МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки)

Электроника и робототехника
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка встроенного ПО шлюза канала передачи данных

Обучающийся К. А. Рахманов _____
(Инициалы Фамилия) (личная подпись)

Руководитель к. т. н., доцент, А. А. Шевцов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к. п. н., доцент, О. В. Лебединская
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Выпускная квалификационная работа посвящена разработке шлюза канала передачи данных, а также его встроенного и служебного ПО. Пояснительная записка разделена на 8 параграфов, в которых описывается процесс разработки структуры устройства, алгоритма его работы, схемотехнического решения с выбором подходящих модулей, проектирование печатной платы и расчет стоимости ее производства, а также приводится процесс разработки ПО.

К пояснительной записке прилагаются следующие графические материалы:

1. обзор аналогов;
2. структурная схема устройства;
3. структура информационной системы.
4. блок-схема алгоритма работы ПО;
5. схема электрическая принципиальная;
6. чертеж печатной платы;
7. сборочный чертеж;

Также с пояснительной запиской в качестве приложений идут полный листинг встроенного ПО и листинг страницы HTML.

Abstract

The title of the graduation project is «Software development for data transferring gate».

Explanatory note consists of an introduction, confirming the relevance of the chosen topic, the main part, consisting of eight chapters, which detail the development process, the conclusion, reflecting the results of the work, list of references including foreign sources and the graphic part on 7 A1 sheets.

The aim of the work is to give some information about the process of the development of the device, data structure design and writing embedded and auxiliary software.

The graduation work may be divided into several logically connected parts which are: analysis of existing analogs; structure development; development of the device functioning algorithm; making the component list and creating the electronics scheme; writing embedded software for Wi-Fi module and auxiliary software; PCB layout; economic calculation.

We first discuss what analogs already exist and what parameters they have. Then we analyze what modules we need for device functionality and choose the components to make an electronics scheme. We also prepare the device functioning algorithm and develop the software which supports it. We outline the process of the PCB layout and perform economic calculations.

Finally, we present PCB drawing, construction drawing and also complete code.

In conclusion we'd like to stress this work is relevant in solving the problem of the management of the smart home devices as well as connection of the different wireless technologies.

Содержание

Введение 5
1 Обзор состояния вопроса 7
2 Разработка структурной схемы устройства 16
3 Разработка алгоритма функционирования устройства 20
4 Разработка схемы электрической принципиальной 25
5 Разработка ПО для модуля Wi-Fi 43
6 Разработка служебного ПО 64
7 Разработка конструкции устройства 74
8 Экономическая часть 79
Заключение 82
Список используемых источников 83
Приложение А Встроенное ПО 85
Приложение Б Главная страница HTML 90

Введение

Разработанное устройство должно выполнять связь между сетями Wi-Fi и Zigbee. В качестве модуля Zigbee используется модуль E18-MS1-PCB. Шлюз выступает в роли клиента в сети устройств Wi-Fi. В качестве сервера выступает компьютер администратора.

Шлюз выполняет подключение к Wi-Fi сети с помощью мобильного приложения или предварительно указанных в программе SSID и пароля. Шлюз имеет индивидуальный идентификатор, который задается на уровне программы.

Шлюз позволяет проводить подключение с помощью мобильного приложения по протоколу Bluetooth для проведения настройки. Через приложение шлюзу передается информация о сети Wi-Fi, к которому он должен подключаться. Эти параметры сохраняются в память SPIFFS.

В основном цикле программы шлюз работает в режиме конечного устройства в сети Wi-Fi, созданной в помещении. При этом используются SSID и пароль, заданные на этапе настройки.

Шлюз обменивается информацией с сервером, отправляет сообщение серверу по запросу, который содержит перечень устройств в сети, включая идентификатор шлюза и головного модуля Zigbee. Шлюз получает сообщение в ответ от сервера, которое содержит продолжительность времени сна и целевое значение освещенности. Так же шлюз связывается с сервером точного времени. Функционирование устройства осуществляется в пределах описанных функций по временному интервалу – момент опроса статуса сетей помещения с диагностическими целями. Данный временной интервал шлюз получает из ответа от сервера.

Шлюз формирует таблицу для сети Zigbee на основании принятых параметров и передает их по UART в модуль Zigbee.

Каждое отключение питания вызывает сброс и перезапуск модулей.

Сервер занимается формированием сообщения целевому шлюзу и таблицы требуемого состояния устройств. Также на сервера располагается база данных с таблицей, которая содержит данные идентификаторов устройств, время сна, целевые параметры. В таблице предусмотрены резервные поля.

1 Обзор состояния вопроса

Разнородная сеть – это беспроводная сеть, состоящая из различных типов технологий беспроводного доступа, таких как сотовые сети, Wi-Fi и Bluetooth, которые объединены вместе для обеспечения бесперебойного подключения пользователей.

Одним из ключевых преимуществ разнородных сетей является их способность поддерживать широкий спектр устройств с различными требованиями. Например, сотовые сети предназначены для поддержки устройств, которым требуется высокоскоростная передача данных и покрытие на большом расстоянии, в то время как Wi-Fi предназначен для локальных сетей и покрытия на меньшем расстоянии. Благодаря интеграции этих технологий такие сети могут обеспечить бесперебойную связь с широким спектром устройств, включая смартфоны, планшеты, ноутбуки и устройства IoT [4].

Еще одним преимуществом разнородных сетей является их способность обеспечивать балансировку нагрузки и разгрузку трафика. В таких сетях устройства могут быть интеллектуально направлены в наиболее подходящую сеть на основе их местоположения, типа и текущего использования. Это может помочь уменьшить перегрузку сотовых сетей и улучшить общую производительность сети [17].

Разнородные сети могут играть важную роль в обеспечении связи устройств умного дома друг с другом и с облаком. Интегрируя различные беспроводные технологии, такие как Wi-Fi, Zigbee и Bluetooth, такие сети могут обеспечить бесшовное соединение между устройствами и позволить им работать вместе скоординированным образом [5].

Например, умный дом может использовать Wi-Fi для высокоскоростной передачи данных между устройствами, Zigbee для связи между датчиками и исполнительными механизмами с низким энергопотреблением и низкой задержкой, а Bluetooth для локальной связи между устройствами. Используя

такой подход, умный дом может воспользоваться преимуществами каждой технологии и обеспечить прочную и надежную инфраструктуру связи [1].

Разнородные сети также могут обеспечить новые варианты использования в умных домах, такие как интеллектуальное управление энергопотреблением и домашняя автоматизация. Благодаря интеграции интеллектуальных датчиков и исполнительных устройств в сеть, разнородные сети могут обеспечить мониторинг и управление энергопотреблением, системами отопления и охлаждения и другими бытовыми приборами в режиме реального времени. Это может помочь снизить потребление энергии и сэкономить расходы домовладельцев [17].

В целом, разнородные сети являются важным достижением в области беспроводных сетей, поскольку они представляют собой гибкое и масштабируемое решение для удовлетворения растущего спроса на услуги беспроводной передачи данных [17].

Разнородные сети могут поддерживать различные протоколы беспроводной связи, в зависимости от конкретного случая использования и требований. Некоторые распространенные протоколы, используемые в таких сетях, включают:

– Wi-Fi – это популярный протокол беспроводной связи, который поддерживает высокоскоростную передачу данных на короткие и средние расстояния;

– сотовые сети, такие как 4G и 5G, обеспечивают широкое покрытие территории и поддерживают высокоскоростную передачу данных для мобильных устройств;

– Zigbee – это протокол беспроводной связи с низким энергопотреблением, который широко используется в «умных домах» для связи между устройствами;

– Bluetooth – это протокол беспроводной связи малого радиуса действия, который обычно используется для связи между устройствами и передачи потокового аудио [9];

– NFC (Near Field Communication) – это протокол беспроводной связи малого радиуса действия, который обычно используется для бесконтактных платежей и передачи данных;

– RFID (радиочастотная идентификация) – это протокол беспроводной связи, который обычно используется для отслеживания и идентификации объектов [5].

Помимо беспроводных технологий, в разнородных сетях могут использоваться проводные средства связи, такие как Ethernet и оптоволоконные кабели, для обеспечения высокоскоростной и надежной магистрали сети. Это позволяет устройствам общаться друг с другом и иметь доступ к Интернету с помощью проводных или беспроводных соединений, в зависимости от доступности и пригодности среды [5].

Интеграция различных беспроводных технологий в единую сеть позволяет создать более эффективную и действенную сеть, способную удовлетворить разнообразные требования различных приложений и устройств. Это может привести к улучшению производительности, повышению надежности и снижению затрат по сравнению с сетью с одной беспроводной технологией.

Для передачи информации используют шлюзы. Шлюз – это сетевое устройство или программная система, которая выступает в качестве посредника между двумя или более сетями, позволяя им взаимодействовать друг с другом, даже если они используют различные протоколы связи или сетевые архитектуры. В контексте разнородных сетей шлюзы играют важную роль в обеспечении связи между различными беспроводными технологиями [2].

Шлюзы обычно обеспечивают преобразование и перевод протоколов, позволяя передавать данные между различными сетями, использующими разные протоколы связи. Они также могут выполнять другие функции, такие как безопасность, маршрутизация и управление сетью [2].

В сетях шлюзы могут использоваться для соединения различных беспроводных технологий, таких как Wi-Fi, сотовая связь и сети Zigbee. Например, шлюз Zigbee-to-Wi-Fi может использоваться для обеспечения связи датчиков и устройств, использующих протокол Zigbee, с сетью Wi-Fi, что позволяет передавать данные на облачный сервер или в мобильное приложение.

Шлюзы также могут использоваться для обеспечения связи между беспроводными технологиями, которые работают в разных частотных диапазонах или используют разные схемы модуляции. Например, шлюз может использоваться для соединения сети LoRaWAN, работающей в диапазоне частот до 1 ГГц, с сотовой сетью, работающей в гигагерцовом диапазоне частот [10].

Некоторые примеры шлюзов включают:

– протокольные шлюзы: Эти шлюзы осуществляют перевод между различными протоколами связи, такими как TCP/IP, HTTP и FTP, позволяя устройствам и приложениям, использующим различные протоколы, взаимодействовать друг с другом;

– сетевые шлюзы: Эти шлюзы соединяют различные типы сетей, такие как локальные сети, глобальные сети и Интернет, позволяя устройствам в одной сети взаимодействовать с устройствами в другой сети;

– шлюзы приложений: Эти шлюзы обеспечивают доступ к определенным приложениям или услугам, размещенным в различных сетях или платформах, например, веб-приложениям или облачным сервисам;

– шлюзы безопасности: Эти шлюзы обеспечивают функции безопасности, такие как брандмауэры, обнаружение и предотвращение вторжений и фильтрация содержимого, для защиты сетей и устройств от вредоносной активности [4].

В целом, шлюзы играют важную роль в обеспечении связи между различными беспроводными технологиями в разнородных сетях, позволяя

обеспечить более эффективную и действенную связь между устройствами и сетями.

Один из самых популярный шлюзов, который используется в сетях «умного дома» – Xiaomi Gateway 4. Он позволяет централизованно контролировать и управлять различными устройствами, производимыми компанией Xiaomi.

Xiaomi Gateway 4 – это мощное и универсальное устройство, позволяющее централизованно контролировать и управлять различными устройствами «умного дома» производства Xiaomi. Он поддерживает протоколы связи Zigbee 3.0 и Wi-Fi, который работает на частотах 2,4 ГГц и 5 ГГц, и оснащен быстрым процессором Cortex-A7 1,2 ГГц, 256 Мб флэш-памяти и 512 Мб оперативной памяти. Устройство поддерживает управление через мобильное приложение. Шлюз питается от источника постоянного тока 5 В/1 А, для этого используется порт USB Type-C [20].

Xiaomi Gateway 4 работает с протоколом Bluetooth версии 5.0, что дает возможность подключиться к устройству с помощью телефона и выполнить первоначальную настройку, которая включает задание данных о Wi-Fi. Помимо этого, шлюз имеет порт Ethernet, через который его можно подключить напрямую к маршрутизатору в домашней сети [20].

Дальность действия шлюза Xiaomi Gateway 4 зависит от используемого протокола связи [20].

При использовании протокола Zigbee радиус действия может варьироваться в зависимости от таких факторов, как количество стен и препятствий между шлюзом и управляемым устройством, а также сила сигнала Zigbee. Как правило, радиус действия Zigbee составляет до 10-100 метров на открытом пространстве [20].

При использовании Wi-Fi радиус действия зависит от силы сигнала и возможностей используемого маршрутизатора. Как правило, радиус действия Wi-Fi составляет до 30-50 метров в помещении, но этот показатель может

варьироваться в зависимости от конкретного маршрутизатора и условий эксплуатации [20].

Радиус действия Xiaomi Gateway 4 может варьироваться в зависимости от ряда факторов, но он предназначен для работы в типичных домашних или офисных условиях [20].

Другое аналогичное устройство – Samsung SmartThings V3 Hub. Это комплексная платформа автоматизации умного дома, предназначенная для инженеров, которые хотят создавать и настраивать свои решения. Платформа предоставляет шлюзовое устройство, которое выступает в качестве центральной точки связи для всех совместимых устройств умного дома, используя беспроводные протоколы, такие как Zigbee, Z-Wave и Wi-Fi. SmartThings поддерживает широкий спектр устройств, включая термостаты, замки, светильники, камеры безопасности и многое другое [18].

Для инженеров доступно рабочее пространство разработчика SmartThings для создания собственных обработчиков устройств и пользовательских приложений, что позволяет интегрировать в платформу новые устройства и функциональные возможности. Это возможно, благодаря открытому API, который позволяет программно получать доступ к данным и управлять устройствами. API может быть использован для создания пользовательских интеграций, автоматизации задач и создания новых пользовательских интерфейсов для собственных решений «умного дома» [18].

Шлюз от Samsung обеспечивает совместимость с популярными голосовыми помощниками, включая Amazon Alexa и Google Assistant [18].

Шлюз SmartThings работает на базе процессора ARM Cortex-A53 с частотой 1,7 ГГц, имеет 512 МБ оперативной памяти и 4 ГБ памяти для хранения данных. Он подключается к домашней сети через Ethernet или Wi-Fi, который поддерживает частоту 2,4 ГГц. Также шлюз поддерживает различные беспроводные протоколы, такие как Zigbee, Z-Wave. Устройство не поддерживает протокол Bluetooth, однако мобильное приложение позволяет

управлять устройствами, использующими такой протокол. В таком случае само приложение выступает в роли шлюза [18].

Радиус действия шлюза Samsung зависит от используемого беспроводного протокола. Производитель заявляет, что устройство работает на расстоянии 15-40 метров [18].

Для первоначальной настройки шлюза его необходимо подключить к домашней сети с помощью кабеля Ethernet к маршрутизатору в домашней сети. После этого в мобильном приложении появится возможность добавить устройство в учетную запись [18].

SmartThings Hub также питается от источника постоянного тока 5 В/1 А, для этого предусмотрен порт. Помимо этого, шлюз оснащен резервной батареей, которая позволяет ему работать в течение 10 часов в случае отключения электроэнергии. Также устройство имеет 1 интерфейс USB 2.0 .

Помимо готовых решений от больших компаний, существую также проекты от энтузиастов. Одним их таки проектов является Home Assistant.

Home Assistant – это платформа домашней автоматизации с открытым исходным кодом, написанная на Python 3. Она разработана таким образом, чтобы быть легкой, расширяемой и простой в использовании. Home Assistant позволяет пользователям управлять и автоматизировать устройства различных производителей, такие как умные светильники, выключатели и термостаты, с помощью единого интерфейса. Он также поддерживает широкий спектр интеграций, позволяя пользователям подключать свои устройства и службы к платформе.

Home Assistant имеет гибкую архитектуру и может работать на различных платформах, включая Linux, macOS и Windows. Его можно установить на Raspberry Pi, виртуальную машину или выделенный сервер. Платформа использует RESTful API и поддерживает MQTT, что позволяет легко обмениваться данными между устройствами.

Одной из функций Home Assistant является его механизм автоматизации. Он позволяет пользователям создавать сложные правила и сценарии

автоматизации с помощью простого интерфейса. Пользователи могут активировать эти средства автоматизации на основе событий, времени или состояний устройств, а также могут использовать широкий спектр условий и действий для создания настраиваемых режимов работы своих устройств.

Еще одна ключевая особенность Home Assistant – поддержка широкого спектра интеграций и устройств. Платформа поддерживает более 1500 интеграций, включая популярные устройства для умного дома, такие как Philips Hue, Nest и Sonos. Он также поддерживает ряд протоколов связи, включая MQTT, Zigbee, Z-Wave, Bluetooth, Wi-Fi, Infrared, RESTful и многие другие, что позволяет пользователям подключать к платформе широкий спектр устройств.

Home Assistant – это мощная и гибкая платформа «умного дома», которая предоставляет пользователям единый интерфейс для управления и автоматизации своих умных домашних устройств. Его открытый исходный код, широкий спектр интеграций и мощный механизм автоматизации делают его популярным выбором среди энтузиастов и инженеров.

Home Assistant – программное решение, однако у компании есть и схемотехническое решение шлюза, которое называется Home Assistant Yellow.

Home Assistant Yellow основан на популярной платформе Raspberry Pi, но со специально разработанной печатной платой для Home Assistant. Он оснащен четырехъядерным процессором ARM Cortex-A53 с 4 ГБ оперативной памяти, что обеспечивает достаточную вычислительную мощность для задач автоматизации умного дома. Шлюз имеет только подключение по Ethernet, а также беспроводное соединение по протоколу Zigbee. Bluetooth и WiFi не поддерживаются. Заявленная дальность действия шлюза – 10-30 метров [14].

Однако Home Assistant Yellow имеет модульную структуру, что позволяет улучшать его со временем. Так, замена Raspberry Pi, на другу версию добавит поддержку WiFi и Bluetooth. На плате предусмотрено место для SSD, что позволит увеличить хранилище, так как стандартная версия имеет всего лишь 16 ГБ [14].

Шлюз питается от источника постоянного тока 12 В/2 А. Home Assistant Yellow имеет 2 порта USB 2.0 и порт USB Type-C. Также он имеет разъем 3,5 мм jack для подключения аудио устройств [14].

Перед использованием шлюза его необходимо подключить к домашней сети с помощью кабеля Ethernet. После этого появится возможность подключиться к устройству через мобильное приложение. Оно автоматически определит шлюз в локальной сети [14].

Таким образом, все обозреваемы устройства имеют сходства: поддержка нескольких типов беспроводной связи.

2 Разработка структурной схемы устройства

Разработка шлюза обычно начинается с определения требований и спецификаций шлюза. Это включает в себя понимание сетевых протоколов и стандартов связи, которые должен поддерживать шлюз, а также аппаратных и программных компонентов, которые потребуются для реализации этих функций.

Для связи с устройствами «умного дома» в шлюзе будет использоваться модуль, который поддерживает протокол Zigbee.

Zigbee – это протокол беспроводной связи с низким энергопотреблением, позволяющий устройствам обмениваться данными друг с другом на коротких. Он основан на стандарте IEEE 802.15.4 и работает в диапазоне частот 2,4 ГГц.

Zigbee использует топологию ячеистой сети, которая позволяет устройствам взаимодействовать друг с другом, даже если они не находятся в пределах досягаемости друг друга. Это возможно за счет того, что устройства могут действовать как маршрутизаторы, передавая сообщения другим устройствам в сети. Это позволяет сети покрывать большую площадь и повышает ее надежность.

Zigbee обычно используется в устройствах умного дома, таких как лампочки, термостаты и дверные замки, что позволяет им взаимодействовать друг с другом и управляться через центральный концентратор. Он также используется в промышленных и коммерческих приложениях для таких задач, как автоматизация зданий, управление освещением и отслеживание активов.

В целом, Zigbee – это мощный и гибкий протокол беспроводной связи, который хорошо подходит для широкого спектра приложений интернета вещей. Он предлагает низкое энергопотребление, надежные функции безопасности и возможность формирования самовосстанавливающихся ячеистых сетей, что делает его идеальным выбором для автоматизации умного дома и промышленной автоматизации.

Для связи с сервером в шлюзе должен присутствовать модуль Wi-Fi. Wi-Fi – это технология беспроводной сети, которая позволяет устройствам взаимодействовать друг с другом через локальную сеть (LAN) без необходимости использования физических кабелей. Wi-Fi использует радиоволны для передачи данных между устройствами и полагается на стандарт IEEE 802.11, определяющий, как должна работать беспроводная связь. Этот стандарт устанавливает набор правил и протоколов, которые определяют, как данные передаются, принимаются и интерпретируются устройствами с поддержкой Wi-Fi.

По своей сути Wi-Fi работает путем передачи данных между маршрутизатором или точкой доступа и одним или несколькими устройствами. Маршрутизатор служит центральным узлом сети, и устройства могут подключаться к нему по беспроводной сети для доступа в Интернет или общаться друг с другом. Wi-Fi использует метод скачкообразной перестройки частоты с расширенным спектром, чтобы избежать помех другим беспроводным устройствам, и работает на нескольких различных частотах, включая 2,4 ГГц и 5 ГГц.

Что касается аппаратных компонентов, система Wi-Fi обычно включает в себя беспроводной маршрутизатор или точку доступа, которая отвечает за передачу и получение данных, а также беспроводной адаптер в каждом устройстве, которому необходимо подключиться к сети. Беспроводной адаптер преобразует данные в радиосигнал и отправляет их на маршрутизатор, который затем пересылает их по назначению.

Для взаимодействия с мобильным приложением необходим модуль Bluetooth. Bluetooth – это технология беспроводной связи, которая использует радиоволны для передачи данных между устройствами на короткие расстояния. Он работает в диапазоне частот 2,4 ГГц и использует расширенный спектр со скачкообразной перестройкой частоты (FHSS) для уменьшения помех от других устройств. Bluetooth использует архитектуру «ведущий-ведомый», в которой одно устройство выступает в качестве

ведущего и управляет связью с другими устройствами, выступающими в качестве ведомых [9].

Bluetooth использует маломощный, недорогой и простой стек протоколов, что делает его идеальным для небольших портативных устройств, таких как смартфоны, наушники и устройства интернета вещей. Он поддерживает несколько профилей для различных типов приложений, включая потоковое аудио, передачу данных и услуги на основе местоположения. Bluetooth также разработан с функциями безопасности для защиты от прослушивания и несанкционированного доступа [9].

Помимо модулей связи в устройстве должны быть модули, отвечающие за питание. Шлюз будет иметь возможность питаться, как от сети 220 В, так и от блоков питания мобильных телефонов.

Питание от сети будет осуществляться через разъем питания. Все модули в шлюзе работают на напряжении ниже 220 В, поэтому между разъемом и модулями должен присутствовать понижающий преобразователь напряжения. При работе от блоков питания мобильных телефонов питание будет осуществляться через разъем USB Type-C, то есть напряжение питания будет равно 5 В. Его тоже необходимо преобразовывать. Для это нужен отдельный модуль.

Для индикации наличия питания шлюза на плате будет предусмотрен 1 светодиод.

Для настройки модуля Wi-Fi на плате будут предусмотрены две кнопки, для настройки модуля Zigbee будет предусмотрена одна кнопка. Они будут скрываться под корпусом устройства и не будут доступны пользователю. На корпусе также будет присутствовать одна пользовательская кнопка для сброса настроек устройства и перехода в режим ожидания.

Структурная схема устройства показана на рисунке 1.

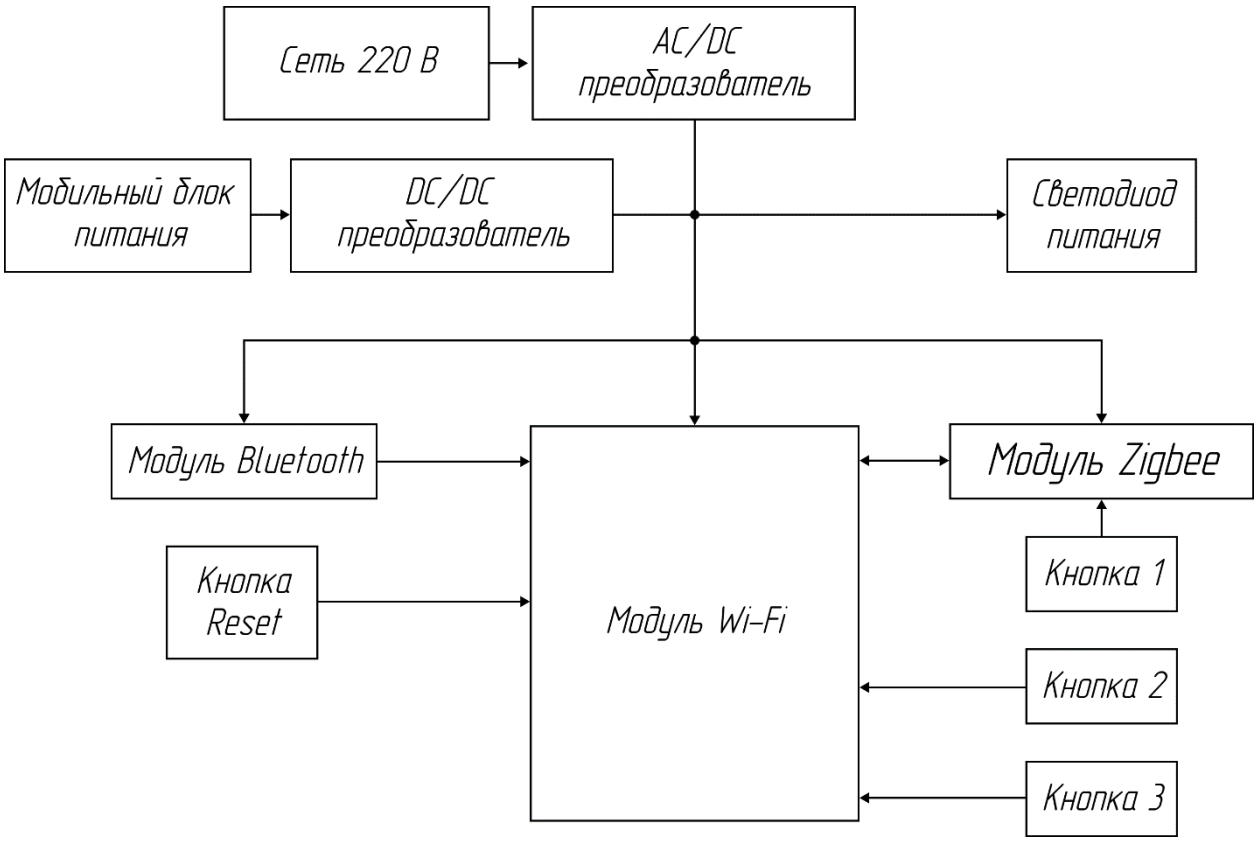

Рисунок 1 – Структурная схема устройства

Таким образом, шлюз будет поддерживать 3 типа беспроводных сетей, каждый из которых будет выполнять свои функции. Wi-Fi для выхода в интернет и связи с сервером. Zigbee для связи с устройствами умного дома. Bluetooth для связи с мобильным приложением и выполнения первичной настройки устройства.

3 Разработка алгоритма функционирования устройства

Алгоритм работы устройства – это набор инструкций или программа, управляющая поведением устройства. Он определяет правила и процедуры, которым должно следовать устройство, чтобы выполнять свои функции. Алгоритм обычно принимает входные данные от пользователя или других внешних источников, обрабатывает информацию и генерирует соответствующие выходные данные.

Устройство должно выполнять несколько функций, а именно подключение по Bluetooth, подключение к Wi-Fi сети, отправка запросов на сервер и получение ответа, обработку данных и отправку их в модуль Zigbee. На рисунке 2 показана блок-схема основного алгоритма работы шлюза.

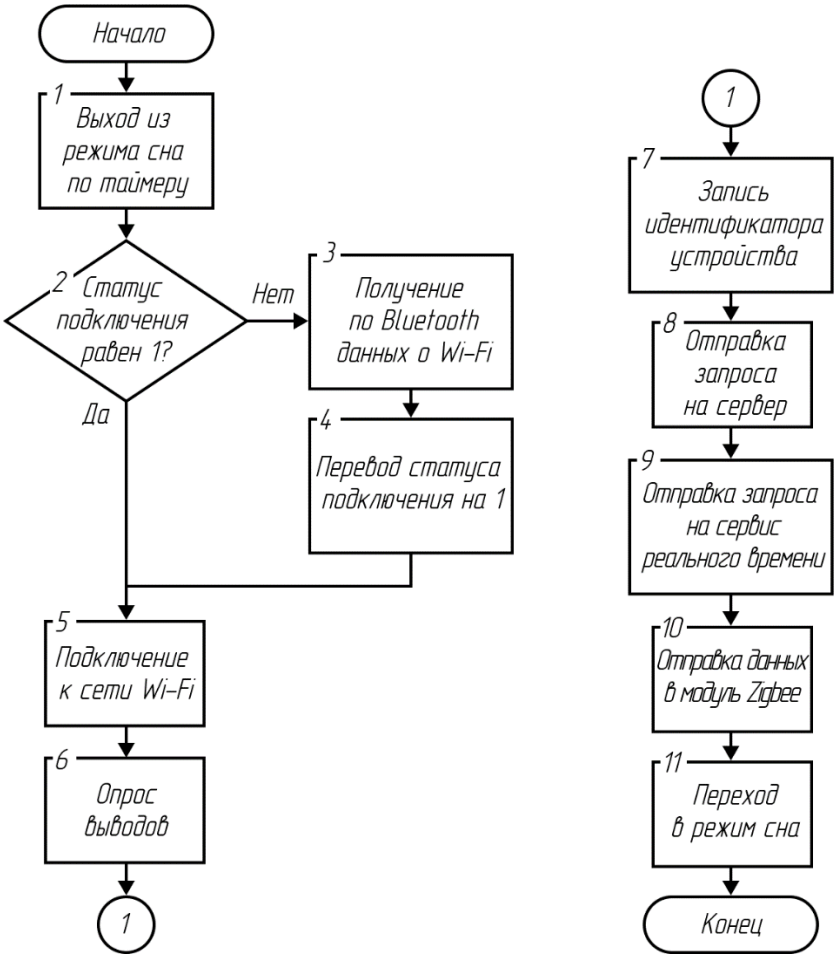

Рисунок 2 – Блок-схема основного алгоритма работы устройства

При первом включении шлюз не имеет на себе никаких данных и не может подключиться к Wi-Fi сети. Он находится в режиме ожидания, то есть статус подключения будет равен логическому нулю, а строки SSID и пароль будут пустыми. В этом режиме шлюз включает модуль Bluetooth, чтобы стать видимым для мобильного приложения. Устройство продолжает находиться в режиме ожидания до тех пор, пока не получит от приложения данные локальной сети: SSID и пароль.

После получения данных о сети модуль Wi-Fi обрабатывает их и записывает в память, а также выполняет подключение к Wi-Fi. При успешном исходе статус подключения устанавливается в логическую единицу. Это важно для того, чтобы в будущем при отключении и повторном включении устройство не переходило в режим ожидания, а сразу подключалось к Wi-Fi.

Когда соединения с Wi-Fi установлено, шлюз начинает собирать данные. Он опрашивает выводы, к которым подключены кнопки, и интерфейс, по которому он соединен с модулем Zigbee. Модуль Wi-Fi собирает внутреннюю информацию о идентификаторе устройства для формирования запроса на сервер.

После сбора данных шлюз устанавливает соединение с сервером. Устройство отправляет на сервер запрос, который содержит идентификатор устройства и другие данные, в зависимости от устройств в сети Zigbee, например, текущий уровень освещенности. После этого от сервера приходит ответ, который содержит целевой уровень освещенности и время сна. Модуль Wi-Fi принимает ответ и парсит его. Данные из ответа имеют разное назначение: часть из них предназначена для модуля Wi-Fi, эти данные он записывает во внутренние переменные, остальные – для модуля Zigbee, они отправляются по интерфейсу взаимодействия.

К шлюзу могут быть подключены устройства, работа которых зависит от текущего времени. Чтобы модуль Wi-Fi можно было настроить для различных сценариев в зависимости от него, он подключается к сервису реального времени.

В постоянной работе шлюза нет необходимости, достаточно выполнять весь вышеперечисленный функционал с определенным периодом. Для этого после выполнения всех функций модуль Wi-Fi переходит в режим сна. Это также снижает энергопотребление устройства.

Перевести устройство в режим ожидания также можно с мощью кнопки Reset, которая будет располагаться на корпусе устройства. По нажатию на кнопку происходит сброс состояния подключения, а также очистка прежних данных о сети Wi-Fi. Блок-схема алгоритма обработки нажатия кнопки Reset показан на рисунке 3.

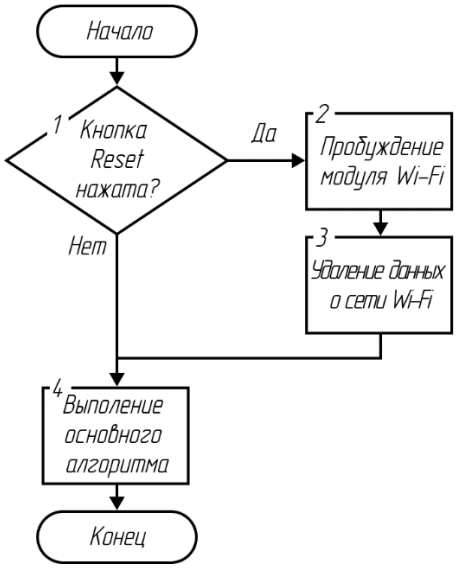

Рисунок 3 – Блок-схема алгоритма обработки нажатия кнопки Reset

На стороне сервера будет присутствовать страничка с формами или с переключателями, через которую можно будет задавать некоторые данные, например, время сна. Также на сервере будет несколько скриптов.

Одни из скриптов будет обрабатывать запрос от шлюза, добавлять или обновлять информацию в базе данных. Этот же скрипт будет формировать ответ из имеющихся данных в базе и отправлять их обратно в шлюз. Другой

скрипт будет обрабатывать информацию из формы на страничке. Блок-схемы алгоритмов скриптов показаны на рисунках 4-5.

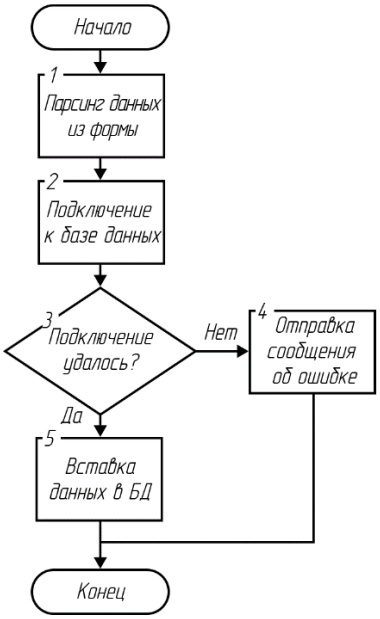

Рисунок 4 – Блок-схема алгоритма скрипта обработки формы

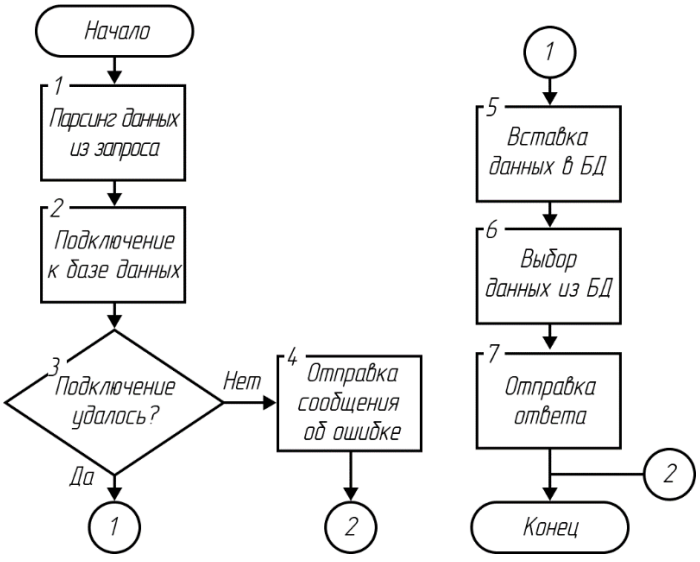

Рисунок 5 – Блок-схема алгоритма скрипта обработки запроса

Алгоритм работы мобильного приложение состоит из подключения к шлюзу по Bluetooth и по нажатию на копку формирования строки из данных в

полях и отправки их в устройство. Блок-схема алгоритма приведена на рисунке 6.

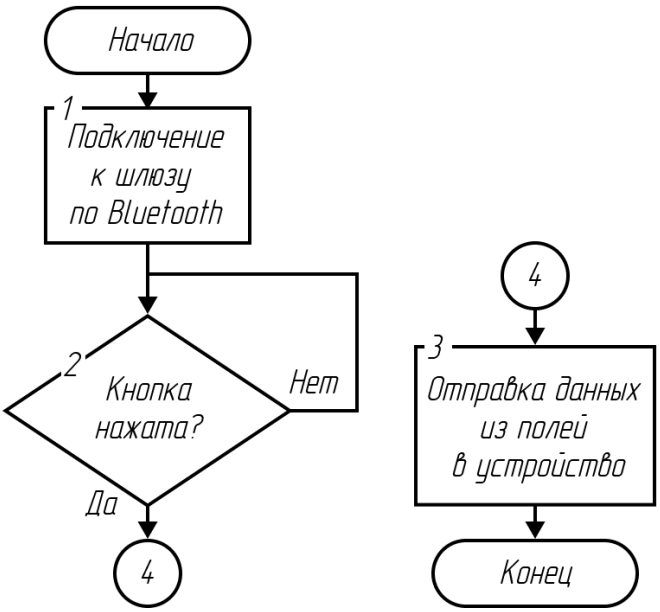

Рисунок 6 – Блок-схема алгоритма работы мобильного приложения

Таким образом, алгоритмы работы основного и вспомогательного ПО учитывают возможности поддерживаемых устройством беспроводных технологий.

4 Разработка схемы электрической принципиальной

Принципиальная схема представляет собой графическое представление электрической цепи. Он показывает компоненты цепи в виде символов и их соединения через провода или другие проводящие пути. Принципиальные схемы используются для проектирования и документирования электрических цепей, а также для устранения неполадок и их ремонта. Они являются важным инструментом для инженеров, техников и любителей, работающих с электроникой.

Разработка электрической схемы начинается с замены блоков структурной схемы на реальные элементы. Также к ним добавляется необходимая обвязка [3].

В качестве модуля Zigbee был выбран модуль E18-MS1-PCB.

Zigbee E18-MS1-PCB — это модуль беспроводной связи Zigbee предназначенный для использования в различных приложениях Интернета вещей (IoT), включая домашнюю автоматизацию, промышленную автоматизацию и умные города. Модуль основан на популярном протоколе Zigbee и обеспечивает надежную и безопасную беспроводную связь для устройств IoT [11].

Модуль E18-MS1-PCB основан на микросхеме CC2530 от Texas Instruments и совместим со стандартом Zigbee 3.0. CC2530 – это система на кристалле (SoC), разработанная для сетевых приложений с беспроводными датчиками. Он является частью семейства устройств CC253x, основанных на архитектуре микроконтроллера 8051 и оснащенных встроенным радиопередатчиком IEEE 802.15.4 с частотой 2,4 ГГц [11].

CC2530 включает 256 КБ флэш-памяти и 8 КБ ОЗУ, которые можно использовать для хранения кода и данных. Он также имеет множество встроенных периферийных устройств, включая таймеры, последовательные интерфейсы и АЦП. Радиотрансивер поддерживает ряд схем модуляции и

скоростей передачи данных, что позволяет ему обмениваться данными с другими устройствами, совместимыми со стандартом IEEE 802.15.4 [11].

Максимальная выходная мощность данного модуля Zigbee 20 дБм. Он имеет PCB антенну и поддерживает дальность передачи до 100 метров на открытом воздухе. Модуль питается от источника питания 3,3 В и имеет низкое энергопотребление – 28 мА в режиме передатчика и 27 мА в режиме приемника, что делает его идеальным для устройств как с сетевым, так и с батарейным питанием [11].

Кроме того, модуль E18-MS1-PCB имеет несколько интерфейсов, включая UART, SPI, I2C и GPIO, что упрощает связь с другими устройствами и датчиками. Он также поддерживает обновления прошивки OTA (по воздуху), что позволяет удаленно обновлять прошивку модуля [11].

Для работы этого модуля необходима дополнительная обвязка. Согласно рекомендациям из технической документации на данный модуль, в цепь питания должны устанавливаться 2 конденсатора параллельно: электролитический емкостью 100 мкФ для гашения низкочастотный пульсаций и керамический емкостью 0,1 мкФ для гашения высокочастотных пульсаций. Питания подается к выводу под номером 2, которые называется VCC. Схема питания показана на рисунке 7.

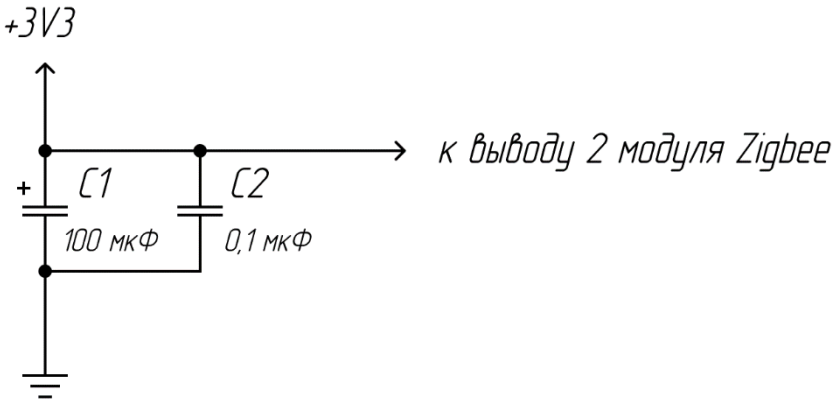

Рисунок 7 – Схема питания модуля Zigbee

В качестве конденсаторов будут использоваться модели EEUFR1E101B емкостью 100 мкФ и GRM155R71C104KA88D емкостью 0,1 мкФ. Они рассчитаны на работу с напряжениями 25 В и 16 В соответственно.

Для задержки запуска микроконтроллера при включении питания используется цепь сброса. Она подключается к выводу под номером 24, который называется RESET. Цепь сброса включается для начальной установки всех внутренних систем процессора в момент включения питания. Производитель рекомендует использовать резистор сопротивлением 1 кОм и конденсатор емкостью 100 пФ в цепи сброса. (рисунок 8).

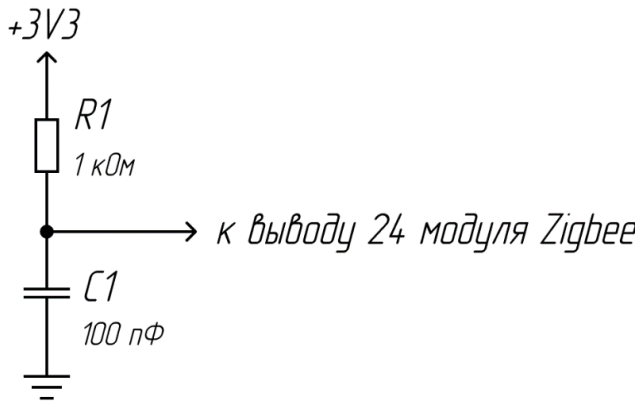

Рисунок 8 – Схема цепи сброса модуля Zigbee

Максимальное напряжение, которое будет прикладываться к конденсатору равно напряжению питания, то есть 3,3 В. Тогда используемая модель конденсатора будет CC0603JRNPO9BN101.

Максимальная мощность будет выделяться на резисторе лишь в момент подключения питания, так как конденсатор будет разряжен, напряжение на резисторе будет равно напряжению питания, тогда мощность можно рассчитать по формуле 1.

$$P = \frac{U_{\text{пит}}^2}{R} = \frac{3{,}3^2}{1000} = 0{,}011 \text{ Вт} \quad (1)$$

где $U_{пит}$ – напряжение питания, В;

R – сопротивление резистора, Ом.

В качестве резистора сопротивлением 1 кОм будет использоваться модель RC0402FR-071KL.

Для программирования модуля необходим разъем на плате. Программатор использует 5 выводов: DC, DD, RESET, а также питание и заземление. Причем, согласно технической документации, рекомендуется вывод DC подтягивать к питания через резистор сопротивлением 1 кОм. В качестве разъема будет использоваться модель XH2.54 5PIN. Схема подключения разъема программатора показана на рисунке 9.

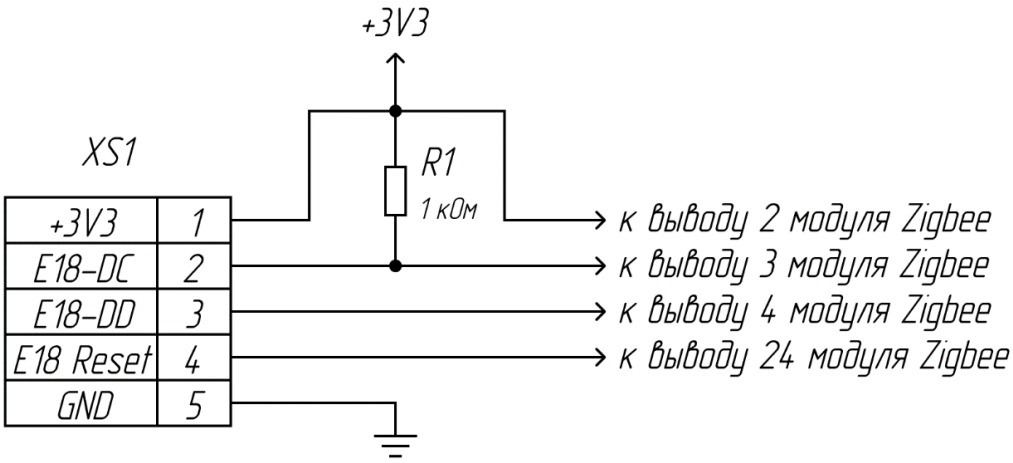

Рисунок 9 – Схема подключения разъема для программатора модуля Zigbee

Максимальная мощность будет выделяться на резисторе лишь в момент, когда напряжение на выводе E18-DC будет равно нулю. Тогда падение

напряжения на резисторе будет равно напряжению питания, и мощность можно будет рассчитать по формуле 2.

$$P = \frac{U_{\text{пит}}^2}{R} = \frac{3{,}3^2}{1000} = 0{,}011 \text{ Вт} \quad (2)$$

где $U_{пит}$ – напряжение питания, В;

R – сопротивление резистора, Ом.

В качестве резистора сопротивлением 1 кОм будет использоваться модель RC0402FR-071KL.

К микроконтроллеру подключается кнопка, которая нужна для внутренней настройки. Кнопка будет подключатся через подтягивающий резистор к земле. Таким образом, пока кнопка не нажата, на вывод микроконтроллера будет приходить сигнал, равный логическому нулю, а при нажатии на кнопку на выводе будет устанавливаться логическая единица.

Для расчета подтягивающего резистора воспользуемся выражением (3).

$$R = \frac{U_{\text{лог.0}}}{I_{\text{ист}}} \quad (3)$$

где $U_{лог.\,0}$ – максимальное напряжение вывода микроконтроллера, равное логическому нулю, В;

$I_{ист}$ – максимальный ток вывода микроконтроллера, А [3].

Указанные выше данные не отображены в технической документации, поэтому положим, что максимальный уровень логического нуля равен половине напряжения питания, то есть 1,65 В. Максимальный ток вывода на большинстве микроконтроллеров не превышает 40 мА. Это справедливо для

ESP32 и микроконтроллеров Arduino. Тогда минимальное сопротивление подтягивающего резистора будет рассчитываться по формуле 4.

$$R = \frac{U_{\text{лог.0}}}{I_{\text{ист}}} = \frac{1{,}65}{0{,}04} = 41{,}25 \text{ Ом} \quad (4)$$

Для снижения количества различных используемых изделий будем использовать резистор сопротивлением 10 кОм. Тогда ток, протекающий через резистор, рассчитывается по формуле 5.

$$I = \frac{U_{\text{лог.0}}}{R} = \frac{1{,}65}{10000} = 0{,}165 \text{ мА} \quad (5)$$

Выделяемая мощность на резисторе рассчитывается по формуле 6.

$$P = I^2 \cdot R = (0{,}165 \cdot 10^{-3})^2 \cdot 10000 = 0{,}27 \text{ мВт} \quad (6)$$

где I – ток, протекающий через резистор, А;

R – сопротивление резистора, Ом.

В качестве резистора сопротивлением 10 кОм будем использовать модель RC0402FR-0710KL.

Кнопка должна подключаться к цифровому выводу. В качестве такого будем использовать вывод номер 17. Этот вывод может работать как вход, так и выход, а также подключен к АЦП и ШИМ (рисунок 10).

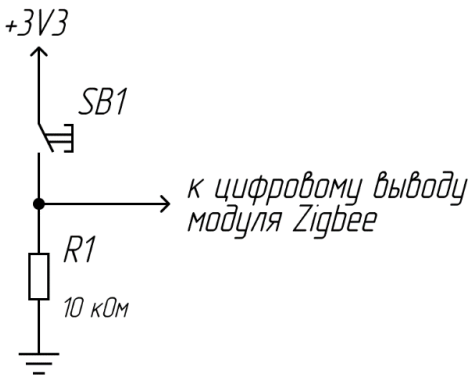

Рисунок 10 – Схема подключения кнопки к модулю Zigbee

В качестве модуля Wi-Fi будет использоваться микроконтроллер ESP32 Wroom. Одной из его особенностей является то, что он также поддерживает Bluetooth, в связи с этим отпадает нужда в наличие отдельного модуля Bluetooth.

ESP32 – это недорогой микроконтроллер с низким энергопотреблением, разработанный компанией Espressif Systems. Модуль питается от источника питания 3,3 В. Он сделан по технологии «система на кристалле» (SoC) и оснащен двухъядерным процессором Xtensa LX6 с тактовой частотой до 240 МГц, который позволяет ему одновременно запускать два процесса, что делает его более эффективным, чем другие одноядерные микроконтроллеры. Он также имеет 520 КБ SRAM, 448 КБ ROM и поддерживает до 16 МБ флэш-памяти. Он также имеет множество периферийных устройств, включая SPI, I2C, UART, АЦП, ЦАП и ШИМ [13].

ESP32 имеет встроенные функции Wi-Fi и Bluetooth, что делает его идеальным выбором для проектов, требующих беспроводного подключения. Он поддерживает Wi-Fi на частотах 2,4 ГГц и 5 ГГц и Bluetooth 4.2 классический и BLE [13].

Модуль Wi-Fi может работать в следующих режимах: станция, программная точка доступа и станция плюс точка доступа. Режим станции позволяет ESP32 подключаться к существующей беспроводной сети, а режим

программной точки доступа позволяет ESP32 функционировать как точка доступа Wi-Fi. В режиме «станция плюс точка доступа» ESP32 может подключаться к существующей сети и предоставлять доступ к другим устройствам [13].

Возможности Bluetooth ESP32 позволяют ему связываться с другими устройствами, используя Bluetooth с низким энергопотреблением (BLE) или Bluetooth Classic. Модуль Bluetooth может действовать как клиент или сервер, что делает его пригодным для широкого круга приложений. Он поддерживает как центральную, так и периферийную роль BLE и может работать с устройствами Bluetooth Classic, такими как смартфоны, планшеты и ноутбуки.

В целом, возможности Wi-Fi и Bluetooth делают ESP32 универсальной платформой для разработки подключенных устройств, которые могут взаимодействовать по беспроводной сети с другими устройствами и сетями.

По аналогии с модулем Zigbee в цепи питания присутствуют 2 конденсатора, которые установлены параллельно. Они также служат для гашения пульсаций. Согласно рекомендациям технической документации их емкости должны равняться 100 мкФ и 0,1 мкФ. Схема питания ESP32 показана на рисунке 11.

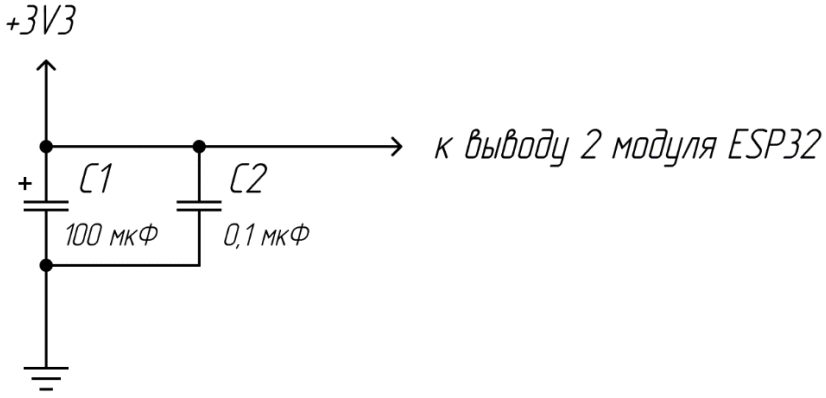

Рисунок 11 – Схема цепи питания модуля ESP32

В качестве конденсаторов будут использоваться модели EEUFR1E101B емкостью 100 мкФ и GRM155R71C104KA88D емкостью 0,1 мкФ. Они рассчитаны на работу с напряжениями 25 В и 16 В соответственно.

Для задержки запуска микроконтроллера при включении питания используется цепь сброса. Она подключается к выводу под номером 3, который называется EN. Цепь сброса включается для начальной установки всех внутренних систем процессора в момент включения питания. Производитель рекомендует использовать резистор сопротивлением 10 кОм и конденсатор емкостью 0,1 мкФ в цепи сброса. (рисунок 12).

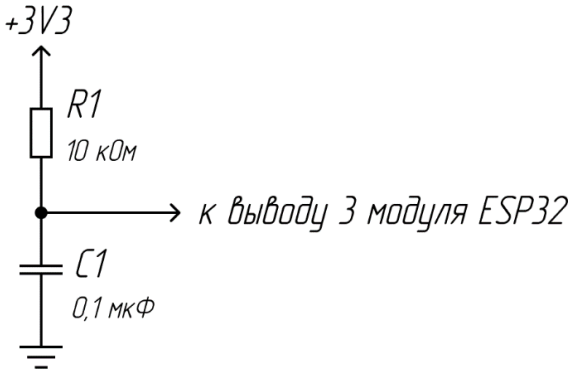

Рисунок 12 – Схема цепи сброса модуля ESP32

Максимальное напряжение, которое будет прикладываться к конденсатору равно напряжению питания, то есть 3,3 В. Тогда используемая модель конденсатора будет GRM155R71C104KA88D.

Максимальная мощность будет выделяться на резисторе лишь в момент подключения питания, так как конденсатор будет разряжен, напряжение на резисторе будет равно напряжению питания, тогда мощность можно рассчитать по формуле 7.

$$P = \frac{U_{\text{пит}}^2}{R} = \frac{3{,}3^2}{10000} = 0{,}0011 \text{ Вт} \quad (7)$$

где $U_{пит}$ – напряжение питания, В;

R – сопротивление резистора, Ом.

В качестве резистора сопротивлением 10 кОм будет использоваться модель RC0402FR-0710KL.

Программирование модуля ESP32 происходит через интерфейс UART. Для этого на плате будет предусмотрен разъем. Интерфейс для своей работы требует 2 провода: 1 для передачи данных и еще 1 для приема. Таким образом, программатор подключается к ESP32, используя 4 вывода: TXD0, RXD0, питание и заземление (рисунок 13). В качестве разъема будет использоваться модель 53047-0410 4PIN.

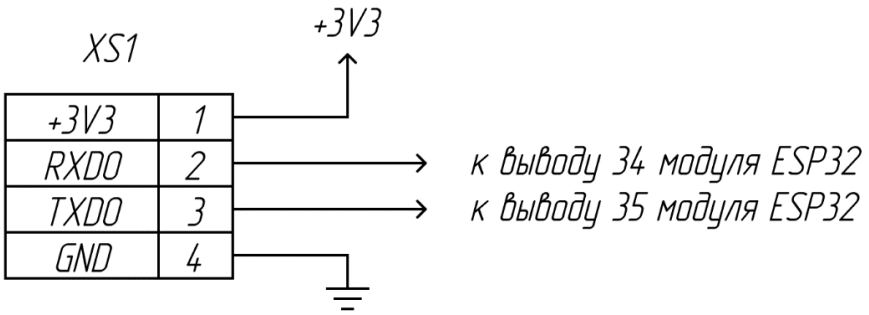

Рисунок 13 – Схема подключения разъема для программирования модуля ESP32

К микроконтроллеру подключаются кнопки, которые нужны для внутренней настройки, а также для сброса настроек. Кнопки будут подключатся через подтягивающий резистор к земле. Таким образом, пока кнопка не нажата, на вывод микроконтроллера будет приходить сигнал, равный логическому нулю, а при нажатии на кнопку на выводе будет устанавливаться логическая единица.

Расчета подтягивающих резисторов происходит аналогично расчету для модуля Zigbee. Максимальное напряжение, которое соответствует уровню

логического нуля, согласно технической документации, равно 25% от напряжения питания, то есть 0,825 В. Максимальный ток вывода равен 40 мА. Тогда минимальное сопротивление подтягивающего резистора будет рассчитываться по формуле 8.

$$R = \frac{U_{\text{лог.0}}}{I_{\text{ист}}} = \frac{0{,}825}{0{,}04} = 20{,}625 \text{ Ом} \quad (8)$$

Для снижения количества различных используемых изделий будем использовать резистор сопротивлением 10 кОм. Тогда ток через резистор рассчитывается с помощью выражения (9).

$$I = \frac{U_{\text{лог.0}}}{R} = \frac{0{,}825}{10000} = 0{,}083 \text{ мА} \quad (9)$$

Выделяемая мощность на резисторе рассчитывается по формуле 10.

$$P = I^2 \cdot R = (0{,}083 \cdot 10^{-3})^2 \cdot 10000 = 0{,}69 \text{ мВт} \quad (10)$$

где I – ток, протекающий через резистор, А;

R – сопротивление резистора, Ом.

В качестве резистора сопротивлением 10 кОм будет использоваться модель RC0402FR-0710KL.

Кнопки должны подключаться к цифровым выводам. ESP32 имеет 30 выводов, которые подключены к различной периферии. Одним из важных критериев выбора выводов, к которым подключаются кнопки, является поддержка RTC GPIO. Эти выводы маршрутизированы в подсистему с низким энергопотреблением RTC, что можно использовать, когда ESP32 находится в состоянии глубокого сна. Эти RTC GPIO могут использоваться для выхода

ESP32 из глубокого сна, когда работает сопроцессор Ultra Low Power (ULP). Такими выводами, например, являются выводы под номерами 13, 16 и 23. Именно к ним и будут подключаться кнопки. (рисунок 14).

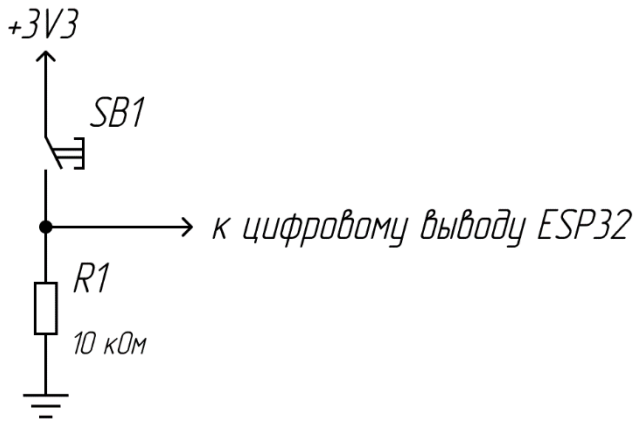

Рисунок 14 – Схема подключения кнопки к модулю ESP32

Поскольку оба модуля имеют интерфейс UART, обмен данными между ними будет происходить именно по этому интерфейсу.

UART (Universal Asynchronous Receiver-Transmitter) – универсальный асинхронный приемник-передатчик, который представляет собой тип протокола связи, используемый для последовательной связи между двумя устройствами. Проще говоря, UART позволяет передавать и принимать данные по одному биту за раз между двумя устройствами.

UART обычно используется во встроенных системах и микроконтроллерах для связи с другими устройствами, такими как датчики, дисплеи и другие контроллеры. Протокол является асинхронным, что означает, что для синхронизации связи между двумя устройствами не требуется тактовый сигнал. Вместо этого передатчик отправляет стартовый бит, чтобы сигнализировать о начале кадра данных, за которым следуют биты данных и необязательные биты четности, и стоповые биты.

Связь UART использует простой двухпроводной интерфейс, один для передачи данных (TX), а другой для приема данных (RX). Скорость связи

определяется скоростью передачи данных, которая представляет собой количество битов, которые могут быть переданы в секунду.

ESP32 имеет 3 интерфейса UART. Одни из них, UART0, используется для программирования и всегда является активным. Остальные 2 сконфигурированы заранее на определенные выводы, но могут быть переназначены программно. Для обмена данными будет использоваться UART2. Вывод RX имеет номер 27, TX – 28.

Модуль Zigbee имеет всего одни интерфейс UART, который расположен на выводах под названиями P1.5 (TX) и P1.4 (RX), под номерами 10 и 11 соответственно.

Для передачи данных по интерфейсу UART вывод RX микроконтроллера ESP32 должен подключаться к выводу TX модуля Zigbee, а вывод TX микроконтроллера ESP32 – к выводу RX модуля Zigbee (рисунок 15).

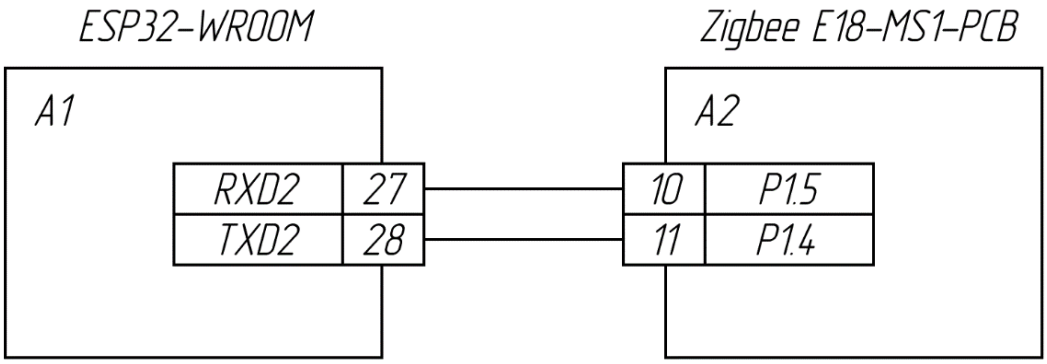

Рисунок 15 – Схема подключения модулей ESP32 и Zigbee по UART

Индикация наличия питания будет происходить с помощью красного светодиода HLMP-3301. Светодиод включается в цепь последовательно с токоограничивающим резистором. Его номинал должен быть таким, чтобы протекающего тока хватило для свечения светодиода. Яркость или сила света светодиода в данном случае не важна, для индикации питания важно лишь само наличие свечения.

Для расчета сопротивления резистора нужно знать падение напряжения на нем и протекающий через него ток. Падение напряжения на резисторе рассчитывается по второму закону Кирхгофа, как разность между напряжением питания и падением напряжения на светодиоде. Падение напряжения на светодиоде будет зависеть от протекающего через него ток. Его можно вычислить по ВАХ (рисунок 16). Пусть протекающий ток будет равен 5 мА, тогда, согласно ВАХ, падение напряжения составит 1,7 В. При таком токе сила света будет равна 0,8 кд.

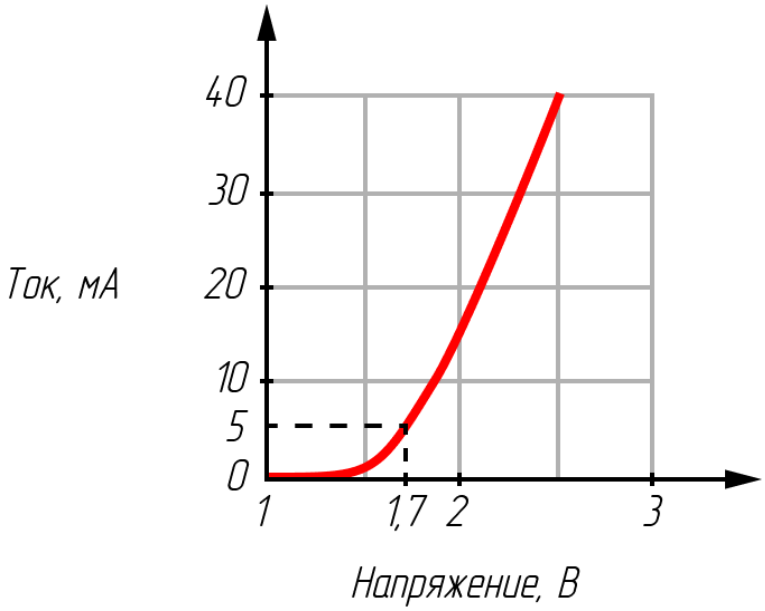

Рисунок 16 – Вольт-амперная характеристика красного светодиода

Расчет сопротивления резистора происходит по формуле 11.

$$R = \frac{U_{пит} - U_{VD}}{I_{VD}} = \frac{3{,}3 - 1{,}7}{0{,}005} = 320 \text{ Ом} \qquad (11)$$

где $U_{пит}$ – напряжение питания, В;

U_{VD} – падение напряжения на светодиоде, В;

I_{VD} – ток, протекающий через светодиод, А.

Таким образом, сопротивление резистора будет равно 320 Ом. Выделяемая мощность на резисторе рассчитывается по формуле 12.

$$P = I^2 \cdot R = 0{,}005^2 \cdot 320 = 0{,}008 \text{ Вт} \quad (12)$$

где I – ток, протекающий через резистор, R – сопротивление резистора.

Такая маленькая выделяемая мощность говорит о том, что можно использовать SMD резистор, то есть тот, который монтируется на поверхность. Ближайший номинал резистора согласно стандартному ряду E24 равен 330 Ом. В качестве него будет использоваться резистор RC0402FR-07330RL.

Для выбора блока питания необходимо рассчитать потребление тока всех элементов схемы. Согласно технической документации, максимальное потребление тока модулем ESP32 равно 260 мА, модулем Zigbee E18 равно 28 мА, светодиод потребляет еще 5 мА. Таким образом, максимальное потребление тока всей цепи будет равно 293 мА. Это значение – минимальное, на которое должен быть рассчитан блок питания.

Питание устройства будет осуществляться двумя способами: от сети 220 В и от мобильных блоков питания.

Для понижения напряжения с 220 В до 3,3 В будет использоваться преобразователь HLK-5M03. Некоторые технические характеристики представлены в таблице 1.

Таблица 1 – Технические характеристики HLK-5M03

Параметр	Значение
Входное напряжение, В	85-264
Выходное напряжение, В	3,1-3,5
Максимальный входной ток, мА	200
Максимальный выходной ток, мА	1500

Включение данного модуля рекомендуется с дополнительной обвязкой. Согласно техническим требованиям, рекомендуется добавить предохранитель на 0,5 А, варистор 7D471K, а также конденсатор емкостью 100 мкФ. Подключение к сети будет происходить с помощью кабеля питания через разъем RAPC322X. Схема питания от сети 220 В показана на рисунке 17.

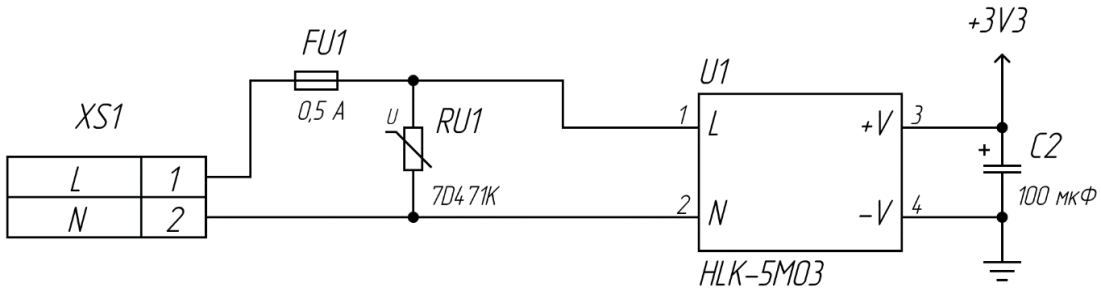

Рисунок 17 – Схема подключения модуля HLK-5M03

В качестве предохранителя будет использоваться съемный плавкий предохранитель 0154.500DRT. В качестве варистора будет использоваться модель MOV-07D471KTR. В качестве конденсаторов будет использоваться модели EEUFR1E101B емкостью 100 мкФ.

Для работы от мобильных блоков питания в устройстве предусмотрен разъем USB Type-C. Такой разъем имеет 12 контактов, но, поскольку он двухсторонний, эти контакты дублируются. Так как разъем будет использоваться только для питания, подключены будут всего 8 контактов: VBUS и GND с каждой из сторон [19].

Мобильные блоки питания выдают 5 В, а модули ESP32 и Zigbee работают на напряжении 3,3 В, следовательно в схеме будет присутствовать микросхема для понижения напряжения. В качестве такой микросхемы будет использоваться LM3904. Некоторые технические характеристике данной микросхемы представлены в таблице 2.

Таблица 2 – Технические характеристики LM3940

Параметр	Значение
Входное напряжение, В	4,5-5,5
Выходное напряжение, В	3,2-3,4
Выходной ток, А	1

Производитель рекомендует устанавливать по одному конденсатору между входом и землей, выходом и землей емкостями 0,47 мкФ и 100 мкФ соответственно. Схема питания через разъем USB Type-C показана на рисунке 18. В качестве разъема будет использоваться модель 105450-0101.

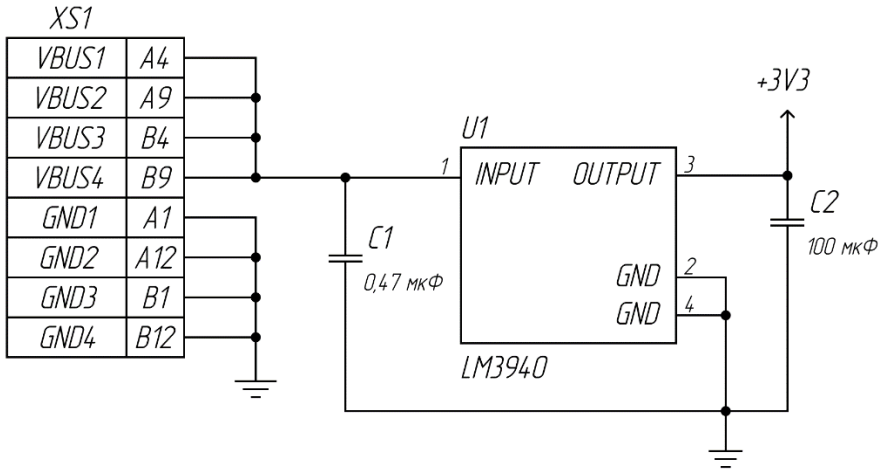

Рисунок 18 – Схема питания через разъем USB Type-C

В качестве конденсаторов будут использоваться модели EEUFR1E101B емкостью 100 мкФ и GRM188R71C474KA88D емкостью 0,47 мкФ.

Схема электрическая принципиальная показана на рисунке 19.

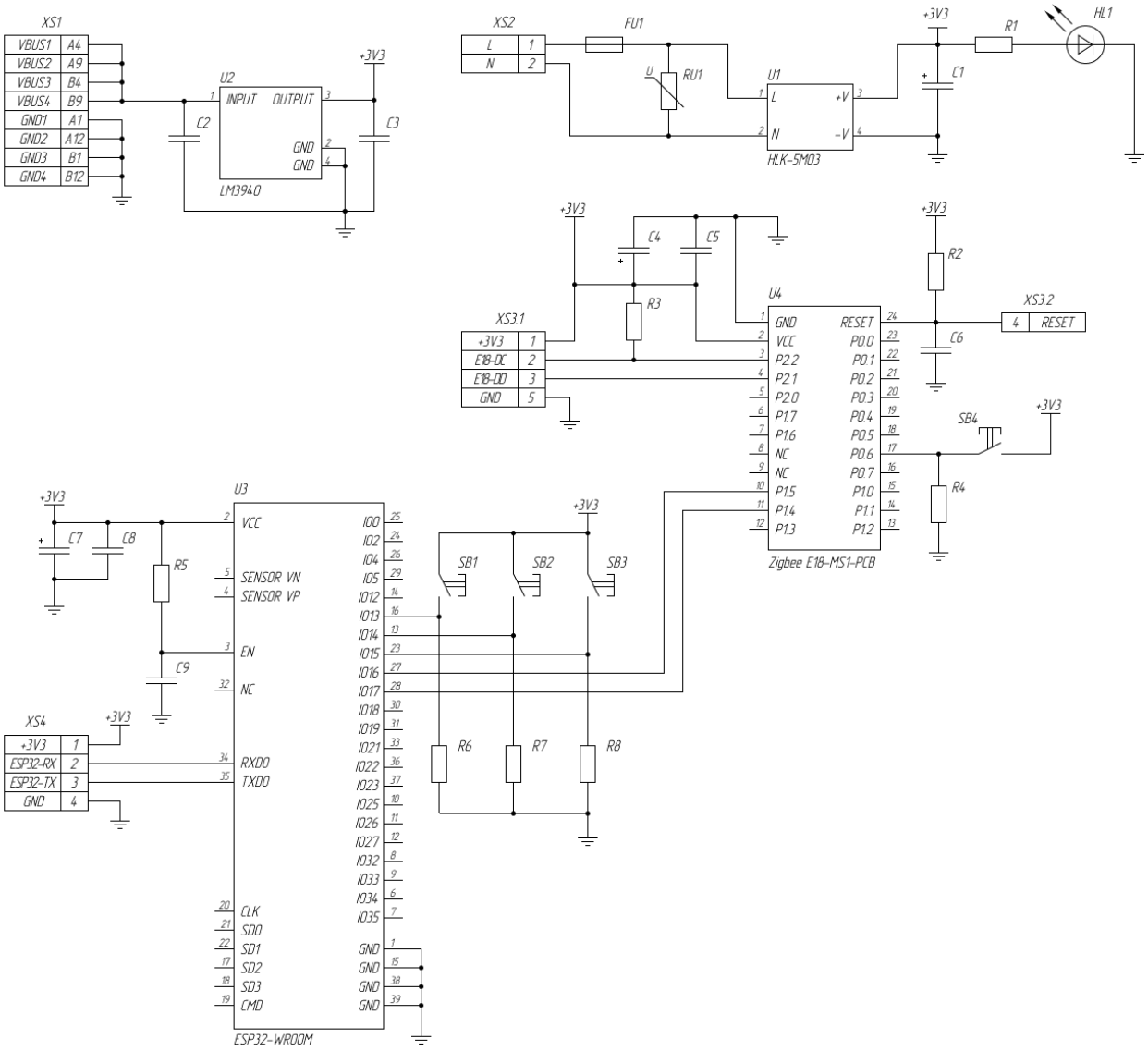

Рисунок 19 – Схема электрическая принципиальная

Таким образом, блоки структурной схемы заменены на реальные элементы. Обвязка модулей выполнена в соответствии с рекомендациями производителей, элементы для подключения кнопок и светодиода расчитаны по формулам.

5 Разработка ПО для модуля Wi-Fi

Поскольку в качестве модуля Wi-Fi выступает микроконтроллер ESP32, программирование будет происходить в среде разработки Arduino IDE.

Arduino IDE – это инструмент разработки программного обеспечения с открытым исходным кодом, который упрощает процесс написания, компиляции и загрузки кода на платы Arduino и другие, в том числе ESP32. Он предоставляет интуитивно понятный графический интерфейс для написания и редактирования кода, а также платформу для управления и загрузки скетчей на платы [8].

Arduino IDE основана на Processing IDE и использует упрощенную версию языка программирования C++. Он включает в себя ряд встроенных библиотек и функций, упрощающих взаимодействие с различными датчиками, двигателями и другими электронными компонентами, обычно используемыми в проектах Arduino [16].

Arduino IDE является кроссплатформенной и работает на Windows, macOS и Linux. Он совместим с широким спектром плат Arduino, включая популярные Arduino Uno, Mega и Nano. Кроме того, его можно расширить с помощью сторонних библиотек, что обеспечивает большую функциональность и гибкость в программировании. Поддержку ESP32 можно включить в менеджере плат [8].

Программа будет написана с использованием функций. Все функции связаны с блоками из блок-схемы алгоритма работы устройства. Список функций и их назначение представлены в таблице 3.

Таблица 3 – Функции и их назначение

Функция	Назначение
get_wifi_settings()	Получение данных о сети Wi-Fi от мобильного приложения
connect_to_wifi()	Подключение к сети Wi-Fi
get_id_esp32()	Получение идентификатора чипа ESP32
send_json_request()	Отправка запроса на сервер
get_local_time()	Получени времени с сервиса реального времени
send_data_to_uart()	Отправка сообщение модулю Zigbee
go_to_deep_sleep()	Перех в режим глубокого сна
reset()	Сброс данныз о сети Wi-Fi

Все используемые функции типа void, то есть они не возвращают никаких значений. Также ни одна из функций не принимает никакие параметры.

Первое, что делает шлюз в начале своей работы – получает данные о сети Wi-Fi посредством подключения к мобильному приложению по Bluetooth. Это происходит в функции get_wifi_settings(). Для работы с Bluetooth ESP32 нуждается в подключении библиотеки BluetoothSerial.h [16].

Библиотека BluetoothSerial.h использует последовательный интерфейс платы ESP32 для отправки и получения данных по Bluetooth. Он предоставляет функции для инициализации модуля Bluetooth, установки имени устройства, включения и отключения соединения Bluetooth, а также отправки и получения данных [16].

Библиотека также предоставляет функции обратного вызова, которые можно использовать для получения уведомлений о таких событиях, как успешное подключение, отключение или получение пакета данных. Это позволяет ESP32 реагировать на события в режиме реального времени, что делает его пригодным для интерактивных приложений [16].

Для работы модуля Bluetooth необходимо создать объект типа BluetoothSerial, он может иметь любое имя, я назвал его ESP_BT. Именно к

этому объекту будут применяться функции используемой библиотеки. В этом проекте понадобятся следующие функции:

– begin() – эта функция включает модуль Bluetooth ESP32 и в качестве параметра принимает название устройства;

– available() – эта функция показывает, подключено ли к ESP32 какое-либо устройство и возвращает логическую единицу, если подключено, в противном случае – логический ноль;

– readString() – эта функция считывает символы, принятые от другого устройства, и возвращает объект типа String.

Шлюз принимает от мобильного приложения идентификатор сети и пароль, разделенные запятой, то есть в формате «ssid,password». Она записывается в локальную переменную s_ssid_pass типа String. Использование строкового типа позволяет легко разделить принятую строку на две: s_ssid и s_pass, с помощью функции substring(). Эта функция записывает в переменную новую строку, которая является обрезанной исходной строкой. Функция принимает 1 или 2 аргумента, которые показывают, с какого по какой символы обрезать. Подсчет символов происходит с помощью функции length(), а номер символа, с которого нужно начинать обрезку определяется в цикле, в котором каждый последующий символ сравнивается с запятой. Как только программа доходит до запятой, ее позиция записывается в локальную переменную i, далее исходная строка разделяется на две.

Важно, что функция работает только в том случае, если данных о Wi-Fi нет в памяти. Для регистрации состояния данных о сети сущестует глобальная переменная wifi_data_state типа bool. По умолчанию она равна нулю. Когда микроконтроллер получил данные о сети, он переводит состояние данных в единицу.

Полученные строки нужно перевести в массивы символов, потому что библиотека WiFi.h не умеет работать с ними. Для перевода в формат char

используется функция toCharArray(). Она принимает в себя название массива и количество символов.

Блок-схема алгоритма получения и записи данных о сети показана на рисунке 20.

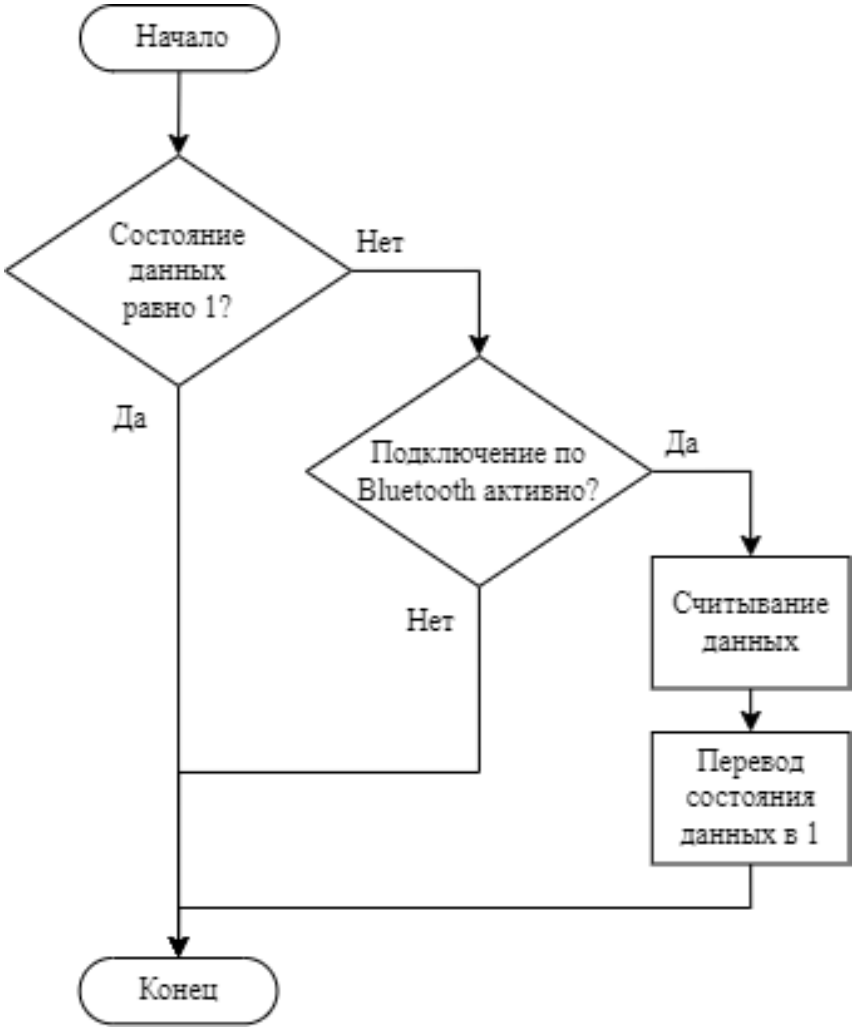

Рисунок 20 – Блок-схема алгоритма получения данных о сети Wi-Fi

Код данной функции представлен в листинге 1.

Листинг 1 – Функция получения данных о сети Wi-Fi

```
void get_wifi_settings() {
  String s_ssid_pass;
  String s_ssid;
```

```
  String s_pass;

  while (!wifi_data_state) {

    while (ESP_BT.available()) {
      s_ssid_pass = ESP_BT.readString();

      for (int i = 0; i < s_ssid_pass.length(); i++) {
        if (s_ssid_pass.substring(i, i + 1) == ",") {
          s_ssid = s_ssid_pass.substring(0, i);
          s_pass = s_ssid_pass.substring(i + 1);

          delay(1000);
          wifi_data_state = true;

          s_ssid.toCharArray(ch_ssid, 20);
          s_pass.toCharArray(ch_pass, 20);

          Serial.println("Recieved wifi settings");

          break;
        }
      }
    }
  }
}
```

Основная функция в работе шлюза – подключение по Wi-Fi. Оно происходит с помощью функции connect_to_wifi(). Именно благодаря такому подключению устройство будет соединяться с сервером для отправки данных и получения ответа. Для работы с Wi-Fi ESP32 нуждается в подключении библиотеки WiFi.h.

Библиотека WiFi.h – это основная библиотека в ядре Arduino для ESP32. Она предоставляет API для подключения ESP32 к сети Wi-Fi и доступа к информации, связанной с сетью.

Библиотека предоставляет функции для сканирования доступных сетей Wi-Fi, подключения к сети с использованием имени пользователя и пароля, отключения от сети и получения текущего состояния сети. Он также предоставляет функции для настройки режима сети Wi-Fi и настройки точки доступа.

Функция выполняется только в том случае, когда состояние данных о сети равно единице.

Для работы с модулем Wi-Fi ESP32 использовались следующие функции библиотеки WiFi.h:

– begin() – эта функция включает модуль Wi-Fi ESP32 и производит подключение к сети, в качестве параметра принимает SSID и пароль;

– status() – эта функция показывает, состояние подключения к сети, в случае успешного исхода возвращает «WL_CONNECTED»;

– localIP() – эта функция возвращает локальный IP-адрес, назначенный микроконтроллеру.

Подключение к Wi-Fi происходит не сразу, поэтому внутри функции используется цикл while, который будет проверять статус подключения каждые 500 мс. Если данные о сети будут неправильные, то микроконтроллер будет выполнять этот цикл бесконечно. Чтобы этого не происходило, количество попыток ограничено до 10. Микроконтроллер выходит из цикла при успешном подключении к сети или по истечению количества попыток.

После выхода из цикла программа проверяет статус подключения. Если подключение состоялось, то в последовательный порт выводится соответствующее сообщение вместе с локальным IP-адресом устройства и функция завершает свою работу. Если подключение не удалось, то микроконтроллер переходит к сбросу настроек.

Блок-схема алгоритма подключения к сети показан на рисунке 21.

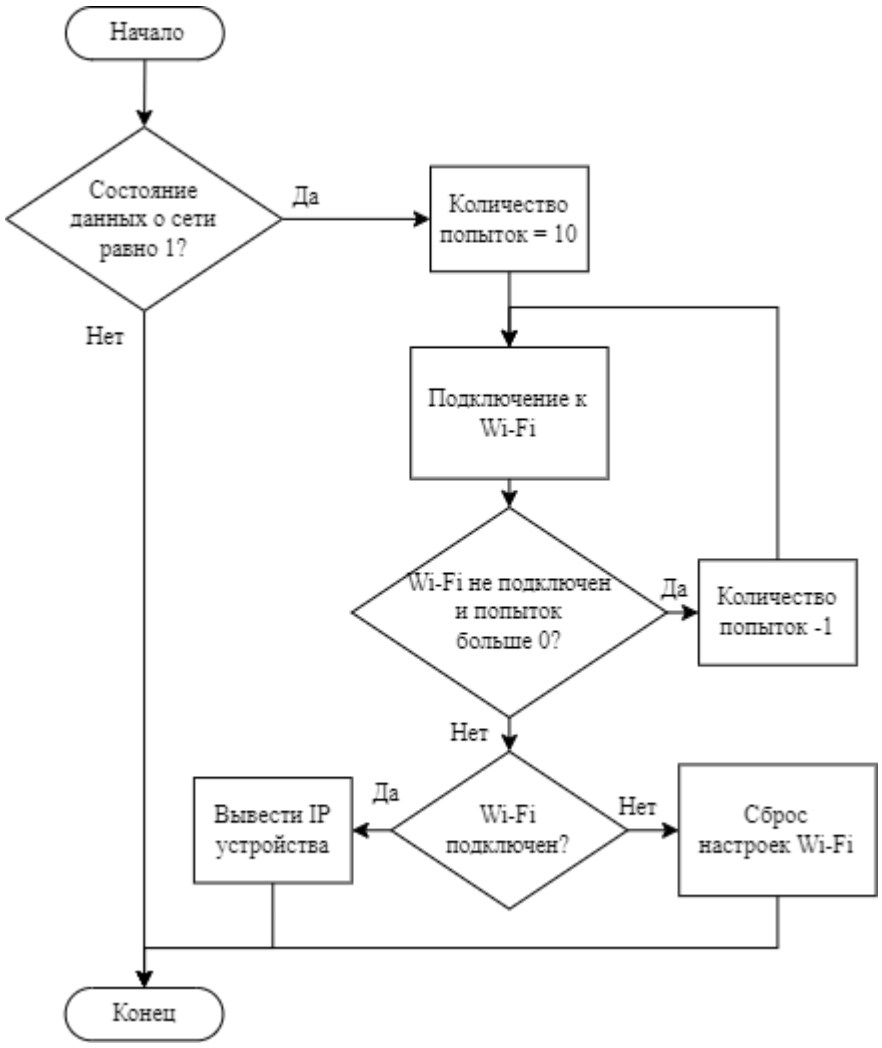

Рисунок 21 – Блок-схема алгоритма подключения к сети Wi-Fi

Код функции подключения к Wi-Fi представлена в листинге 2.

Листинг 2 – Функция подключения к локальной сети

```
void connect_to_wifi() {
  char* SSID = ch_ssid;
  char* PASSWORD = ch_pass;
  byte attempt = 10;

  WiFi.begin(SSID, PASSWORD);

  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED && attempt > 0) {
    delay(500);
    Serial.print(".");
```

```
    attempt--;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println();
    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println();
    Serial.println("Couldn't connect to wifi...");
    reset();
  }
}
```

После подключения к сети микроконтроллер записывает свой идентификатор. В Arduino IDE уже есть функция, которая это делает – get_chip_id(). Она возвращает число типа uint32_t. Я изменил ее, чтобы привести все функции к типу void, в таком случае функция ничего не возвращает, а просто присваивает глобальной переменной id_esp32 значение идентификатора. Код функции представлен в листинге 3.

Листинг 3 – Функция записи идентификатора ESP32

```
void get_id_esp32() {
  uint32_t chipid = 0;
  for (int i = 0; i < 17; i = i + 8) {
    chipid |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
  }
  id_esp32 = chipid;
}
```

Когда шлюз подключен к сети и идентификатор устройства записан, микроконтроллер устанавливает связь с сервером и отправляет на него запрос. Обмен данными происходит по протоколу HTTP.

HTTP – это протокол прикладного уровня для передачи гипермедиа-документов, таких как HTML, XML, изображений, видео и других ресурсов. HTTP использует модель клиент-сервер. В качестве клиента выступает микроконтроллер ESP32. Он делает запрос на сервер и получает от него ответ.

Сообщения запроса и ответа структурированы определенным образом, включая метод запроса (например, GET, POST или PUT) и заголовки, предоставляющие дополнительную информацию о запросе или ответе.

Для обмена данными с сервером используется библиотека HTTPClient.h. Все функции библиотеки применяются к объекту типа HTTPClient. В проекте используются следующие функции этой библиотеки:

– begin() – эта функция подключается к серверу, в качестве параметра принимает указатель на адрес сервера, записанного в строку;

– addHeader() – эта функция добавляет заголовок к запросу, которые показывает, какой тип данных будет отправляться, например, простой текст;

– POST() – эта функция отправляет запрос на сервер методом POST и возвращает код ответа;

– getString() – эта функция возвращает строку, в которую записан ответ с сервера.

Прежде чем перейти к обмену данными, микроконтроллер проверят подключение к сети. Если оно не установлено, то обмена данными не происходит, и функция завершает свою работу. Если же подключение к Wi-Fi установлено, ESP32 подключается к серверу, для этого в коде указан URL-адрес вместе с названием файла, в котором расположен скрипт обработки запроса.

Отправка запроса производится методом POST. В отличие метода GET он более защищенный, потому что данные не передаются через URL, они не сохраняются в кэше, их можно увидеть только с помощью инструментов разработчика. Помимо этого, с помощью метода POST можно отправлять фалы, например, JSON объект. Для этого в заголовке запроса прописывается соответствующий тип контента – application/json [12].

JSON – это облегченный формат обмена данными, который легко читать и писать, что делает его популярным выбором для веб-API. Запрос JSON обычно содержит данные в виде пар ключ-значение, которые могут быть легко

проанализированы и интерпретированы сервером. По сравнению с другими форматами данных, такими как XML или CSV, JSON весит меньше, что ускоряет его передачу по сети. Он также поддерживается широким спектром языков программирования и инструментов, что упрощает работу с ним.

Для работы с объектами JSON понадобится библиотека ArduinoJson.h. Библиотека поддерживает создание статических и динамических документов. Разница между ними заключается в размере и использовании памяти. Разработчики библиотеки рекомендуют использовать статический документ для файлов размером меньше 1 Кбайт, в таком случае объект будет располагаться в стэке. В остальных случаях рекомендуется использовать статический документ, он будет располагаться в куче [12].

Создание и работа с документами происходит с помощью следующего синтаксиса:

– StaticJsonDocument<48> sjd_request – создание статического документа с названием sjd_request с выделением 48 Кбайт памяти;

– sjd_request["id_esp32"] = id_esp32 – создание в документе пары ключ-значение, где в квадратных скобках указывается ключ, которому присваивается значение;

– serializeJson(sjd_request, json) – преобразование документа sjd_request в строку json с сжатием, то есть без пробелов и разрывов между значениями, именно эта строка будет отправлена методом POST на сервер;

– deserializeJson(sjd_response, ch_response) – эта функция анализирует ввод JSON, то есть файл sjd_response и помещает результат в ch_resonse. В случае возникновения ошибки функция возвращает 0;

– sleep_time = sjd_response["sleep_time"] – расшифровка данных, в переменную sleep_time записывается значение, которое соответствует ключу sleep_time в документе sjd_response.

Создание статического JSON документа сопровождается выделением памяти, которое задает пользователь. Размер памяти определяется исходя из

количества символов. На сайте разработчика библиотеки есть инструмент для определения рекомендуемого размера памяти. Так для запроса, содержащего одну пару ключ-значение с идентификатором устройства, рекомендуется выделить 48 Кбайт памяти. Для ответа, содержащего время сна и целевое значение освещенности рекомендуется использовать 96 Кбайт [15].

После отправки запроса сервер присылает код состояния HTTP. Он показывает, был ли успешно выполнен определённый HTTP запрос. Например, код 200 говорит о том, что запрос был успешно обработан и результат действий сервера передан в теле ответа. Если соединение с сервером не удалось, код ответа в программе остается равным нулю, обработка ответа пропускается, функция завершает свою работу [15].

Результат работы сервера записывается в переменную payload типа String. В данном случае ответ тоже приходит в формате JSON. Далее с помощью функции toCharArray() строка переводится в массив символов. Это нужно для анализа ответа с помощью библиотеки ArduinoJson.h. После чего происходит парсинг ответа и запись полученных значений в соответствующие переменные.

Блок-схема алгоритма взаимодействия с сервером показана на рисунке 22.

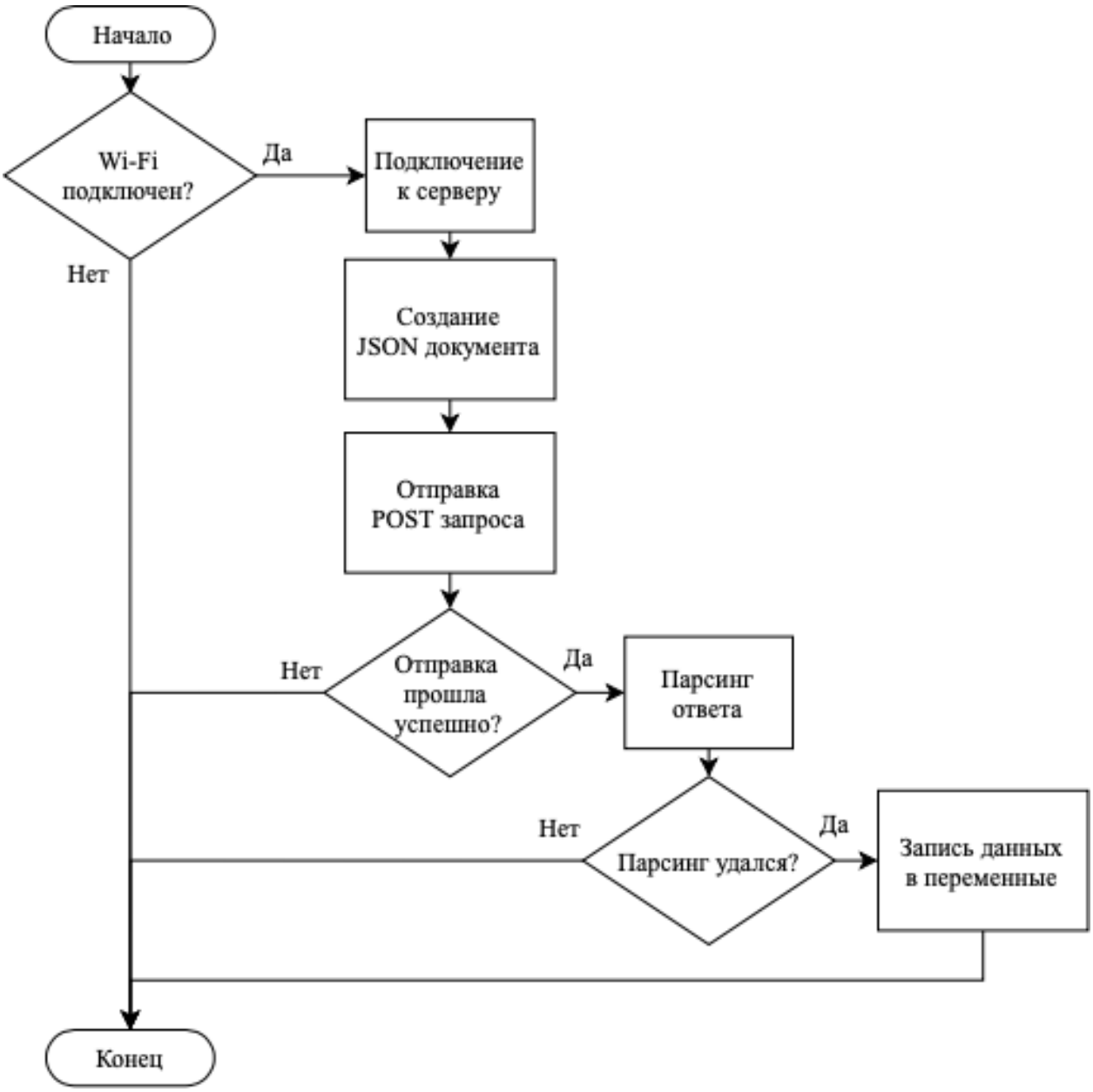

Рисунок 22 – Блок-схема алгоритма отправки запроса на сервер

Функции взаимодействия ESP32 с сервером представлен в листинге 4.

Листинг 4 – Функция взаимодействия ESP32 с сервером

```
void send_json_request() {
  String serverName = "http://192.168.50.163:81/json.php";

  if ((WiFi.status() == WL_CONNECTED)) {
```

```
        HTTPClient http;

        http.begin(serverName.c_str());
        http.addHeader("Content-Type", "application/json");

        StaticJsonDocument<48> sjd_request;
        sjd_request["id_esp32"] = id_esp32;

        String json;
        serializeJson(sjd_request, json);

        int httpResponseCode = http.POST(json);

        if (httpResponseCode > 0) {
          Serial.print("HTTP Response code: ");
          Serial.println(httpResponseCode);

          String payload = http.getString();

          char ch_response[60];
          payload.toCharArray(ch_response, 60);

          StaticJsonDocument<96> sjd_response;
          DeserializationError              error              =
deserializeJson(sjd_response, ch_response);

          if (error)
            return;

          sleep_time = sjd_response["sleep_time"];
          target_light = sjd_response["target_light"];

        } else {
          Serial.print("Error code: ");
          Serial.println(httpResponseCode);
        }
        http.end();
      } else {
        Serial.println("WiFi Disconnected");
      }
    }
```

Некоторые функции шлюза могут выполняться по заранее заданным сценариям, которые привязаны к реальному времени. Например, в рабочее время уровень освещенности должен быть равен 100%, в нерабочее – 50%, а в ночное – освещение должно быть отключено. Шлюз связывается с NTP-сервером, то есть с сервером реального времени, чтобы по данным от него выбрать правильный сценарий. Для этого в программе микроконтроллера есть

функция get_local_time(). Эта функция использует библиотеку time.h, которая позволяет связываться с сервером реального времени и переводить данные в удобный для работы формат.

В составе библиотеки уже есть пример программы, которая выводит реальное время с сервера pool.ntp.org. Это одни из самых популярных сервисов с большим кластером серверов по всему миру.

Основная функция, которая используется для связи с сервером - configTime(gmtOffset_sec, daylightOffset_sec, ntpServer), она принимает в себя следующие параметры:

– gmtOffset_sec – сдвиг по времени относительно UTC в секундах. Для Самарской области часовой пояс UTC+4, поэтому в данную переменную записывается количество секунд в одном часе, то есть 3600, умноженное на 4;

– daylightOffset_sec – сдвиг по времени в летнее время. Актуален только для тех мест, где летом переводят время на 1 час;

– ntpServer – URL NTP-сервера, в данном случае pool.ntp.org.

Далее вызывается функция getLocalTime(). Она отправляет пакет запроса на NTP-сервер, анализирует полученный ответ и сохраняет информацию о дате и времени в структуре времени, называемой timeinfo. Для этого используется специальный тип данных struct tm.

Для записи времени в переменную используется функция strftime(timeMinute, 3, "%M", &timeinfo). Она принимает в себя следующие параметры:

– массив символов char, timeMinute, в котором будет храниться один член временной структуры, например, минуты;

– размер массива;

– спецификатор данных временной структуры, например %M, которая обозначает минуты;

– ссылка на временную структуру, которая содержит информацию о времени.

В таблице 4 показаны другие спецификаторы временной структуры.

Таблица 4 – Спецификаторы временной структуры

Спецификатор	Расшифровка
%A	День недели
%B	Месяц
%d	День
%Y	Год
%H	Час
%M	Минута
%S	Секунда

Код функции get_locat_time() представлен в листинге 5.

Листинг 5 – Функция связи с сервером реального времени

```
void get_local_time() {

  const char* ntpServer = "pool.ntp.org";
  const long gmtOffset_sec = 4 * 3600;
  const int daylightOffset_sec = 0;

  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    return;
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");

  char timeHour[3];
  strftime(timeHour, 3, "%H", &timeinfo);

  char timeMinute[3];
  strftime(timeMinute, 3, "%M", &timeinfo);
}
```

ESP32 связывает 2 протокола беспроводной связи: Wi-Fi и Zigbee. Шлюз с помощью Wi-Fi подключается к серверу и обменивается с ним данными. Часть из этих данных предназначены для отправки в модуль Zigbee.

Микроконтроллер ESP32 связывается с Zigbee по интерфейсу UART. Для отправки данных по этому интерфейсу в программе предусмотрена функция send_data_to_uart().

Исходя из схемы электрической принципиальной, связь между модулями происходит по UART2. Прежде чем работать с ним, его необходимо инициализировать. Для этого используется библиотека HardwareSerial.h. Перед инициализацией нужно создать объект типа HardwareSerial. Конструктор принимает одно значение – номер порта UART. Инициализация порта происходит аналогично инициализация стандартного порта, то есть функцией begin(). Эта функция принимает лишь одни параметр – скорость передачи данных в битах в секунду или в бодах. Стандартная скорость равно 115200 бод, именно на ней и будет передаваться информация модулю Zigbee.

Далее внутри функции формируется строка с данными в формате «целевая освещенности=значение». Значение целевой освещенности приходит в ответе от сервера. После того, как строка сформирована, она отправляется по UART с помощью функции Serial2.println(e18_data). Функция работает аналогично стандартной функции взаимодействия микроконтроллера с последовательным портом.

Код отправки данных в модуль Zigbee представлен в листинге 6.

Листинг 6 – Функция отправки данных в модуль Zigbee

```
void send_data_to_uart() {

  String e18_data = "target_light=" + (String)target_light;
  Serial2.println(e18_data);
}
```

Отправка данных по интерфейсу UART – последнее действие в алгоритме, которое выполняет микроконтроллер. Повторять весь алгоритм циклично не имеет смысла, данные на сервере меняются нечасто. Достаточно выполнять программу с периодичностью в несколько секунд или минут. Пока

микроконтроллер не выполняет никаких функций, он продолжает потреблять энергию. Чтобы снизить потребление энергии, в программе предусмотрена функция, переводящая его в режим сна go_to_deep_sleep().

ESP32 имеет 5 режим работы, которые отличаются активными функциями и, как следствие, потреблением электричества.

В активном режиме все модули ESP32 работают, и потребление тока составляет до 260 мА. Оптимальный режим для работы микроконтроллера – режим глубокого сна.

В режиме глубокого сна основной ЦП отключается, тогда как сопроцессор со сверхнизким энергопотреблением (ULP Coprocessor) может снимать показания датчиков и активировать ЦП по мере необходимости. Вместе с ЦП отключается и основная память чипа. Активным остается также модуль RTC со всей периферией (рисунок 23). Потребление тока в таком режиме работы составляет до 10 мкА, что существенно меньше, чем в активном режиме.

Рисунок 23 – Активные модули ESP32 в режиме глубокого сна

Поскольку основная память неактивна, все, что хранится в этой памяти, стирается и становится недоступным. В таком случае данные о сети и другая важная информация будет уничтожена, когда микроконтроллер будет

переходить в режим глубокого сна. Но, поскольку память RTC остается активной, ее содержимое сохраняется даже во время глубокого сна и может быть восстановлено после пробуждения чипа. Чип будет сохранять данные о соединениях Wi-Fi в памяти RTC перед переходом в режим глубокого сна.

Чтобы ESP32 сохранил данные в RTC память, при объявление глобальных переменных необходимо указать атрибут RTC_DATA_ATTR. Однако, тип данных String не может быть сохранен в память RTC, именно поэтому данные о сети сохраняются в виде массива char символов.

Помимо данных о сети в RTC память также сохраняется состояние данных о сети – переменная wifi_data_state – и время сна. Хоть время сна и приходит в ответе от сервера, его лучше сохранять на время сна на случай, если шлюз не сможет установить соединение с сервером. Если время сна не будет сохраняться и микроконтроллер не сможет подключиться к серверу, то оно будет равно нулю и микроконтроллер не будет засыпать.

Режим глубокого сна накладывает и другие ограничения. Переход в этот режим завершает все функции, в том числе и инициализацию модулей, и стирает из памяти значения всех переменных. Это приводит к тому, что весь код должен быть написан в блоке setup(), блок loop() выполнен не будет.

Есть несколько причин пробуждения ESP32, в данном проекте оно происходит по таймеру и по нажатию на кнопку RESET. Эти причины должны быть указаны перед переходом в режим сна. Для этого используются следующие функции:

- esp_sleep_enable_ext0_wakeup(GPIO_NUM_13, HIGH) – включается возможность пробуждения по внешнему источнику, при котором на вывод GPIO13 подается высокий уровень сигнала, в данном случае при нажатии на кнопку;
- esp_sleep_enable_timer_wakeup(sleep_time * uS_TO_S_FACTOR) – включает возможность пробуждения по таймеру. Причем время сна указывается в микросекундах, поэтому в функцию передается значение в

секундах sleep_time, умноженное на коэффициент uS_TO_S_FACTOR, который равен 1000000.

Для перехода в режим сна используется стандартная функция ESP32 – esp_deep_sleep_start(). Код функция настройки режима глубокого сна и перехода в него показан в листинге 7.

Листинг 7 – Функция перехода в режим глубокого сна

```
void go_to_deep_sleep() {
  Serial.println("Going to sleep...");

  esp_sleep_enable_ext0_wakeup(GPIO_NUM_13, HIGH);
  esp_sleep_enable_timer_wakeup(sleep_time * uS_TO_S_FACTOR);
  esp_deep_sleep_start();
}
```

Последняя функция, которая используется при работе с ESP32 – сброс настроек reset(). Программа выполняет эту функцию, если не смогла подключиться к сети Wi-Fi или при нажатии на кнопку RESET. Функция сброса настроек стирает данные о сети Wi-Fi, переводит состояние данных о сети в 0, а также сбрасывает время сна (листинг 8). Тип данных SSID и пароль – массив символов, поэтому для его сброса используется функция memset().

Листинг 8 – Функция сброса настроек

```
void reset() {

  wifi_data_state = 0;
  memset(ch_ssid, 0, sizeof(ch_ssid));
  memset(ch_pass, 0, sizeof(ch_pass));
  sleep_time = 0;
  Serial.println("Wifi settings reset, please connect your bluetooth device");
}
```

Весь цикл программы с подключением к сети и обменом данными с сервером занимает около 2-3 секунд. Все остальное время ESP32 находится в

спящем режиме. Длинное удержание кнопки должно принудительно выводить микроконтроллер из глубокого сна и запускать функцию сброса. Именно поэтому кнопка подключена к выводам, которые связаны с RTC. В самом начале программы проверяется причина выхода из сна с помощью функции esp_sleep_wakeup_cause(). Если она равна ESP_SLEEP_WAKEUP_EXT0, то чип переход к функции сброса.

Блок-схема алгоритма сброса настроек по нажатию на кнопку показана на рисунке 24.

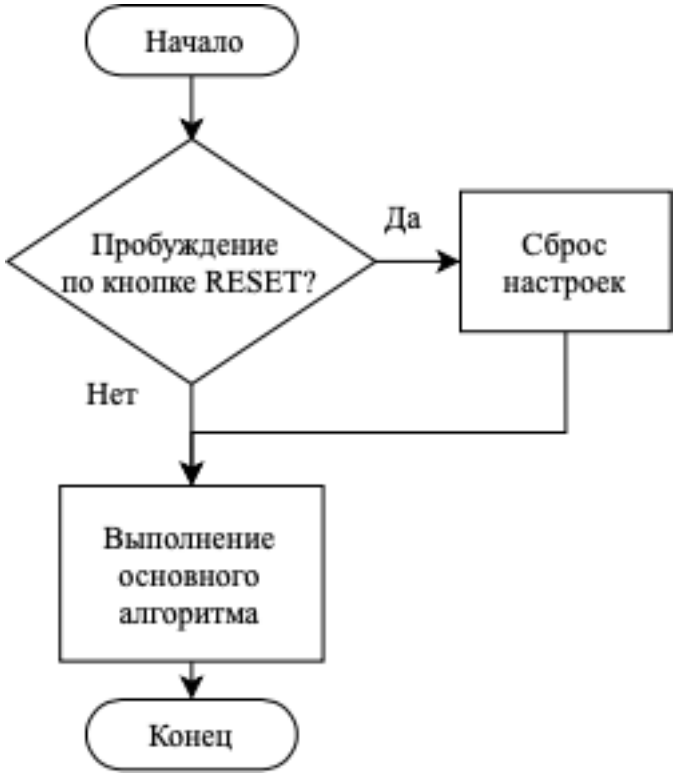

Рисунок 24 – Блок-схема алгоритма сброса настроек по нажатию на кнопку

Код проверки причины пробуждения представлен в листинге 9.

Листинг 9 – Выход из сна и сброс настроек по нажатию на кнопку

```
        if (esp_sleep_get_wakeup_cause() == ESP_SLEEP_WAKEUP_EXT0)
{
          Serial.println("Waken up by button");
          reset();
        }
```

Помимо основных функций в программе также инициализируется последовательный порт с помощью функция Serial.begin(115200). Основная его задача – проверка правильности работы всех функций с помощью отправки сообщений в последовательный монитор функцией Serial.print(). Данные из него доступны только разработчику.

Полный листинг программы ESP32 представлен в приложении А.

Таким образом, встроенное ПО учитывает особенности микроконтроллера, а именно поддержку сразу двух беспроводных технологий и возможность перехода в режим сна для снижения потребляемого тока. Встроенное ПО написано с использованием подключаемых библиотек и функций.

6 Разработка служебного ПО

Служебное ПО – это программное обеспечение, которое используется для поддержки, управления или обеспечения работоспособности других программ или компьютерных систем. Это могут быть программы для управления базами данных, программы для мониторинга и управления сетями, операционные системы, драйверы устройств, антивирусное ПО и другие подобные приложения.

В качестве служебного ПО в данном проекте выступает мобильное приложение и сервер, на котором располагается html-страничка, база данных и скрипты для обработки запросов.

Предназначение мобильного приложения заключается в том, чтобы подключиться к шлюзу по Bluetooth и передать в него данные о сети Wi-Fi. Для написания мобильного приложения использовался конструктор мобильных приложений MIT App Inventor.

MIT App Inventor – это веб-приложение с открытым исходным кодом, которое позволяет пользователям создавать мобильные приложения для устройств Android и iOS. Платформа использует графический интерфейс с готовыми функциональными компонентами, а программирование происходит с помощью программных блоков. Это позволяет пользователям создавать проекты без необходимости обширных знаний в области кодирования, поскольку имеет высокую степень абстракции.

Для создания пользовательского интерфейса используются следующие компоненты:

- надпись – визуальный компонент, который упрощает навигацию;

- выбор из списка – раскрывает список устройств для подключения по Bluetooth;

- текст – поле для ввода текста, которое используется для заполнения SSID и пароля от сети;

- клиент Bluetooth – невидимый компонент, который позволяет приложению использовать Blurtooth;

кнопка – запускает скрипт по отправке данных о сети в устройство, с которым установлена связь по Bluetooth.

Внешний вид мобильного приложения в окне разработки показан на рисунке 25.

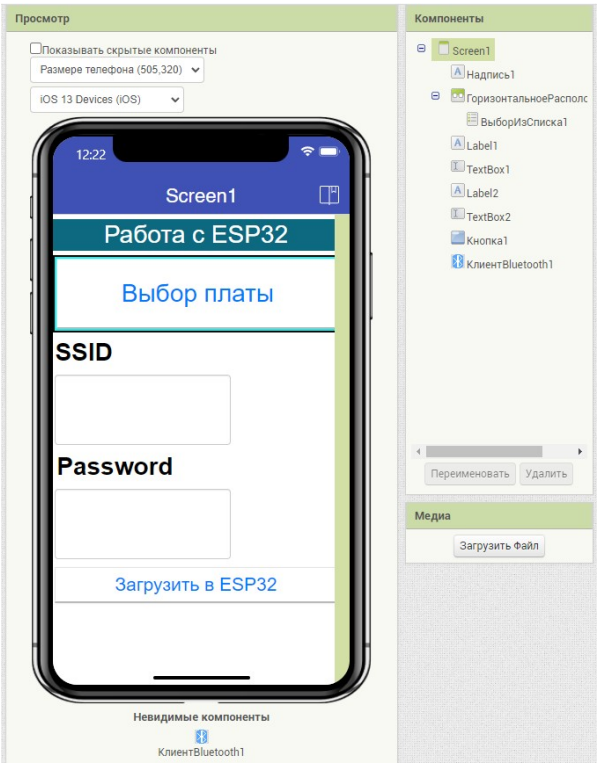

Рисунок 25 – Пользовательский интерфейс мобильного приложения в окне разработчика

Алгоритм работы приложения начинается с инициализации двух переменных: SSID и пароль. Изначально они пустые. Затем пользователь нажимает на кнопку «Выбор платы», после чего раскрывается список со всеми доступными устройствами Bluetooth. Ранее в ПО шлюза было прописано название устройства – «ESP32». Именно его и нужно будет выбрать. При успешном подключении приложение свернет список, а в случае неудачи на экране появится сообщение об ошибке. Далее пользователь заполняет поля

SSID и пароль. Чтобы отправить их микроконтроллеру, пользователь нажимает кнопку «Загрузить в ESP32», после чего запускается скрипт. Во время работы скрипта данные из полей записываются в соответствующие переменные, из них формируется строка, которая затем отправляется в микроконтроллер.

Программа из блоков в окне разработчика показана на рисунке 26.

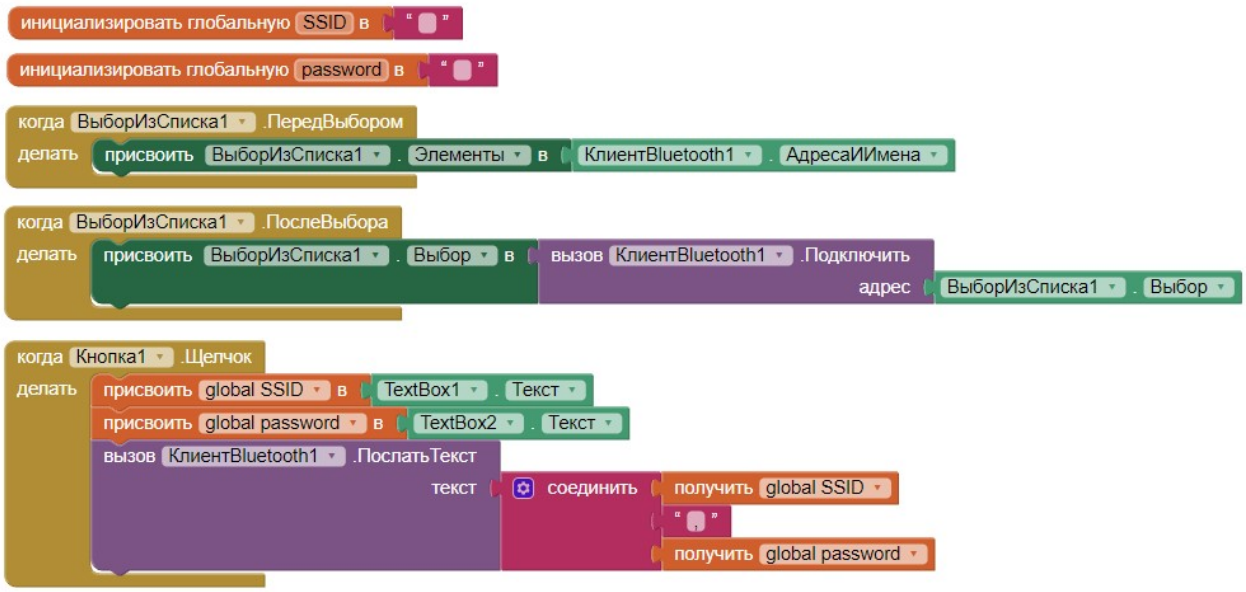

Рисунок 26 – Программа мобильного приложения

В качестве сервера используется утилита Open Server Panel. Он позволяет быстро разворачивать сервер и дает возможность создавать несколько доменов. Также он поддерживает некоторые веб-приложения, например, phpMyAdmin, который позволяет работать с базами данных. Open Server Panel поддерживает несколько СУБД.

В качестве СУБД для данного проекта была использована mySQL. MySQL – это система управления реляционными базами данных с открытым исходным кодом, которая использует язык структурированных запросов (SQL) для управления данными. Он широко используется для веб-приложений

и других программ, которые требуют хранения, поиска и управления данными в реляционной базе данных.

Благодаря phpMyAdmin не обязательно знать язык SQL, поскольку это приложение предлагает удобный графический интерфейс для выполнения различных операций с базами данных, таких как создание, удаление, изменение и экспорт баз данных, таблиц и полей, а также выполнение запросов SQL [15].

На сервере будет располагаться всего одна база данных, для простоты она будет называться esp32. В базе данных будет одна таблица под названием mytable. В ней будет предусмотрено несколько полей, их список показан в таблице 5.

Таблица 5 – Поля таблицы mySQL

Поле	Описание
number	Порядковый номер записи
id_esp32	Идентификатор ESP32
date_time	Дата и время
sleep_time	Время сна ESP32
target_light	Целевая освещенность

Каждое поле имеет свой тип, а некоторые поля еще и дополнительные свойства. Так поле number является первичным ключом, то есть оно будет иметь уникальное значение. Помимо этого, в этом поле стоит настройка автоматического увеличения. Это означает, что, когда в таблицу будет добавляться новая запись, его значение будет автоматически увеличено на 1, будет присвоен следующий порядковый номер [15].

Поле date_time имеет тип, datetime. В него записывается автоматически генерируемое текущее время, когда создается новая запись или изменяется имеющаяся. На рисунке 27 показана структура таблицы в приложении phpMyAdmin.

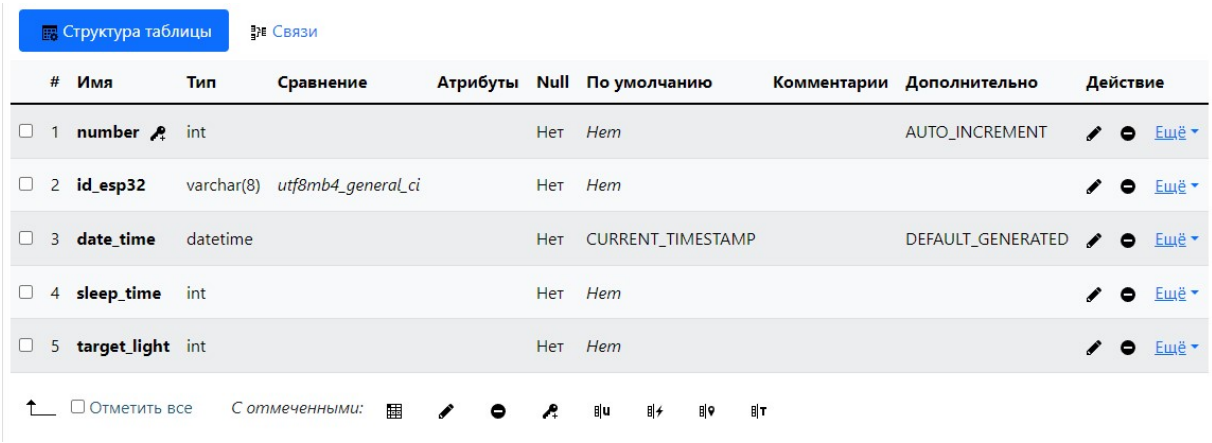

Рисунок 27 – Структура таблицы mytable

Работа сервера возможна благодаря трем файлам php. Их список и назначение указаны в таблице 6.

Таблица 6 – Список файлов на сервере

Файл	Назначение
index.php	Главная страница html, графический интерфейс
json.php	Скрипт обработки JSON запроса от ESP32
form.php	Скрипт обработки формы

Взаимодействие шлюза с таблицей происходит через скрипт json.php, который написан на языке PHP. Это серверный скриптовый язык, используемый для веб-разработки. Он бесплатный для использования, имеет открытый исходный код и широко используемый для создания динамических веб-страниц и веб-приложений. PHP предназначен для работы с HTML и может быть встроен в код HTML для обеспечения таких функций, как доступ к базе данных, обработка форм и другие динамические функции. Он обычно используется в сочетании с MySQL [15].

PHP может автоматически обрабатывать GET и POST запросы, но файл JSON с его помощью автоматически обработать нельзя. Для этого его нужно

поймать с помощью функции file_get_contents(‘php://input’). Эта функция возвращает объект, который попал в скрипт. Перед работой с данными запроса его нужно спарсить, для этого используется функция json_decode(), которая преобразует файл json в массив, из которого потом можно легко достать данные. В JSON-запросе приходит идентификатор ESP32. После парсинга его значение присваивается переменной $id_esp32 [15].

Для работы с базой данных с ней нужно создать подключение, то есть объект $sql, с помощью функции new mysqli(). Она принимает в себя информацию о базе данных: хостинг, логин, пароль, база данных. К этому объекту применяются запросы SQL. Для обновления данных используется запрос UPDATE. В данном скрипте обновляется только время.

В ответ на запрос скрипт формирует свой файл JSON, в который записываются время сна и целевая освещенность. Для этого скрипт выполняет поиск строки с соответствующим идентификатором и выбирает из него нужные данные. Выбор данных происходит с помощью запроса SELECT. Эти данные записываются в массив, который потом преобразуется в документ JSON с помощью функции json_encode(). Когда ответ сформирован, он отправляется командой echo.

Код скрипта json.php представлен в листинге 10.

Листинг 10 – Скрипт обработки запроса JSON

```
<?php
$jsonData = file_get_contents('php://input');
$decodedArray = json_decode($jsonData, true);

$id_esp32 = $decodedArray['id_esp32'];

$conn = new mysqli("openserver", "root", "", "esp32");

if ($conn->connect_error) {
   die("Error: couldn't connect to database: " . $conn-
>connect_error);
}
```

```
    $sql = "UPDATE mytable SET date_time = NOW() WHERE
id_esp32=$id_esp32";

    if ($conn->query($sql) === TRUE) {

      $sql = "SELECT sleep_time, target_light FROM mytable WHERE
id_esp32=$id_esp32";
      $result = $conn->query($sql);

      $response = array();
      if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
          $response['sleep_time'] = $row['sleep_time'];
          $response['target_light'] = $row['target_light'];
        }
      }

      $myJSON = json_encode($response);

      header('Content-Type: application/json');
      echo $myJSON;
    } else {
      echo "Error: " . $sql . "<br>" . $conn->error;
    }

    $conn->close();
    ?>
```

Файл index.php является главой страницей, именно с ней взаимодействует пользователь. На ней видно текущие настройки и время последнего запроса. Также на ней есть форма, через которую можно изменять настройки времени сна и освещенности. Страница поддерживает работу как на компьютере, так и на мобильных телефонах. На рисунке 28 показан внешний вид главной страницы.

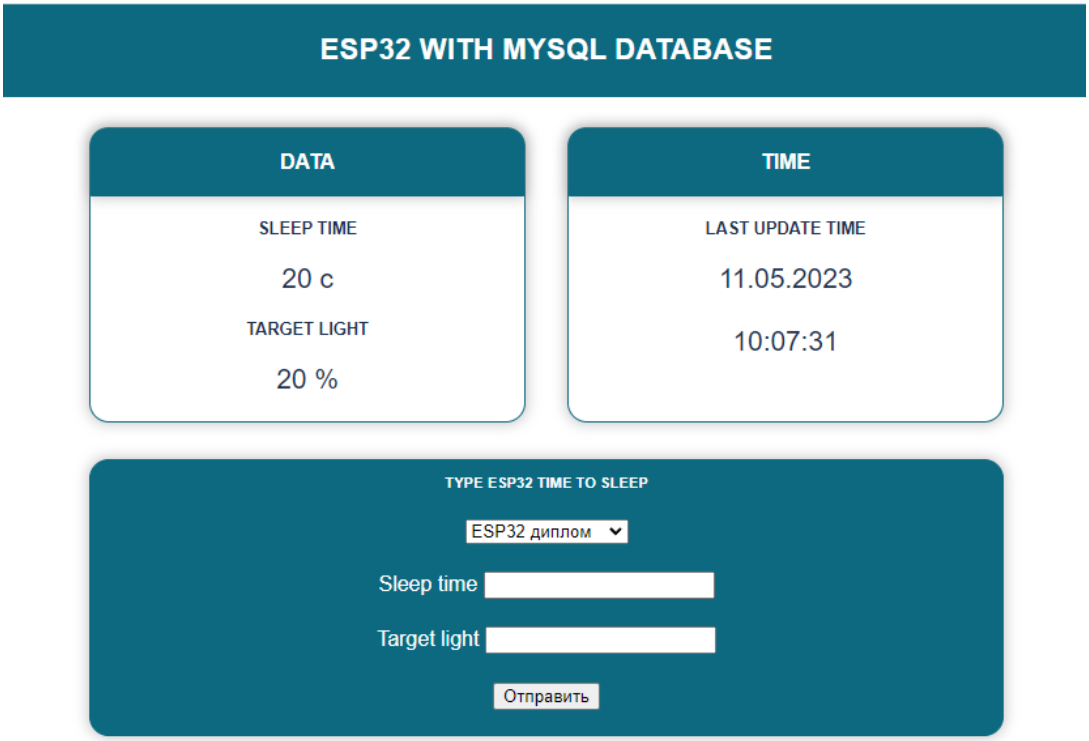

Рисунок 28 – Интерфейс главной страницы

Страница написана на языке HTML с использование стилей CSS. Поскольку дизайн не является главной задачей на разработку, был использован готовый стилевой шаблон. Для этого в теге link указана ссылка на стиль. В тэге style заданы цвета, шрифты, формы и положение элементов для различных классов. Структура страницы блочная, то есть написана с использованием тэгов div. В каждом блоке указывается класс элемента с помощью атрибута class.

Отображение данных на странице в блоках DATA и TIME происходит с помощью php, то есть скрипты php встроены в код HTML. Скрипты производят поиск значений с помощью SQL запроса SELECT в нужных полях по идентификатору, который был получен заранее и равен 13539700 (листинг 11).

Листинг 11 – SQL запрос на получение времени сна

```
SELECT sleep_time FROM mytable WHERE id_esp32 = 13539700
```

Форма имеет 3 поля: ID ESP32 – поле выбора идентификатора устройства, Sleep time и Target light – числовые поля, в которое пользователь записывает желаемое время сна и освещенность. Идентификатор устройства соответствует «ESP32 Диплом» в списке выбора устройства. Данные из формы отправляются в скрипт form.php методом POST по нажатию на кнопку. Ниже представлен HTML код формы (листинг 12).

Листинг 12 – Форма на главной странице

```
<form action="form.php" align=center method="post">
    <p><select name="id_esp32">
                    <option  selected  value="13539700">ESP32
диплом</option>
                <option value="12345678">ESP32 тестовая</option>
            </select>
    </p>
    <p><label style="font-size: 1rem;">Sleep time </label>
        <input type="number" name="sleep_time" />
    </p>
    <p><label style="font-size: 1rem;">Target light </label>
        <input type="number" name="target_light" />
    </p>
    <p><input type="submit" /></p>
</form>
```

Полный код главной страницы представлен в приложении Б.

Скрипт form.php работает аналогично скрипту обработки JSON запроса, но в отличии от него не отправляет ответ. В языке PHP данные POST запроса могут быть расшифрованы автоматически с помощью функции $_POST[], где в скобках указывается название ключа. Функция возвращает значение, присвоенное ключу, то есть значение из формы.

Данные из формы присваиваются внутренним переменным скрипта, затем происходит подключение к таблице. С помощью SQL запроса UPDATE

новые значения из формы заносятся в таблицу, после чего скрипт завершает свою работу (листинг 13).

Листинг 13 – Скрипт обработки данных из формы

```
<?php

$id_esp32 = $_POST['id_esp32'];
$sleep_time = $_POST['sleep_time'];
$target_light = $_POST['target_light'];

$conn = new mysqli("openserver", "root", "", "esp32");
if ($conn->connect_error) {
     die("Error: couldn't connect to database: " . $conn->connect_error);
}
echo 'Successfully connected to database.<br>';

$sql = "UPDATE mytable SET sleep_time = $sleep_time, target_light = $target_light, date_time = NOW() WHERE id_esp32 = $id_esp32";

$result = mysqli_query($conn, $sql);
?>
<meta http-equiv="Refresh" content="0; URL=\index.php">
```

7 Разработка конструкции устройства

Все элементы устройства расположим на одной печатной плате. Все компоненты будут крепиться непосредственно к плате, поэтому разместим их на одной из поверхностей.

Печатный узел будет выполнен из двухстороннего стеклотекстолита фольгированного СФ-2-105Г-1,5, параметры которого приведены в таблице 7.

Таблица 7 – Параметры стеклотекстолита фольгированного СФ-2-105Г-1,5

Марка	Электрические параметры фольгированного стеклотекстолита			
	Толщина изоляции d, мм	Толщина фольги h, мкм	Максимальная плотность тока J, А/мм^2	Удельное сопротивление изолятора ρ, Ом/м
СФ-2-105Г-1,5	1,5	105	4	1,2·109

Ширина дорожки будет рассчитываться по формуле 13.

$$l = \frac{I}{J \cdot h} \tag{13}$$

где I – потребляемый ток, А;

J – максимальная плотность тока, А/м^2;

h – толщина фольги, м.

Максимально возможное потребление тока устройством равно 293 мА, причем ESP32 может потребляет 260 мА. Ток всех остальных дорожек, включая информационные не будет превышать 20 мА. В таблице 8 показана минимальная ширина дорожек для разных цепей, рассчитанная по приведенной выше формуле.

Таблица 8 – Минимальная ширина дорожек для разный цепей

Цепь	Ток, мА	Ширина дорожки, мм
Питание	0,293	0,698
Питание ESP32	0,26	0,619
Питание Zigbee	0,028	0,067
Светодиод питания	0,005	0,012
Информационные дорожки	0,02	0,048

При разводке примем преобладающую ширину дорожек равную 0,1 мм, для цепи питания ESP32 примем ширину дорожки равную 1 мм.

Поскольку рабочие напряжения схемы не поднимаются выше 3,3 В, примем минимальное расстояние между печатными дорожками до 0,2мм.

Монтаж элементов, в основном, проводится поверхностным способом, однако, электролитические конденсаторы, светодиод, разъемы и кнопки монтируются в отверстия. В монтажных и переходных отверстиях необходимо предусмотреть металлизацию. Диаметр контактных площадок для переходных отверстий выбираем равным 1мм.

Разводку печатной платы будем осуществлять в программе Altium Designer. Печатную плату будем делать квадратной формы размером 70х70мм. В соответствии с изложенными правилами настроим пакет трассировки в Altium Designer и выполним разводку печатной платы. Результаты трассировки платы на стеклотекстолите фольгированном двустороннем показаны на рисунках 29-30 [6].

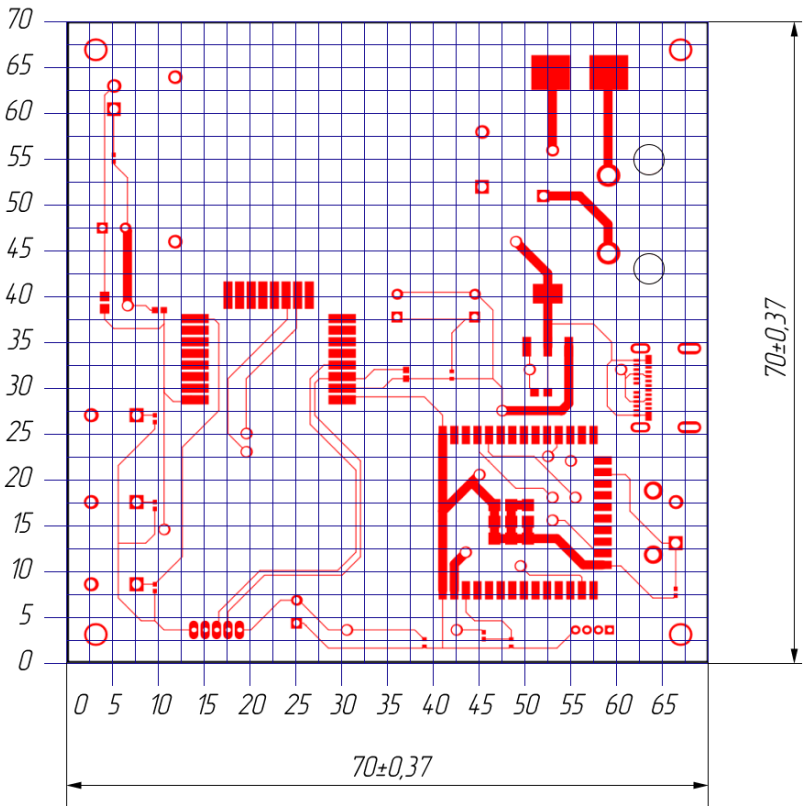

Рисунок 29 – Вид печатной платы сверху

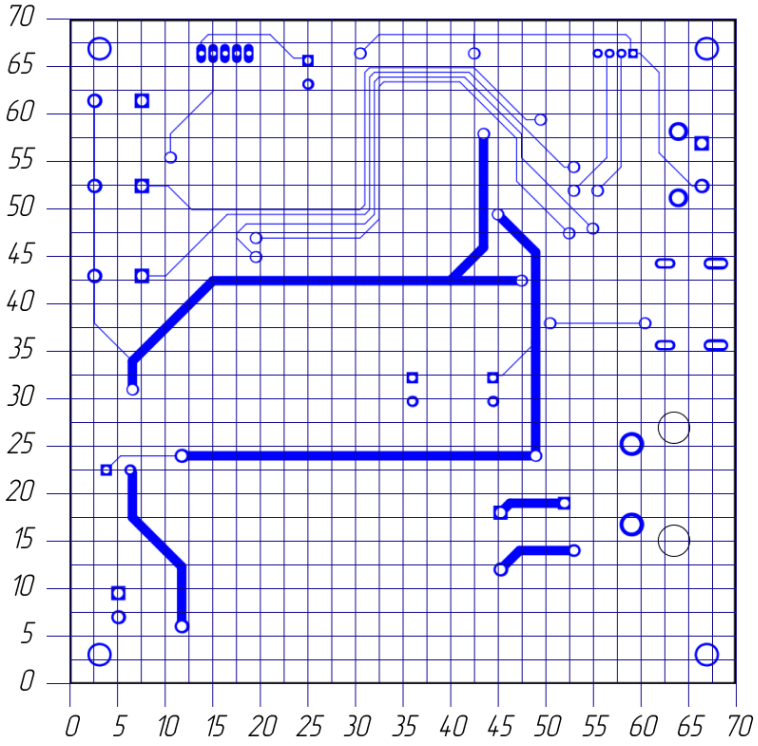

Рисунок 30 – Вид печатной платы снизу

В результате разводки на плате получилось 65 отверстий: 4 для монтажа платы в корпус, 2 для крепления разъема питания, 18 переходных и 41 для монтажа элементов.

Примем, что изготовление печатной платы будет проводиться химическим способом, путем вытравливания дорожек с помощью химически активного вещества – хлорного железа. Металлизированные отверстия, как правило, получают путем осаждения меди электрохимическим способом. Для формирования рисунка проводящих дорожек, которые необходимо оставить на плате, наносят защитный слой лака, или другого материала, стойкого к хлорному железу. При этом под лаком медь остается, а в других местах растворяется в активном веществе – хлорном железе.

После этого в стеклотекстолите высверливают необходимые монтажные и переходные отверстия и оставшиеся дорожки обслуживаются припоем. Для защиты от воздействия окружающей среды, печатную плату покрывают защитными составами необходимо цвета, наносят с помощью шелкографии необходимые графические и символьные изображения.

Логика размещения элементов следующая: вокруг предохранителя есть достаточно пространства, чтобы его можно было легко заменить, разъемы питания и кнопка RESET размещаются на одной стороне, чтобы быть незаметными пользователю, светодиод питания – на противоположной, напротив, чтобы его было видно, разъемы для программирования размещаются на краю платы, чтобы к ним был удобный доступ [6].

Расположение элементов показано на рисунке 31.

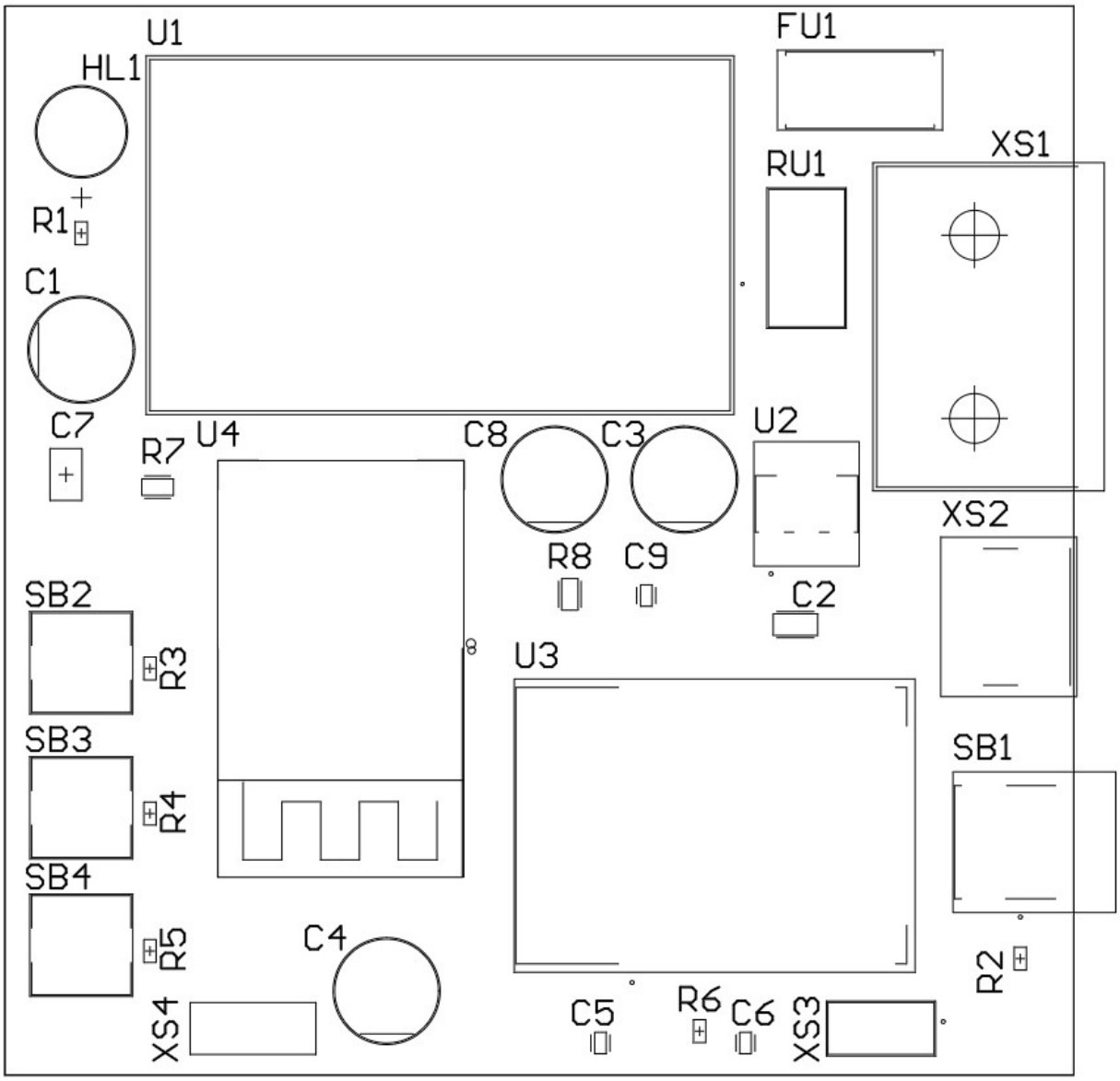

Рисунок 31 – Эскиз сборочного чертежа

Таким образом, готовая плата будет иметь небольшой размер и возможность установки в корпус. Программирование модулей и обслуживание устройства не будет вызывать сложностей за счет удобного расположения элементов.

8 Экономическая часть

Расчет стоимости материалов и комплектующих, необходимых для изготовления одного экземпляра шлюза будем проводить с учетом цен на электронные компоненты и материалы в Российской Федерации [7].

Будем учитывать, что ряд материалов, таких как хлорное железо, стеклотекстолит имеются в продаже только в минимальных партиях поставки, стоимость которых фиксирована и поэтому может появиться избыток этих материалов.

Результаты расчета затрат на компоненты и материалы свели в таблицу 9.

Таблица 9 – Затраты на основные материалы для изготовления шлюза

Наименование материала	Марка	ГОСТ, ТУ	Ед. имз.	Норма расхода, шт	Цена за единицу (руб.)	Затраты (руб.)
Флюс	Канифольный раствор на спирту	19113-84	шт.	1	70	70
Припой оловянно свинцовый	ПОС60	48-0220-57-93	шт.	1	55	55
Стеклотекстолит фольгированный двухсторонний	СФ-2-105Г-1,5	10316-78	шт.	1	100	100
Хлорное железо	$FeCl_3$	6-00-05763458-129- 91	шт.	1	247	247
Итого						472

Полную стоимость покупных комплектующих изделий определили по таблице 9.

Таблица 10 – Стоимость покупных комплектующих изделий

Наименование изделия	Марка, размер	Количество, шт.	Цена за единицу (руб.)	Затраты (руб.)
Микросхемы	ESP32-WROOM-32D	1	160	160
	E18-MS1-PCB	1	280	280
	HLK-5M03	1	210	210
	LM3940	1	120	120
Кнопки	1825027-8	1	17	17
	FSM4JRT	3	42	126
Разъемы	RAPC322X	1	160	160
	105450-0101	1	50	50
	53047-0410	1	8	8
	53047-0510	1	35	35
Варистор	MOV-07D471KTR	1	130	130
Предохранитель	0154.500DRT	1	470	470
Светодиод	HLMP-3301	1	46	46
Резитсторы	RC0402FR-07330RL - 330 Ом	1	0,2	0,2
	RC0402FR-071KL - 1 кОм	2	0,07	0,14
	RC0402FR-0710KL - 10 кОм	5	0,2	1
Конденсаторы	CC0603JRNPO9BN101 - 100 пФ	1	0,6	0,6
	EEUFR1E101B - 100 мкФ	4	19	76
	GRM188R71C474KA88D - 0,47 мкФ	1	0,7	0,7
	GRM155R71C104KA88D - 0,1 мкФ	3	13	39
Итого				1929,64

С учетом транспортно-заготовительных расходов стоимость покупных изделий рассчитывается по формуле 14.

$$C_{\text{покуп}} = \sum_{i=1}^{n} S_{\text{покуп}} \cdot (1 + K_{\text{тз}}) \qquad (14)$$

где $К_{тз}$ – коэффициент транспортно-заготовительных расходов, Ктз = 0,04;

$S_{покуп}$ – стоимость покупных комплектующих изделий, руб.

Таким образом, стоимость будет равна:

$$С_{покуп} = 1929{,}64 \cdot (1 + 0{,}04) = 2006{,}83 \text{ руб}$$

Результаты расчёта затрат материалы и комплектующие сведены в таблицу 11.

Таблица 11 - Результаты расчета затрат на материалы и комплектующие

Наименование статей затрат	Абсолютная величина затрат, руб.
Затраты на основные материалы	472
Затраты на комплектующие	2006,83
Итого:	2478,83

Таким образом, величина затрат на приобретение по розничным ценам с учетом минимально продаваемых количеств материалов и комплектующих для изготовления одного экземпляра шлюза канала передачи данных составят 2478,83 рублей.

Заключение

В итоге был проведен анализ существующих аналогов, на основе которого были выделены схожие признаки: поддержка различных беспроводных технологий, поддержка питания от напряжения 5 В.

На основе анализа быра разработана информационная инфраструктура, которая включала в себя мобильное приложение, шлюз, сервер и сеть Zigbee.

Разработанная структура устройства включает в себя модули беспроводной связи Bluetooth для взаимодействия с мобильным приложением, Wi-Fi для обмена данными с сервером и Zigbee, для работы с устройствами умного дома. Также устройства включает в себя модули питания и индикации наличия питания, сервисные кнопки и кнопку сброса настроек.

Предложено словесное описание алгоритма работы и его оформление в виде блок-схемы. Алгоритм работы устройства разделен на части, каждая из которых соответствует работе шлюза, мобильного приложения и сервера.

Разработана схема электрическая принципиальная шлюза канала передачи данных в соответствии со структурной схемой. Приведено описание процесса выбора компонентов: в соответствии с рекомендациями производителей или расчеты по формулам.

На основе алгоритма написано основное ПО для модуля ESP32 и вспомогательное ПО в виде мобильного приложения и веб-страницы и скриптов. Проработаны вопросы взаимодействия микроконтроллера ESP32 с сервером, кнопками и модулем ZigBee.

На базе электрической схемы разработана и разведена печатная плата. Подсчитана стоимость комплектующих.

Список используемых источников

1. Грингард С. Интернет вещей. Будущее уже здесь. - М.: Издательская группа Точка, 2017. - 224 с.
2. Робачевский А. Интернет изнутри. Экосистема глобальной Сети. - М.: Альпина Паблишер, 2015. - 223 с.
3. Хоровиц, П., Хилл У. Искусство схемотехники. - 2-е изд. - М.: БИНОМ, 2014. - 704 с.
4. Северанс Ч. Как работают компьютерные сети и интернет - М.: ДМК Пресс, 2022. – 116 с.
5. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы. - 5-е изд. - СПб.: Питер, 2016. - 992 с.
6. Лопаткин А. В. Проектирование печатных плат в системе Altium Designer. - 2-е изд. - М.: ДМК, 2017. - 554 с.
7. ЧИП и ДИП URL: https://www.chipdip.ru/ (дата обращения: 12.05.2023).
8. Кэмерон Н. Электронные проекты на основе ESP8266 и ESP32: Создание приложений и устройств с поддержкой Wi-Fi. - М.: ДМК Пресс, 2022. - 456 с.
9. Bluetooth URL: https://www.bluetooth.com/ (дата обращения: 11.04.2023).
10. CSA URL: https://csa-iot.org/ (дата обращения: 12.04.2023).
11. EBYTE URL: https://www.cdebyte.com/ (дата обращения: 14.04.2023).
12. ESP32 HTTP GET and HTTP POST with Arduino IDE (JSON, URL Encoded, Text) // Random Nerd Tutorials URL: https://randomnerdtutorials.com/esp32-http-get-post-arduino/ (дата обращения: 6.04.2023).
13. ESPRESSIF URL: https://www.espressif.com/ (дата обращения: 10.04.2023).

14. Home Assistant Yellow URL: https://www.home-assistant.io/yellow/ (дата обращения: 14.04.2023).

15. Прохоренюк Н. А. HTML, JavaScript, PHP и MySQL. Джентельменский набор Web-мастера. - 4-е изд. - СПб.: БВХ-Петербург, 2015. - 768 с.

16. Kolban N. Kolban's book on ESP32. - 2017. - 785 с.

17. Drew G. Zigbee wireless networking. - 2007. - 427 с.

18. Samsung URL: https://www.samsung.com/ru/ (дата обращения: 13.04.2023).

19. SnapEDA URL: https://www.snapeda.com/ (дата обращения: 17.04.2023).

20. Xiaomi URL: https://www.mi.com/ru (дата обращения: 12.04.2023).

Приложение А

Встроенное ПО

Листинг А.1 – Полный код программы для ESP32

```
#include "WiFi.h"
#include "HTTPClient.h"
#include "ArduinoJson.h"
#include "BluetoothSerial.h"
#include "time.h"
#include "HardwareSerial.h"

#define uS_TO_S_FACTOR 1000000
#define button_pin 13

RTC_DATA_ATTR bool wifi_data_state;
RTC_DATA_ATTR char ch_ssid[20];
RTC_DATA_ATTR char ch_pass[20];
RTC_DATA_ATTR uint64_t sleep_time;

uint32_t id_esp32;
int target_light;

BluetoothSerial ESP_BT;

void setup() {

  HardwareSerial Serial2(2);

  Serial.begin(115200);
  Serial2.begin(115200);
  pinMode(button_pin, INPUT);

  if (esp_sleep_get_wakeup_cause() == ESP_SLEEP_WAKEUP_EXT0)
{
    Serial.println("Waken up by button");
    reset();
  }

  ESP_BT.begin("ESP32");
  Serial.println("ESP32 is awake!");

  if (!wifi_data_state) {
    get_wifi_settings();
  }
  connect_to_wifi();
  get_id_esp32();
  send_json_request();
  get_local_time();
  send_data_to_uart();
```

Продолжение листинга А.1

```
 go_to_deep_sleep();
}

void loop() {
  //no code here
}

void get_id_esp32() {
  uint32_t chipid = 0;
  for (int i = 0; i < 17; i = i + 8) {
    chipid |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
  }
  id_esp32 = chipid;
}

void get_wifi_settings() {

  String s_ssid_pass;
  String s_ssid;
  String s_pass;

  while (!wifi_data_state) {

    while (ESP_BT.available()) {
      s_ssid_pass = ESP_BT.readString();

      for (int i = 0; i < s_ssid_pass.length(); i++) {
        if (s_ssid_pass.substring(i, i + 1) == ",") {
          s_ssid = s_ssid_pass.substring(0, i);
          s_pass = s_ssid_pass.substring(i + 1);

          delay(1000);
          wifi_data_state = true;

          s_ssid.toCharArray(ch_ssid, 20);
          s_pass.toCharArray(ch_pass, 20);

          Serial.println("Recieved wifi settings");
          break;
        }
      }
    }
  }
}

void connect_to_wifi() {
  if (wifi_data_state) {
```

Продолжение листинга А.1

```
        char* SSID = ch_ssid;
        char* PASSWORD = ch_pass;
        byte attempt = 10;

        Serial.print("SSID = ");
        Serial.println(SSID);
        Serial.print("PASSWORD = ");
        Serial.println(PASSWORD);

        WiFi.begin(SSID, PASSWORD);

        Serial.print("Connecting");
        while (WiFi.status() != WL_CONNECTED && attempt > 0) {
          delay(500);
          Serial.print(".");
          attempt--;
        }

        if (WiFi.status() == WL_CONNECTED) {
          Serial.println();
          Serial.print("Connected, IP address: ");
          Serial.println(WiFi.localIP());
        } else {
          Serial.println();
          Serial.println("Couldn't connect to wifi...");
          reset();
        }
      }
    }

    void send_json_request() {
      String serverName = "http://192.168.50.163:81/json.php";

      if ((WiFi.status() == WL_CONNECTED)) {  //Check the current
connection status

        HTTPClient http;

        http.begin(serverName.c_str());
        http.addHeader("Content-Type", "application/json");

        StaticJsonDocument<48> sjd_request;
        sjd_request["id_esp32"] = id_esp32;
        String json;
        serializeJson(sjd_request, json);
        int httpResponseCode = http.POST(json);
```

Продолжение листинга А.1

```
      if (httpResponseCode > 0) {
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);

        String payload = http.getString();

        char ch_response[60];
        payload.toCharArray(ch_response, 60);

        StaticJsonDocument<96> sjd_response;
                    DeserializationError      error      =
deserializeJson(sjd_response, ch_response);
        if (error)
          return;

        sleep_time = sjd_response["sleep_time"];
        target_light = sjd_response["target_light"];

      } else {
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
      }
      http.end();
    } else {
      Serial.println("WiFi Disconnected");
    }
  }

  void get_local_time() {

    const char* ntpServer = "pool.ntp.org";
    const long gmtOffset_sec = 4 * 3600;
    const int daylightOffset_sec = 0;

    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

    struct tm timeinfo;
    if (!getLocalTime(&timeinfo)) {
      Serial.println("Failed to obtain time");
      return;
    }
    Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");

    char timeHour[3];
    strftime(timeHour, 3, "%H", &timeinfo);

    char timeMinute[3];
    strftime(timeMinute, 3, "%M", &timeinfo);
```

Окончание листинга А.1

```
}

void go_to_deep_sleep() {

  Serial.println("Going to sleep...");
  esp_sleep_enable_ext0_wakeup(GPIO_NUM_13, HIGH);
          esp_sleep_enable_timer_wakeup(sleep_time         *
uS_TO_S_FACTOR);
  esp_deep_sleep_start();
}

void reset() {

  wifi_data_state = 0;
  memset(ch_ssid, 0, sizeof(ch_ssid));
  memset(ch_pass, 0, sizeof(ch_pass));
  sleep_time = 0;
   Serial.println("Wifi settings reset, please connect your
bluetooth device");
}

void send_data_to_uart() {

  String e18_data = "target_light=" + (String)target_light;
  Serial2.println(e18_data);
}
```

Приложение Б

Главная страница HTML

Листинг Б.1- Код главной страницы

```
<!DOCTYPE html>
<html>

<head>
    <title>ESP32 WITH MYSQL DATABASE</title>
       <meta   name="viewport"   content="width=device-width,
initial-scale=1">
                                <link            rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
                                                integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr
" crossorigin="anonymous">
    <link rel="icon" href="data:,">
    <style>
        html {
            font-family: Arial;
            display: inline-block;
            text-align: center;
        }

        p {
            font-size: 1.2rem;
        }

        h4 {
            font-size: 0.8rem;
        }

        body {
            margin: 0;
        }

        .topnav {
            overflow: hidden;
            background-color: #0c6980;
            color: white;
            font-size: 1.2rem;
        }

        .content {
            padding: 5px;
        }

        .cards {
            max-width: 700px;
            margin: 0 auto;
```

Продолжение листинга Б.1

```
                display: grid;
                grid-gap: 2rem;
                        grid-template-columns:  repeat(auto-fit,
minmax(300px, 1fr));
            }

            .card {
                background-color: white;
                box-shadow: 0px 0px 10px 1px rgba(140, 140, 140,
.5);
                border: 1px solid #0c6980;
                border-radius: 15px;
            }

            .card.header {
                background-color: #0c6980;
                color: white;
                border-bottom-right-radius: 0px;
                border-bottom-left-radius: 0px;
                border-top-right-radius: 12px;
                border-top-left-radius: 12px;
            }

            .reading {
                font-size: 1.3rem;
            }

            .packet {
                color: #bebebe;
            }

            .textColor {
                color: #183153;
            }
        </style>
    </head>

    <body>
        <div class="topnav">
            <h3>ESP32 WITH MYSQL DATABASE</h3>
        </div>

        <br>

        <div class="content">
            <div class="cards">
                <div class="card">
                    <div class="card header">
```

Продолжение листинга Б.1

```
                        <h3 style="font-size: 1rem;">DATA</h3>
                    </div>

                    <h4 class="textColor"></i>SLEEP TIME</h4>
                    <p class="textColor"><span class="reading">
                            <?php
                            $servername = "openserver";
                            $username = "root";
                            $password = "";
                            $dbname = "esp32";
                                $conn = new mysqli($servername,
$username, $password, $dbname);

                            if ($conn->connect_error) {
                              die("Connection failed: " . $conn-
>connect_error);
                            }

                           $sql = "SELECT sleep_time FROM mytable
WHERE id_esp32 = 13539700";
                            $result = $conn->query($sql);

                            if ($result->num_rows > 0) {
                                // Output data of first row
                                $row = $result->fetch_assoc();
                               $sleep_time = $row["sleep_time"];
                                echo $sleep_time;
                            } else {
                                echo "NN";
                            }

                            $conn->close();
                            ?> c
                        </span></p>

                    <h4 class="textColor"></i>TARGET LIGHT</h4>
                    <p class="textColor"><span class="reading">
                            <?php
                            $servername = "openserver";
                            $username = "root";
                            $password = "";
                            $dbname = "esp32";
                                $conn = new mysqli($servername,
$username, $password, $dbname);

                            if ($conn->connect_error) {
                              die("Connection failed: " . $conn-
>connect_error);
```

```
                            }

                               $sql = "SELECT target_light FROM
mytable WHERE id_esp32 = 13539700";
                            $result = $conn->query($sql);

                            if ($result->num_rows > 0) {
                                // Output data of first row
                                $row = $result->fetch_assoc();
                                                $target_light  =
$row["target_light"];
                                echo $target_light;
                            } else {
                                echo "NN";
                            }

                            $conn->close();
                            ?>
                            &percnt;
                        </span></p>
                </div>

                <div class="card">

                    <div class="card header">
                        <h3 style="font-size: 1rem;">TIME</h3>
                    </div>

                    <h4 class="textColor">LAST UPDATE TIME</h4>
                    <p class="textColor"><span class="reading">
                            <?php
                            $servername = "openserver";
                            $username = "root";
                            $password = "";
                            $dbname = "esp32";
                                $conn = new mysqli($servername,
$username, $password, $dbname);

                            if ($conn->connect_error) {
                              die("Connection failed: " . $conn-
>connect_error);
                            }

                              $sql = "SELECT MAX(date_time) FROM
mytable WHERE id_esp32 = 13539700";
                            $result = $conn->query($sql);

                            $row = mysqli_fetch_row($result);
                            $last_update_timestamp = $row[0];

                               $last_update_date = date("d.m.Y",
strtotime($last_update_timestamp));
```

Продолжение листинга Б.1

```
                                $last_update_time = date("H:i:s",
strtotime($last_update_timestamp));

                                echo $last_update_date;
                                echo "<br/>";
                                echo "<br/>";
                                echo $last_update_time;

                                $conn->close();
                                ?>
                            </span></p>

                    </div>

                </div>
            </div>

            <br>

            <div class="content">
                <div class="cards">
                     <div class="card header" style="border-radius:
15px;">
                        <h3 style="font-size: 0.7rem;">TYPE ESP32
TIME TO SLEEP</h3>

                            <form  action="form.php"  align=center
method="post">
                            <p><select name="id_esp32">
                                                <option  selected
value="13539700">ESP32 диплом</option>
                                        <option value="12345678">ESP32
тестовая</option>
                                <select>
                            </p>

                            <p><label style="font-size: 1rem;">Sleep
time </label>
                              <input type="number" name="sleep_time"
/>
                            </p>
                           <p><label style="font-size: 1rem;">Target
light </label>
                                                <input  type="number"
name="target_light" />
                            </p>
                            <p><input type="submit" /></p>
                        </form>
```

Окончание листинга Б.1

```
            </div>
        </div>
    </div>

    <br>
    <br>
    <br>

</body>

</html>
```