом, направленной на управление осветительными приборами жилого помещения.

Заключение

Целью дипломного проектирования являлась разработка электронного устройства регулировки мощности осветительных приборов сети 220В, а также пользовательского Web-интерфейса для удаленного управления устройствами системы «Умный дом». Перед началом конструирования электронного устройства был проведен обзор и анализ умных устройств и коммерческих киберфизических систем. Процесс создания свето-регулятора включал стандартный маршрут проектирования электронных устройств: от разработки функциональной схемы до сборки прототипа устройства.

Процесс создания Web-интерфейса состоял из написания программного обеспечения интерфейса с последующим внедрением в киберфизическую систему с помощью облачных услуг.

Таким образом, внедрение разработанных компонентов в развивающуюся систему «Умный дом» позволило запустить полный цикл работы системы для конечного пользователя. Дальнейшие действия будут направлены на улучшение имеющихся устройств системы, например, помещение диммера в корпус, а также на разработку и подключение новых устройств киберфизической системы, например, датчика утечки газа. Работа над устройствами «Умного дома» не ограничивается данной дипломной работой и будет продолжаться в сотрудничестве с преподавателями и студентами кафедры №3 НИЯУ МИФИ.

Выбранный подход к проектированию системы гарантирует её масштабируемость, а открытый исходный код и использование современных электронных и информационных технологий позволяет инженерам – электронщикам и программистам слаженно развивать систему «Умный дом», находясь при этом в совершенно разных уголках земного шара, что выгодно отличает открытую киберфизическую систему «Умный дом» кафедры №3 от коммерческих проектов крупных компаний.

Список источников

1. E. A. Lee, «The Past, Present and Future of Cyber-Physical Systems: A Focus on Models,» Sensors, pp. 1-17, 2015.

2. C. Hong, «Applications of Cyber-Physical System: A,» Word Scientific, pp. 1-8, 2017.

3. K. Matthews, «Five smart factories – and what you can learn from them,» 2020. [В Интернете]. Available: https://internetofbusiness.com/success-stories-five-companies-smart-factories-can-learn/#:~:text=Established%20in%201989%2C%20the%20Siemens,for%20industry%2C%E2%80%9D%20he%20says.. [Дата обращения: 12 09 2020].

4. S. Poslad, Ubiquitous Computing: Smart Devices, Environments and Interactions, Wiley, 2009.

5. С. Бобков, Высокопроизводительные вычислительные системы, Москва: НИИСИ РАН, 2014.

6. А. Фисун и Л. Гращенко, Теоретические и практические основы человеко-компьютерного взаимодействия: базовые понятия человеко-компьютерных систем в информатике и информационной безопасности, Орел: Орловский государственный университет, 2004.

7. А. Сагалович, «Лучшие системы "Умный дом",» 06 04 2020. [В Интернете]. Available: https://simplerule.ru/12-luchshikh-sistem-umnyy-dom/. [Дата обращения: 1 11 2020].

8. «Умный дом Xiaomi,» Xiaomi, 2020. [В Интернете]. Available: https://ru-mi.com/device/umnyiy-dom/. [Дата обращения: 10 11 2020].

9. П. Черемушкин, «Как мы взломали умный дом коллеги,» 2020. [В Интернете]. Available: https://securelist.ru/fibaro-smart-home/94294/. [Дата обращения: 1 11 2020].

10. К. Семенов, «Обзор умного дома REDMOND,» Redmond, 2020. [В Интернете]. Available: https://zoom.cnews.ru/publication/item/63103. [Дата обращения: 10 11 2020].

11. T. Harris, «How Dimmer Switches Work,» HOWSTUFFWORKS, 2015.

12. Institute for Innovation & Technology, «Solving the Phase-Cut Dimming Challenge,» LED Professional, 2015.

13. «Управление мощной нагрузкой переменного тока,» 2008. [В Интернете]. Available: http://easyelectronics.ru/upravlenie-moshhnoj-nagruzkoj-peremennogo-toka.html. [Дата обращения: 10 10 2020].

14. Ю. А. Евсеев и С. С. Крылов, Симисторы и их применение в бытовой электроаппаратуре, Москва: Электроатомиздат, 1990.

15. В. Павлов и В. Ногин, Схемотехника аналоговых электронных устройств, Москва: Горячая линия - Телеком, 2001.

16. Т. Агаханян, Основы транзисторной электроники, Москва: Энергия, 1974.

17. R. Elliot, «3-Wire Trailing Edge Dimmer,» Rod Elliott, 2015. [В Интернете]. Available: https://sound-au.com/project157.htm. [Дата обращения: 31 10 2020].

18. А. Бартош, «Схемотехника блоков питания для светодиодных лент,» 2010. [В Интернете]. Available: http://electrik.info/main/praktika/1337-shemotehnika-blokov-pitaniya-dlya-svetodiodnyh-lent.html. [Дата обращения: 03 11 2020].

19. «Bresenham's line algorithm for power control,» 13 04 2013. [В Интернете]. Available: https://trolsoft.ru/en/articles/bresenham-algo. [Дата обращения: 01 11 2020].

20. ON Semiconductor, «MOC306X 6-Pin DIP Zero-Cross Triac driver Optocoupler,» Literature Distribuition Center for ON Semiconductor, Colorado, 2015.

21. K. H. King, «Phase cut dimming control and protection». Hong Kong Патент PCT/IB2015/055681, 04 02 2016.

22. Ю. Розанов, Основы транзисторной электроники, Москва: Энергоатомиздат, 1992.

23. В. А. Жмудь, И. В. Трубин и М. В. Трубин, «Проектирование сенсорных кнопок на базе микросхемы TTP-224,» Автоматика и программная инженерия , № 1, pp. 70-74, 2015.

24. Espressif, «ESP8266 Technical Reference,» Shanghai, 2020.