

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**УСТРОЙСТВО УПРАВЛЕНИЯ
ЭЛЕКТРОПРИБОРАМИ НА ОСНОВЕ
МИКРОКОНТРОЛЛЕРА ESP32**

Студент

 11.06.24 Халилов Алмаз Рамилевич
(подпись, дата)

Руководитель

 11.06.24 Францев Сергей Михайлович
(подпись, дата)

Нормоконтролёр

 11.06.24 Бычков Андрей Станиславович
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от 27.05.2024г. № 11)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

17.06.2024г., № 26)

Секретарь ГЭК


(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 19 » 04 2024 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Халилов Алмаз Рамилевич гр.20ВВП2 факультета вычислительной техники
2. Тема работы: Устройство управления электроприборами на основе микроконтроллера ESP32
Тема утверждена приказом ПГУ № 405/о от 19 апреля 2024 г.
3. Руководитель бакалаврской работы к.т.н. Францев С. М.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать устройство управления электроприборами на основе микроконтроллера ESP32, предоставляющее пользователю возможность включения и отключения подключаемого электроприбора и настройки условий для автоматического выключения и включения. Устройство должно позволять пользователю отслеживать данные окружающей среды, такие как температура, относительная влажность воздуха, выстраивать логику работы микроконтроллера. Также пользователю должна быть предоставлена возможность настраивать текущее время и задавать условия при которых устройство будет само включать или отключать подключенный электроприбор, возможность экстренного включения и выключения электроприбора, настройки сети для подключения. Взаимодействие должно происходить через web – приложение в браузере.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки. Также необходимо изучить технологии для разработки устройства, такие как Arduino IDE, VS code и т.д.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2024– 30.04.2024	
2.	Анализ требований	10	01.05.2024– 02.05.2024	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2024– 07.05.2024	
4.	Описание алгоритмов	40	08.05.2024– 11.05.2024	
5.	Разработка устройства управления электроприборами на основе микроконтроллера ESP32	150	12.05.2024– 29.05.2024	
6.	Оформление пояснительной записки	24	30.05.2024– 31.05.2024	

Дата выдачи задания 19 апреля 2024 г.

Руководитель бакалаврской работы  19.04.2024 Францев С.М.
(подпись, дата)

Задание к исполнению принял студент  19.04.2024 Халилов А.Р.
(подпись, дата)

Работу к защите допустить
Декан факультета  19.06.2024 Фионова Л.Р.

Реферат

Бакалаврская работа содержит 97 листов, 30 рисунков, 5 источников и 6 приложений

ESP32, ARDUINO IDE, VS CODE, WEB-ПРИЛОЖЕНИЕ, УПРАВЛЕНИЕ УСТРОЙСТВОМ

Цель работы – разработать устройство управления электроприборами на основе микроконтроллера ESP32.

Разработанное устройство должно позволять пользователю отслеживать данные окружающей среды, такие как температура и относительная влажность воздуха, выстраивать логику работы микроконтроллера. Также пользователю должна быть предоставлена возможность настраивать текущее время и задавать условия, при которых устройство будет само включать или отключать подключенный электроприбор, возможность экстренного включения и выключения электроприбора, настройки сети для подключения. Взаимодействие должно происходить через web – приложение в браузере.

Результатом проделанной работы является проект, который удовлетворяет поставленной задаче. А именно, разработано, реализовано и протестировано устройство управления подключенным электроприбором. При разработке было использовано большое количество технологий, к которым относятся: язык программирования C++, среда разработки Arduino IDE, среда разработки VS Code, язык гипертекстовой разметки HTML, язык стилей CSS, язык программирования JavaScript, сетевой протокол HTTP, а также различные браузеры для тестирования.

					ПГУ 1.090301.38. 001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата	Лит.	Лист	Листов
Разраб.		Халилов А. Р.	<i>А.Р. Халилов</i>	14.06.24			
Провер.		Францев С. М.	<i>С.М. Францев</i>	14.06.24		4	
У. Контр.		Бычков А. С.	<i>А.С. Бычков</i>	14.06.24	ФВТ, каф. ВТ		
Утвержд.		Митрохин М.А.	<i>М.А. Митрохин</i>	14.06.24	зр. 20ВВП2		
					Устройство управления электроприборами на основе микроконтроллера ESP32		
					Пояснительная записка		

Содержание

Перечень сокращений и обозначений.....	6
Введение.....	7
1 Устройство управления электроприборами на основе микроконтроллера ESP32	8
1.1 Анализ предметной области.....	8
1.2.1 Hiper IoT	10
1.2.2 GSM–розетка Теплэко	13
1.3 Постановка задачи.....	17
1.4 Выбор инструментов разработки.....	18
1.4.1 Arduino IDE для программирования ESP32	18
1.4.2 Visual Studio Code (VS Code) для написания веб–страниц.....	20
1.4.3 Сравнение Arduino IDE с PlatformIO	21
1.4.4 Сравнение VS Code с Sublime Text	23
1.5 Архитектура проекта	24
1.5.1 Используемые технологии, протоколы и библиотеки	24
1.5.2 Аппаратная архитектура.....	26
1.5.3 Программная архитектура.....	36
1.5.4 Разработка веб–интерфейса	41
1.5.5. Реализация логики управления электроприборами.....	43
1.6 Тестирование и отладка	45
2 Безопасность жизнедеятельности.....	54
2.1 Вред здоровью при работе с ПК	54
2.2 Снижение вреда для здоровья при работе с ПК.....	54
2.3 Техника безопасности при работе с компьютером.....	55

2.4 Требования безопасности при использовании устройства, коммутирующего сетевое напряжение 220В	56
2.5 Требования условий эксплуатации устройства управления электроприборами на основе микроконтроллера ESP32	57
Заключение	58
Список использованной литературы.....	59
Приложение А – UML-диаграммы	60
Приложение Б – Электрические схемы	64
Приложение В – Листинг программы.....	66
Приложение Г – Презентация	97

Перечень сокращений и обозначений

ПК – персональный компьютер.

BME280 – Датчик фирмы Bosch Sensortec предназначенный для измерения атмосферного давления, температуры и влажности.

CSS (Cascading Style Sheets) – каскадные таблицы стилей, язык разметки для визуального оформления веб-сайтов.

ESP32 – система на кристалле с интегрированным контроллером радиосвязи Wi-Fi.

HTML (Hyper Text Markup Language) – стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере.

HTTP (HyperText Transfer Protocol) – сетевой протокол передачи гипертекста прикладного уровня.

IDE (integrated development environment) – комплекс программных средств, используемый программистами для разработки программного обеспечения.

I2C (Inter-Integrated Circuit) — последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов.

IIC (Inter-Integrated Circuit) — последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов.

RTC (Real Time Clock) – модуль часов реального времени.

URL (Uniform Resource Locator) – адрес ресурса в интернете.

WDT (Watchdog Timer) – таймер для контроля над зависанием системы, который периодически сбрасывается контролируемой системой.

Введение

В современном мире, где технологический прогресс неумолимо нарастает, вопрос эффективного управления электроприборами приобретает все большее значение. Оптимизация использования ресурсов, включая энергию, становится необходимостью как в домашнем, так и в промышленном окружении. В этом контексте разработка и внедрение устройств управления электроприборами на базе микроконтроллера ESP32 становится важным шагом в направлении повышения эффективности и удобства использования современных технологий.

Основной целью работы является создание комплексной системы, способной не только осуществлять контроль за электроприборами, но и автоматизировать процессы на основе собираемых данных. Устройство будет оборудовано датчиком температуры и влажности, предназначенными для сбора информации о окружающей среде, а также модулем часов реального времени RTC. С помощью собранных данных и загруженных с веб-страницы инструкций устройство будет управлять реле, контролирующим состояние подключенного электроприбора.

Разработка устройства не только упростит и автоматизирует процессы управления, но и позволит существенно экономить ресурсы, повышая энергоэффективность. Помимо этого, устройство будет обладать возможностью управления через веб-интерфейс, что делает его востребованным решением для домашнего использования, оставляя возможность для модернизации. Внедрение разработанного устройства открывает перспективы для дальнейшего развития сферы автоматизации и управления энергопотреблением.

1 Устройство управления электроприборами на основе микроконтроллера ESP32

1.1 Анализ предметной области

В связи с увеличением количества различных устройств и необходимости эффективного управления ими, автоматизация управления электроприборами становятся неотъемлемой частью повседневной жизни. Разработка систем управления на базе микроконтроллеров, таких как ESP32, представляет собой ключевой аспект современных технологических решений. Этот тренд диктуется как потребностями промышленных предприятий, так и обычных потребителей, желающих упростить и автоматизировать свои повседневные дела. Имея в арсенале современные решения мира микросхем, возможности устройства не должны ограничиваться простым управлением узкого круга устройств и событий, необходимо создать такое устройство, которое можно было бы использовать в различных сценариях.

На сегодняшний день сложно перечислить все устройства, автоматизация работы которых принесла бы огромную пользу. Ими могут быть устройства увлажнения, для поддержания оптимальных условий комфортного нахождения в помещении, что применимо дома, в офисе, в небольшом производственном помещении. Круг возможностей на этом не ограничивается, к примеру, допустимо использования устройства в теплице, для поддержания условий роста культур, в системах автоматических ворот, в системах освещения, или же в хозяйстве, для управления наполнением резервуаров. Можно долго продолжать перечисление существующих проблем, это может значить только одно – проект имеет востребованность.

Если подытожить, основной задачей данной предметной области является создание устройства, способного контролировать состояние и функционирование электроприборов с использованием доступа через беспроводное соединение. Устройство должно быть оснащено не только средствами управления, но и датчиками, предназначенными для сбора данных о состоянии окружающей среды,

таких как температура, влажность и другие параметры. Такой подход позволит создать не просто устройство управления, а целую систему мониторинга и контроля, способную адаптироваться к различным условиям эксплуатации.

Анализ существующих решений в данной области позволяет выявить как положительные, так и отрицательные аспекты существующих технологий и подходов к решению задач управления. Важно учитывать как технические, так и пользовательские требования, чтобы разработать оптимальное решение, удовлетворяющее потребности конечных пользователей. Подобный анализ помогает определиться с выбором конкретных технологий и методов, которые будут наилучшим образом соответствовать целям и задачам проекта.

Реализация системы, подходящей под описание, потребует внимательного проектирования, а также тщательного выбора компонентов и применения технологий в области программного обеспечения и аппаратной разработки. Эффективное управление электроприборами строится не только на высокой функциональности, но и на надежности работы системы в различных условиях эксплуатации. Несомненно, кроме того, что система должна быть удобной в использовании обычными пользователями и обладать большим функциональным потенциалом, не менее важно обеспечить стабильную и бесперебойную работу.

Данное решение позволит создать интеллектуальные устройства, способные оптимизировать энергопотребление, повысить комфорт и безопасность жизни пользователей, а также сделать процесс управления электроприборами более гибким и эффективным.

1.2 Анализ существующих аналогов

Среди аналогов можно выделить:

- Hiper IoT;
- GSM-розетка Теплэко;

1.2.1 Hiper IoT

HIPER IoT представляет собой управляемую розетку. Она предоставляет возможность удаленного контроля за электроприборами в доме через мобильное приложение, позволяя пользователям проверить статус подключенных устройств и, при необходимости, включить или выключить их удаленно.

В данное устройство встроена функция мониторинга электроэнергии. Она позволяет отслеживать расход электроэнергии при использовании устройств и оптимизировать его. На борту так же имеется таймер, с помощью него можно выставить время работы электроприбора. Все вышеперечисленное помогает достичь снижения расходов на коммунальные услуги.

Кроме того, данная розетка совместима с голосовыми помощниками. Она интегрируется в различные системы, что в свою очередь обеспечивает дополнительный уровень удобства при использовании.

Она предназначена исключительно для использования внутри помещений. Устройство обладает мощностью до 3600 Вт (250В/16А) и поддерживает входное напряжение АС 100–250В, 16А; 50/60 Гц.

Устройство и пользовательский графический интерфейс представлены на рисунках 1–4.



Рисунок 1 – Иконка приложения

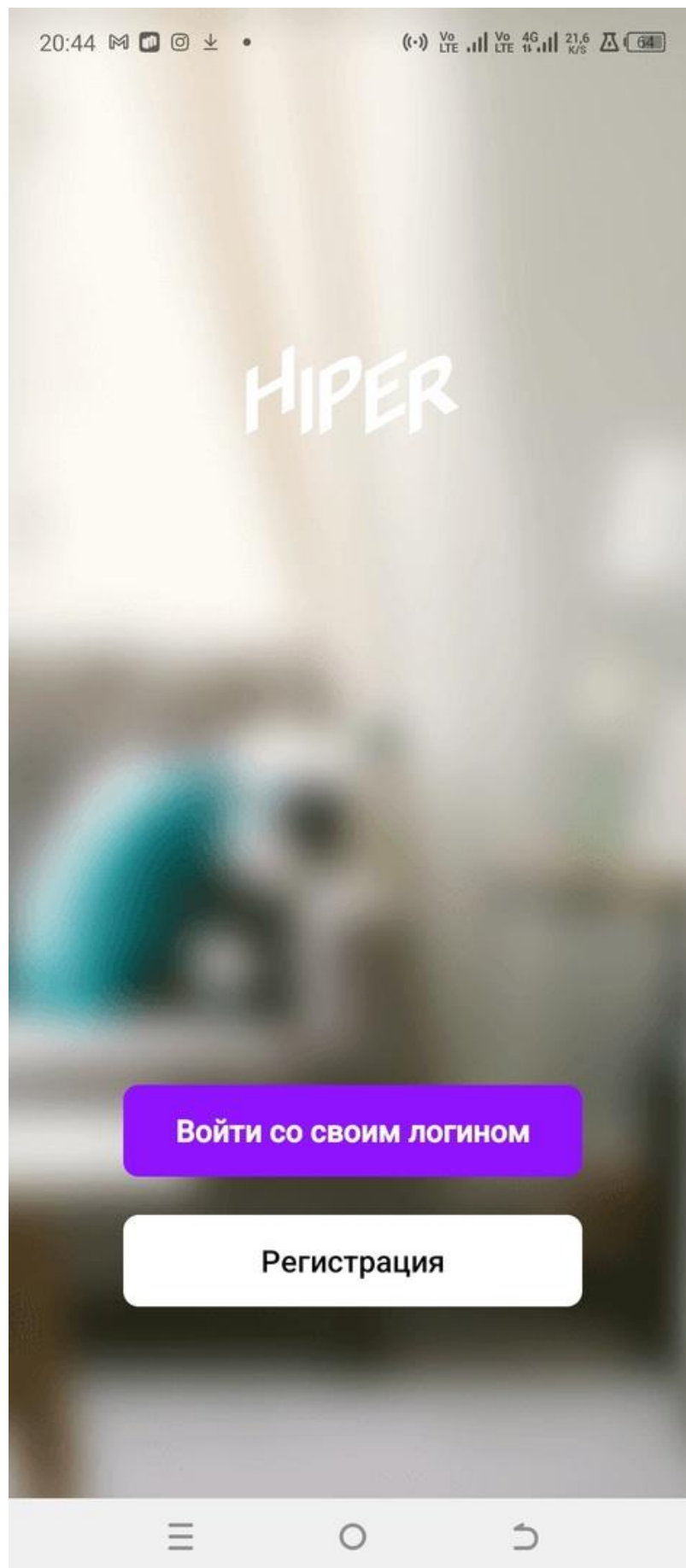


Рисунок 2 – Регистрация в приложении

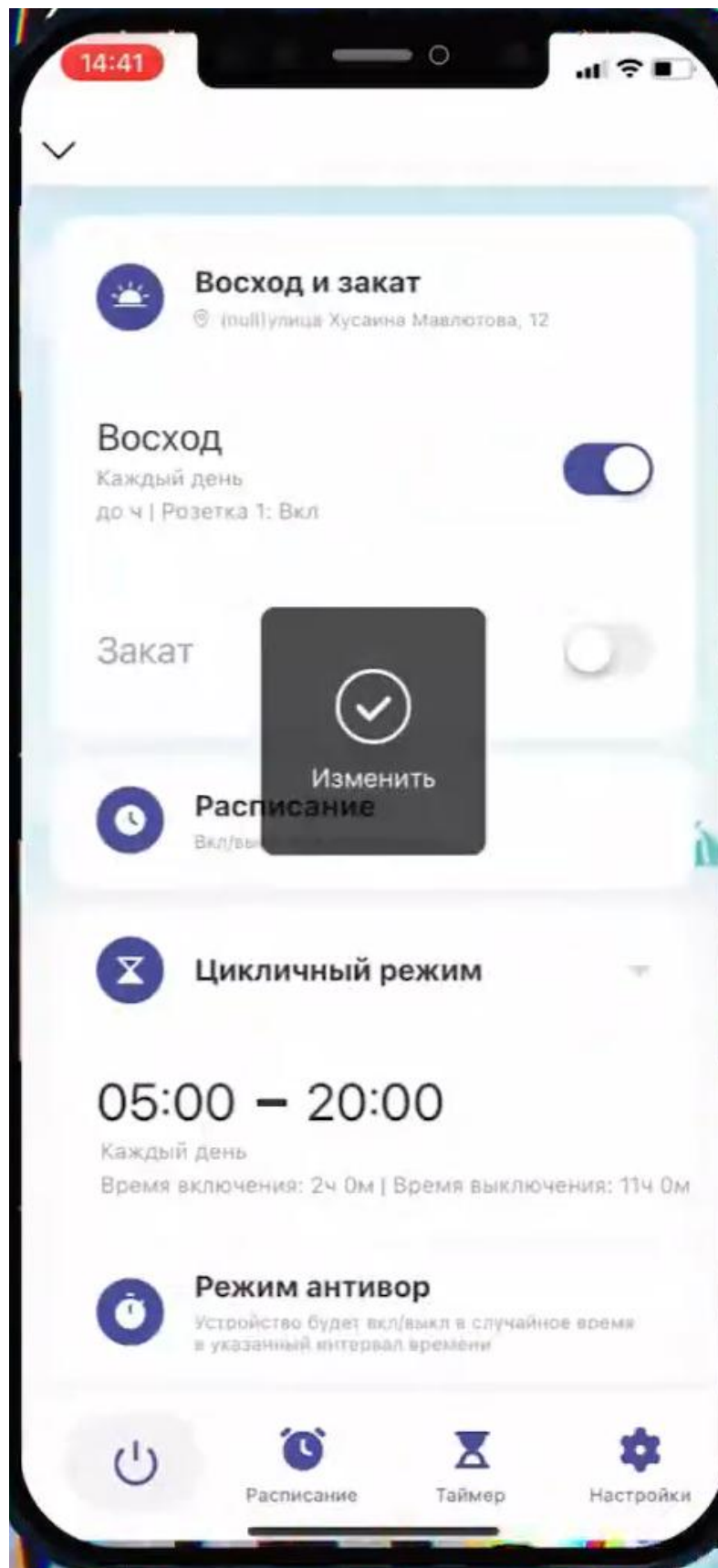


Рисунок 3 – Управление устройством



Рисунок 4 – Вид устройства

Плюсы Hiper IoT:

- небольшая стоимость;
- удобный интерфейс для управления;
- возможность работы с таймером;
- управление через Wi-Fi.

Минусы Hiper IoT:

- невозможность подключения датчиков;
- необходимость скачивания приложения для управления устройством;
- не рассчитан на использование вне помещений.

1.2.2 GSM-розетка Теплэко

GSM розетка представляет собой устройство, управляемое с помощью SMS сообщений. Данная розетка имеет датчик температуры. Это позволяет управлять электроприборами через сеть мобильной связи и одновременно с этим отслеживать температуру в помещении. Пользователь может включать или выключать подключенные устройства с помощью SMS-сообщений, а также получать информацию о текущей температуре с датчиков в реальном времени.

Розетка обладает также функцией мониторинга энергопотребления и возможностью установки таймеров для автоматического включения и выключения устройств в зависимости от температуры. Предназначена для

использования внутри помещений с надежным уровнем подключения к сети мобильной связи.

Устройство представлено на рисунках 5 и 6.



Рисунок 5 – Вид устройства



Рисунок 6 – Вид подключенного устройства

Плюсы GSM–розетка Теплэко:

- возможность работы с таймером;
- возможность работы с датчиком температуры.

Минусы GSM–розетка Теплэко:

- высокая стоимость;
- неудобный интерфейс для управления;
- невозможность подключения иных датчиков;
- отсутствие гибкой логики;
- необходимость покупки sim–карты с платным тарифом;
- присутствие только одного датчика.

Сравнив аналоги, выявили сильные и слабые стороны каждого из них. Поэтому при создании устройства необходимо учесть преимущества аналогов и устранить недостатки. Очевидно, удобство управления играет важную роль для

пользователей. Отличным решением будет создание веб-интерфейса, обеспечивающего доступность и простоту использования, позволяя управлять устройством с помощью ПК или мобильных устройств. Благодаря данному решению можно устранить необходимость в специальном программном обеспечении и повысить универсальность устройства.

Следующее, что делает устройство привлекательным для пользователей, — возможность автоматизации управления, что позволяет упростить его эксплуатацию. Данная функция критически важна для тех задач, которые требуют постоянного мониторинга и оперативного реагирования на изменения. Из этого следует, что интеграция автоматизированных функций существенно поможет повысить эффективность и надежность устройства, обеспечивая более стабильную и предсказуемую работу.

Выбор программируемого микроконтроллера ESP32 в качестве платформы обусловлен его мощными характеристиками и экономичностью. ESP32 обеспечивает высокую производительность при низком энергопотреблении. Поддержка Wi-Fi — одна из особенностей, которая необходима в работе. Таким образом, сочетание веб-интерфейса, автоматизации управления и платформы ESP32 позволит создать устройство, которое превзойдет существующие аналоги по своим характеристикам и возможностям.

1.3 Постановка задачи

При постановке задачи перед проектом стоит учитывать ряд факторов и требований. В рамках данного проекта необходимо реализовать устройство на базе микроконтроллера ESP32, которое будет предназначено для управления электроприборами на основе заданной через веб-интерфейс логики. Важно обеспечить не только функциональность, но и максимальную простоту использования, а также гибкость настройки, чтобы удовлетворить потребности различных категорий пользователей.

Целью данного проекта является создание устройства управления электроприборами на основе микроконтроллера ESP32.

Задачи:

- Разработка и реализация макетного образца устройства на базе микроконтроллера ESP32, обеспечивающего надежное и стабильное функционирование.
- Интеграция в макет различных датчиков и формирование веб-страницы, для обеспечения актуальности информации и эффективного управления;
- Создание гибкой системы настройки логики управления, позволяющей пользователям определять условия и правила работы электроприборов в соответствии с их потребностями и предпочтениями;
- Разработка веб-страницы управления, обладающего приятным для восприятия интерфейсом, чтобы обеспечить удобство использования для конечных пользователей.

Для достижения поставленных целей и задач необходимо провести комплекс работ, включающий в себя анализ требований, проектирование аппаратной и программной части устройства, реализацию функционала, тестирование, а также оптимизацию и доработку на основе полученных результатов.

Важно отметить, что успешная реализация поставленных задач требует не только высокой квалификации и опыта в области разработки, но и внимательного анализа потребностей пользователей и особенностей эксплуатационных условий.

Только таким образом можно обеспечить создание устройства, которое будет надежно и эффективно выполнять свои функции в любых условиях использования.

1.4 Выбор инструментов разработки

Процесс выбора инструментов разработки для данного проекта является ключевым этапом при его создании. В данном случае были тщательно проанализированы доступные инструменты, учтены их функциональность, гибкость и соответствие требованиям проекта. После проведенного анализа были выбраны следующие инструменты, которые порекомендовали себя с лучшей стороны и подходят для разработки данного проекта:

1.4.1 Arduino IDE для программирования ESP32

Arduino IDE – популярная интегрированная среда разработки. Она предназначена для программирования микроконтроллеров, включая ESP32. Данная среда разработки обладает простым и интуитивно понятным интерфейсом, и это делает ее идеальным выбором как для начинающих разработчиков, так и для опытных специалистов. В данном разделе уделим внимание основным особенностям, а также преимуществам использования Arduino IDE. Вид рабочего пространства предоставлен на рисунке 7.

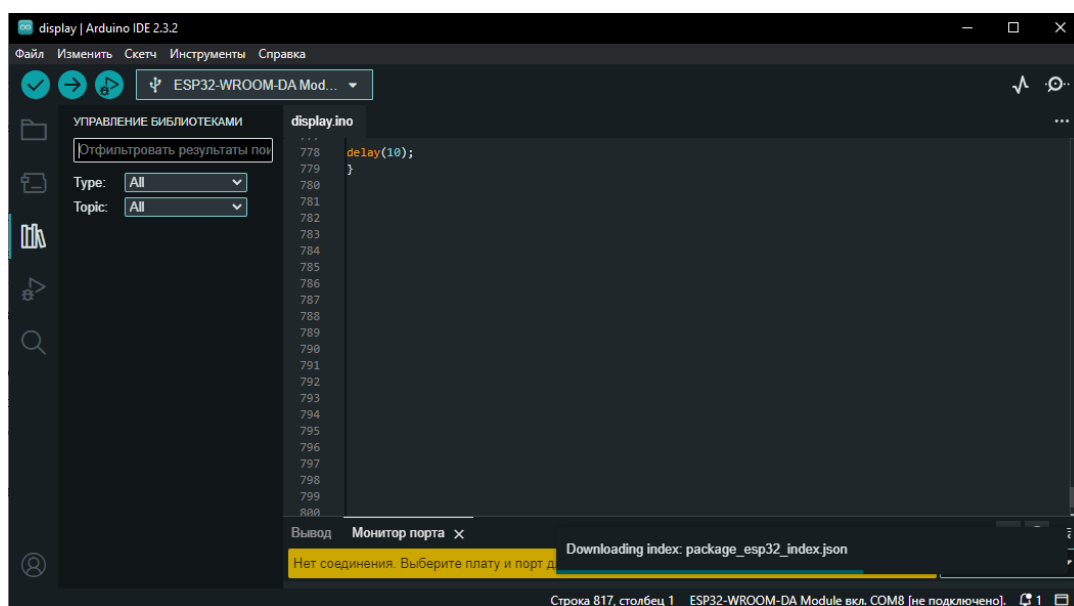


Рисунок 7 – Среда Arduino IDE

Одна из основных и главных преимуществ Arduino IDE – простота использования. Интерфейс IDE интуитивно понятен и легок для освоения даже для тех, кто ранее не разрабатывал устройства на базе ESP и Arduino. Данное преимущество делает процесс разработки более доступным и позволяет быстрее начать создавать свои проекты на ESP32.

Arduino IDE обладает встроенной поддержкой многих систем, в число которых входит микроконтроллер ESP32. Для работы с данной платой даже не требуется устанавливать дополнительные плагины или расширения, так как она уже включена в список поддерживаемых платформ.

Данная среда разработки использует знакомый многим язык программирования C/C++, являющийся стандартом для микроконтроллеров. Это немаловажное преимущество позволяет разработчикам легко писать эффективный и понятный код, используя знакомый синтаксис и структуры данных.

Ко всему прочему, большим плюсом в пользу данной среды является обширная библиотека. Большинство библиотек к тому же, для большей наглядности имеют набор примеров кода. Данная особенность позволяет разработчикам в кратчайших сроках находить решения для своих задач, а используя уже готовые компоненты и функции без необходимости писать код с нуля, уделить больше внимания другим аспектам разработки.

В процессе разработки программы очень важно следить за ошибками и правильной работой каждого этапа выполнения кода. Благодаря инструментам для отладки кода и мониторинга порта, у пользователей появляется возможность отладки проектов, что облегчает и значительно ускоряет процесс разработки. Разработчики могут легко отслеживать вывод программы и взаимодействовать с устройством через последовательный порт, что сильно помогает сократить количество непредвиденных ошибок в работе устройства.

Стоит отметить огромную популярность данной среды разработки. Это очень важно, так как такая особенность позволяет быстрее находить ответы на

вопросы, и следовательно, уходит меньше времени и сил для решения различных проблем.

Исходя из всего вышеперечисленного можно сказать, что Arduino IDE представляет собой мощный и простой в использовании инструмент для разработки данного проекта на микроконтроллере ESP32. Его преимущества включают простоту использования, поддержку ESP32, язык программирования C/C++, обширную библиотеку и примеры кода, инструменты отладки и мониторинга порта, а также поддержку сообщества. Эти возможности позволяют ей быть идеальным выбором для разработчиков, и создавать проекты на большинстве плат с минимальными затратами времени и усилий.

1.4.2 Visual Studio Code (VS Code) для написания веб–страниц

Visual Studio Code (VS Code)– это современный, мощный и довольно продвинутый текстовый редактор, который не без оснований позиционируется как идеальное средство разработки для широкого круга задач, включая создание веб–страниц. Он отличается высокой производительностью, богатым функционалом и обширной поддержкой расширений. Все это делает его популярным выбором среди веб–разработчиков и не только. Вид рабочего пространства представлен на рисунке 8.

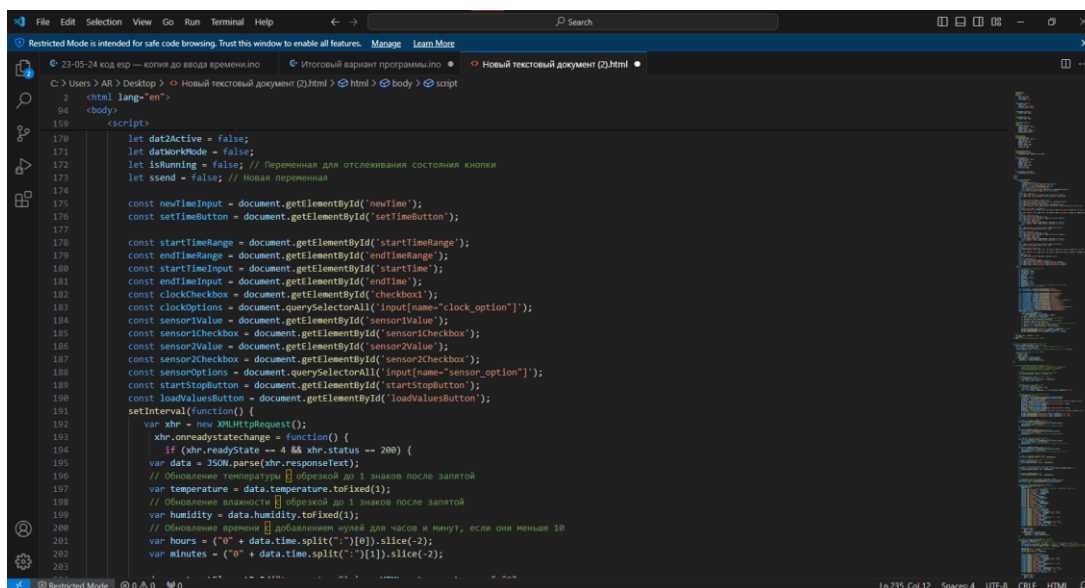


Рисунок 8 – Среда VS Code

VS Code предлагает гибкую настройку и так интуитивно понятного интерфейса. Каждый пользователь имеет возможность адаптировать среду разработки под свои потребности. Возможность поменять каждую мелочь, вплоть до цветовой схемы и расположения панелей инструментов с расширениями делает работу комфортной и максимально эффективной для разработчика.

Одна из ключевых преимуществ VS Code – его мощная база поддержки множества различных языков программирования. В этот список входят HTML, JavaScript и CSS, которые будут использованы в данном проекте. Подсветка синтаксиса, автозавершение кода и многочисленные функции разительно упрощают написание кода и отладку веб-страниц.

Стоит отметить, что VS Code славится своей обширной экосистемой расширений. Эти расширения позволяют добавлять дополнительную функциональность, а также интегрировать различные инструменты и сервисы.

Встроенная поддержка Git делает VS Code отличным выбором для работы с системами контроля версий. Контроль версий имеет удобный интерфейс для работы с репозиториями Git. Это позволяет выполнять коммиты, слияния, просматривать историю изменений из среды разработки.

Просмотр веб-страниц в реальном времени без необходимости обновлять браузер каждый раз вручную – очень полезная особенность. Реализуется это с помощью расширения «Live Server». Это значительно ускоряет процесс разработки и облегчает отладку веб-приложений.

В конечном итоге, Visual Studio Code – максимально подходящий инструмент для разработки веб-страниц. Он обеспечивает высокую производительность, богатый функционал и простоту использования. В конечном итоге, широкие возможности и обширная экосистема расширений делают его идеальным выбором.

1.4.3 Сравнение Arduino IDE с PlatformIO

PlatformIO – среда разработки, которая предоставляет разработчикам удобные инструменты для управления библиотеками, зависимостями и проектами на различных микроконтроллерах. Она также обеспечивает поддержку различных

языков программирования, включая C/C++ и Python. Это делает ее универсальным инструментом для разработки встраиваемых систем.

Однако, несмотря на свои многочисленные преимущества, PlatformIO является более сложным в понимании. Расширенные возможности и богатый функционал, которыми она обладает, могут показаться избыточными для разработки проектов на ESP32.

Arduino IDE является более простой и интуитивно понятной средой. Она имеет более простой интерфейс. Наличие базовых инструментов делают ее идеальным выбором как для опытных, так и для начинающих разработчиков, которые планируют начать путь создания микропроцессорных программ.

Несмотря на отсутствие некоторых продвинутых функций, Arduino IDE остается предпочтительным выбором для многих разработчиков, и имеет большую популярность. Ее обширная база пользовательских библиотек и большое сообщество пользователей позволяют этой платформе называться одной из лучших решений для быстрого создания простых и средних проектов на микроконтроллерах.

Несомненно, PlatformIO и Arduino IDE имеют различия и для их сравнения важно учитывать потребности разработчика и требования к будущим проектам. Для опытных пользователей, которым нужны продвинутые функции и возможности, возможно и подошла бы PlatformIO. Однако, для тех, кто ценит простоту и удобство использования, Arduino IDE остается наилучшим инструментом.

1.4.4 Сравнение VS Code с Sublime Text

Обе среды разработки, Visual Studio Code (VS Code) и Sublime Text, пользуются популярностью среди разработчиков благодаря своей гибкости, производительности и расширяемости. Однако, при сравнении этих инструментов, можно выделить несколько ключевых различий.

VS Code разрабатывается компанией Microsoft, которая уже успела зарекомендовать себя на рынке, и имеет огромное сообщество пользователей. Это обеспечивает регулярные обновления, активную поддержку и развитие инструмента. Sublime Text, напротив, разрабатывается небольшой командой, и его обновления могут быть менее частыми и масштабными.

Также, на первый взгляд, обе среды разработки предоставляют обширный набор функциональных возможностей, но VS Code обычно считается более мощным инструментом благодаря своей обширной экосистеме расширений и интеграций. Не стоит забывать про широкий набор инструментов для веб-разработки, включая интеграцию с Git, поддержку различных языков программирования и удобные средства отладки. Несомненно, Sublime Text также имеет поддержку расширений, но его экосистема, в сравнении, является менее развитой и менее обширной по сравнению с VS Code.

В целом, нельзя сказать, что Visual Studio Code сильно превосходит Sublime Text, и выбор между VS Code и Sublime Text может зависеть от индивидуальных предпочтений и пожеланий разработчика.

1.5 Архитектура проекта

1.5.1 Используемые технологии, протоколы и библиотеки

Для работы были выбраны следующие технологии и протоколы:

- Inter-Integrated Circuit (IIS);
- Wi-Fi;
- HTTP (Hypertext Transfer Protocol);
- HTML (Hypertext Markup Language);
- CSS (Cascading Style Sheets);
- JS (JavaScript);
- JSON (JavaScript Object Notation).

Протокол связи Inter-Integrated Circuit. Он позволяет микросхемам обмениваться данными с помощью подключения двух проводов:

- Serial Data Line (SDA), расшифровывается как линия данных;
- Serial Clock Line (SCL), расшифровывается как линия тактового сигнала.

Каждому устройству присваивается уникальный адрес. В самом начале передачи посылается стартовый сигнал. После этого происходит адресация устройства и указывается, будет передача данных, или же их прием. Данные передаются побайтно, причем каждый байт подтверждается. Простота подключения, при котором требуются только два провода являются основными преимуществами IIS.

Для обеспечения беспроводного соединения и доступа к Интернету была выбрана технология Wi-Fi, которая позволяет устройству подключаться к локальной сети и взаимодействовать с другими устройствами и сервисами через Интернет.

Протокол HTTP используется для обмена данными между веб-сервером на устройстве и клиентскими приложениями, такими как веб-браузеры. Этот протокол обеспечивает передачу запросов и ответов, что позволяет пользователям взаимодействовать с устройством через веб-интерфейс.

Для создания веб–интерфейса были использованы технологии HTML, CSS и JavaScript. HTML определяет структуру страницы, CSS обеспечивает оформление и стиль элементов, а JavaScript добавляет интерактивность и возможность динамического обновления данных на странице.

JSON используется для форматирования и передачи структурированных данных между устройством и клиентским приложением. Этот формат данных является легким и удобным для обмена информацией, такой как настройки устройства и текущие показания с датчиков.

Так же рассмотрим каждую из использованных библиотек.

Wire.h – библиотека для работы с IIS. Позволяет общаться с IIS – устройствами. Основная библиотека для связи с датчиками и часами реального времени через IIS.

Adafruit_Sensor.h – универсальная библиотека для работы с различными датчиками от Adafruit, в число которых входит BME280. Обеспечивает стандартный интерфейс для работы с датчиками температуры и влажности.

Adafruit_BME280.h – библиотека для работы с датчиком BME280. Благодаря данной библиотеке, мы имеем возможность измерять температуру и влажность, используя BME280.

RTCLib.h – требуется если предстоит работа с реальными часами (RTC), такими как DS1307 и DS3231, что требуется для отслеживания времени.

WiFi.h – библиотека для успешной работы с Wi-Fi сетями. Обеспечивает возможность подключения к интернету и локальным сетям для отправки и приема данных.

WebServer.h – благодаря данной библиотеке был создан веб–сервер. Она позволяет создавать веб–интерфейсы для управления и мониторинга устройства через браузер.

Preferences.h – библиотека для хранения данных в Non-Volatile Storage, то есть в энергонезависимой памяти. Используется для сохранения данных, требуемых для подключения к сети.

ESPmDNS.h – обеспечивает поддержку Multicast DNS. Используя данную библиотеку, мы можем объявлять сетевые имена, при этом не используя фиксированные IP–адреса.

ArduinoJson.h – библиотека для работы с форматом JSON. Используется для парсинга и генерации JSON–данных, что удобно для работы с веб–сервисами и API.

esp_task_wdt.h – библиотека предоставляет функциональность для работы с Watchdog Timer (WDT) на ESP32. WDT используется для мониторинга выполнения задач и перезапуска системы в случае зависания или сбоев. Используется для поддержки стабильности и надежности работы устройства.

Использование данных технологий, протоколов и библиотек обеспечили надежное и эффективное управление, а также современный и удобный интерфейс для взаимодействия.

1.5.2 Аппаратная архитектура

Центральным элементом аппаратной архитектуры проекта является микроконтроллер ESP32 (рисунки 9–11). Он обеспечивает основное управление и взаимодействие с другими компонентами системы. Этот микроконтроллер отличается высокой производительностью и широким набором функциональных возможностей.

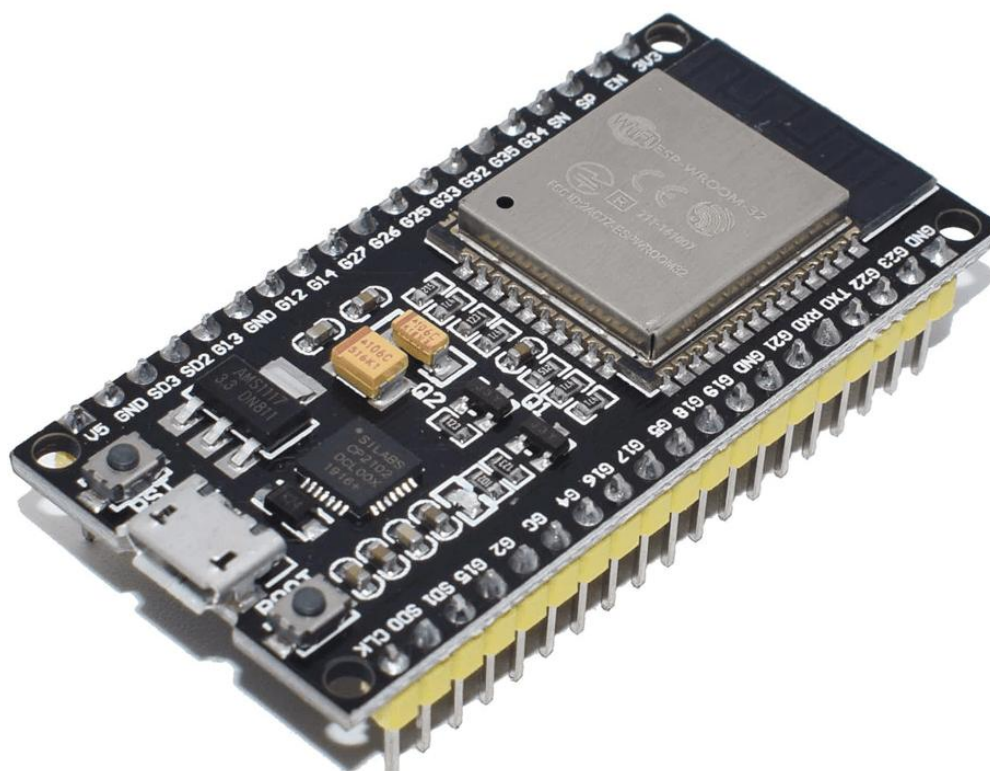


Рисунок 9 – Вид используемой платы на базе ESP32

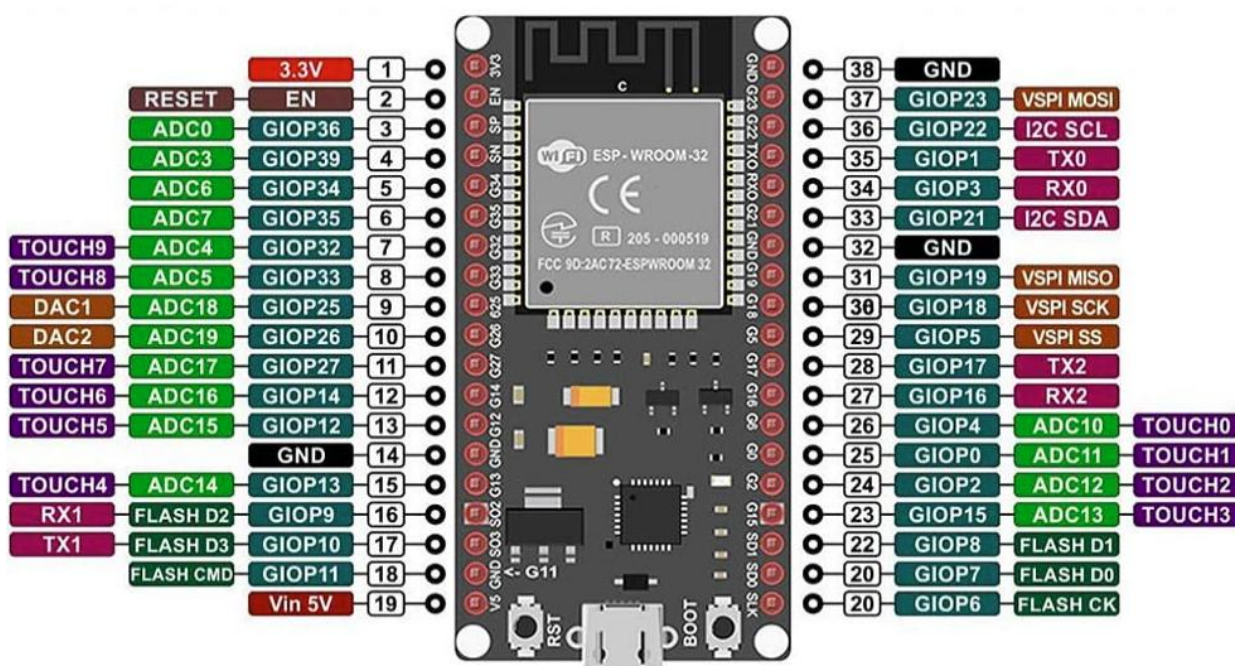


Рисунок 10 – Обозначение доступных к использованию контактов ESP32 WROOM

электромеханическое устройство, способное переключать высокие электрические нагрузки. Для целей данной работы подходит реле с параметрами 10 А 250 В. Этого достаточно для надежного и безопасного управления подключенным прибором.

Взаимодействие между всеми компонентами системы осуществляется через микроконтроллер. Он принимает данные от датчиков, обрабатывает их и, при необходимости, управляет реле. Пользователям предоставляется возможность мониторить параметры окружающей среды, а также управлять подключенными электроприбором с помощью беспроводного соединения.

У ESP32 имеется ряд преимуществ и особенностей, что делает его популярным выбором.

Одним из плюсов можно выделить мощность и производительность, благодаря двухъядерному процессору и частоте работы до 240 МГц, чего вполне достаточно под цели данной работы. Интегрированные Wi-Fi, обеспечивающие скорость передачи данных до 150 Мбит/с и Bluetooth для беспроводного подключения и коммуникации с другими устройствами. Большое количество цифровых и аналоговых портов GPIO, позволяют подключить большое количество периферии. Поддержка различных протоколов и интерфейсов, таких как SPI, I2C, UART делают процесс разработки более удобным, и позволяют использовать наиболее эффективные и подходящие способы взаимодействия компонентов. Для выполнения задач, а также хранения программ и данных плата оснащена 520 кБ оперативной памяти и 4 МБ флэш-памяти. Имея столь большое количество преимуществ, данная плата все еще может похвастаться низким энергопотреблением и сравнительно небольшой стоимостью, что является очень большим плюсом для использования именно этого варианта. [1]

Для определения и подсчета времени выбран модуль часов реального времени DS3231 (рисунок 12, 13), так как он имеет множество преимуществ в сравнении с аналогами. DS3231– высокоточный интегрированный силовой модуль реального времени (RTC) с кварцевым генератором на кристалле. В данной плате предусмотрен элемент питания CR2032 с напряжением 3 вольта,

благодаря ему время не будет сбрасываться даже при отключении основного питания.



Рисунок 12 – DS3231

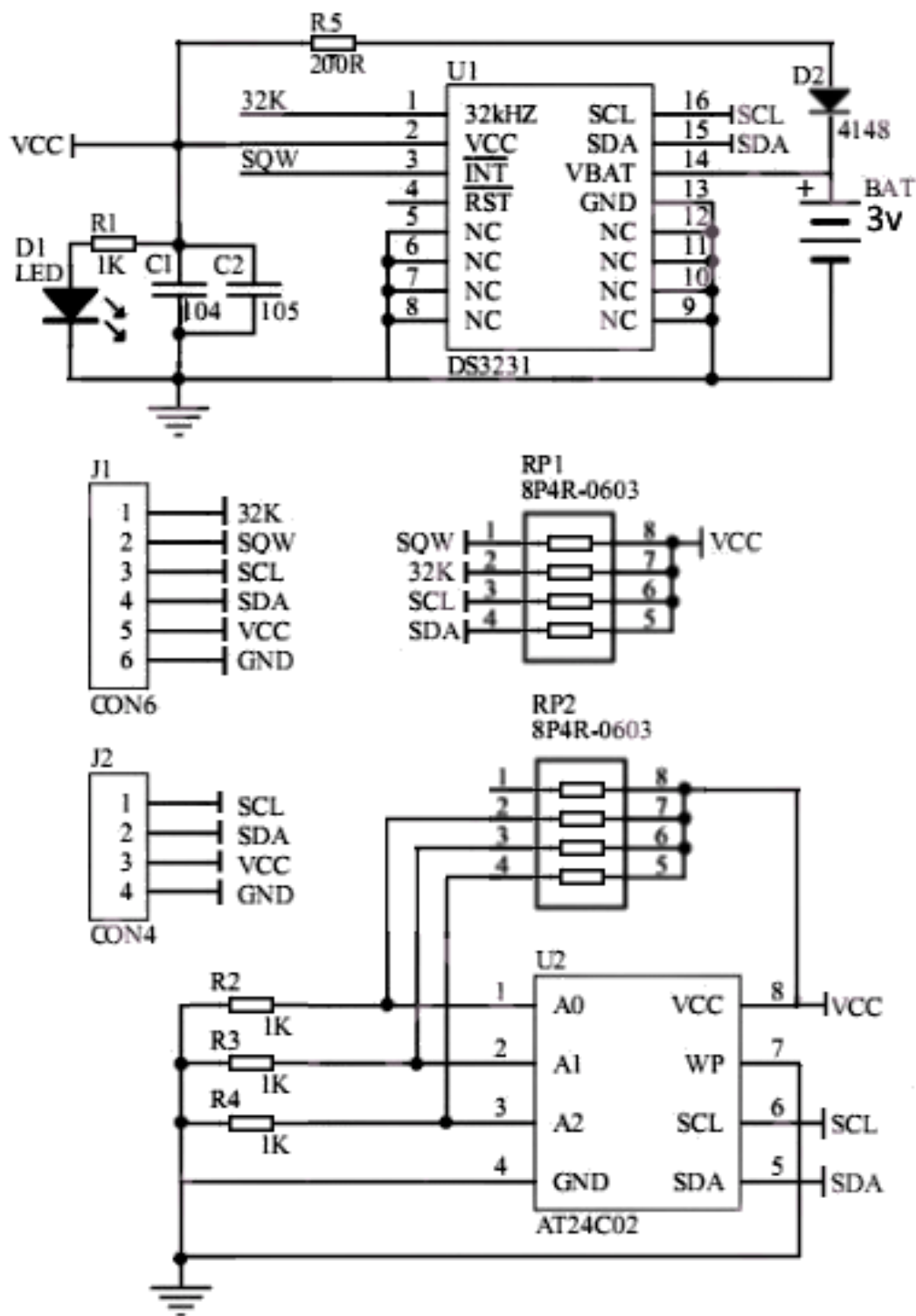


Рисунок 13 – Принципиальная схема DS3231

DS3231 имеет ряд особенностей, на которых стоит заострить внимание.

DS3231 обладает сравнительно высокой точностью хода времени. Кварцевый генератор на кристалле имеет температурную компенсацию, это обеспечивает стабильность и точность в широком диапазоне температур. При выборе RTC, также важна поддержка I2C интерфейса, и данный вариант удовлетворяет этим запросам. Ко всему прочему, он имеет встроенный календарь с функциями времени, даты и дня недели, встроенные таймеры, которые можно настраивать под различные параметры, такие как формат времени, режим работы, сигналы прерывания. Все это обеспечивает хорошую гибкость в использовании и настройке. При всех своих достоинствах, данный вариант сохраняет низкое энергопотребление.[2]

Для отслеживания температуры и влажности был использован модуль BME280 (рисунок 14, 15), так как обладает лучшими характеристиками, нежели его аналоги. Разберем подробнее технические характеристики и особенности модуля BME280.



Рисунок 14 – BME280

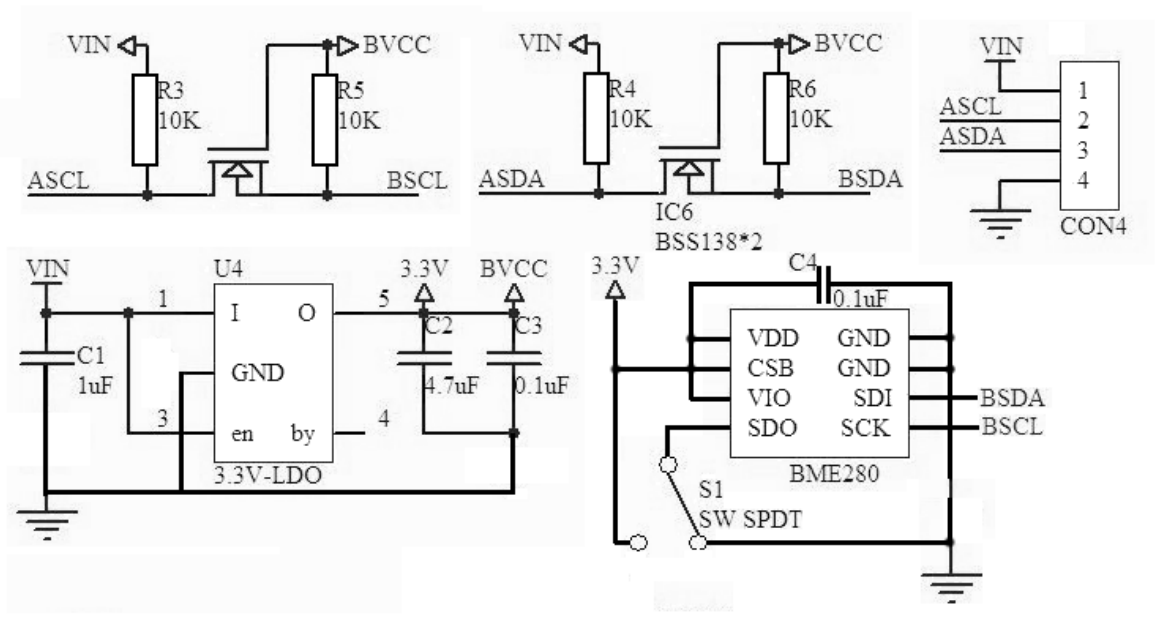


Рисунок 15 – Принципиальная схема BME280

BME280 – это высокоточный интегрированный сенсорный модуль, который предоставляет информацию об окружающей среде как температура, влажность и давление. Если сравнивать его с другими вариантами, он отличается высокой точностью и стабильностью измерений. BME280 работает в широком диапазоне температур, влажности и давления.

Все эти особенности делают модуль BME280 универсальным и многофункциональным инструментом для измерения и мониторинга погодных условий и параметров окружающей среды.[3]

Для управления конечными электроприборами выбор пал на модуль реле максимальным напряжением 250В и силой тока 10А (рисунок 16, 17).



Рисунок 16 – Модуль реле с максимальным напряжением 250В и силой тока 10А

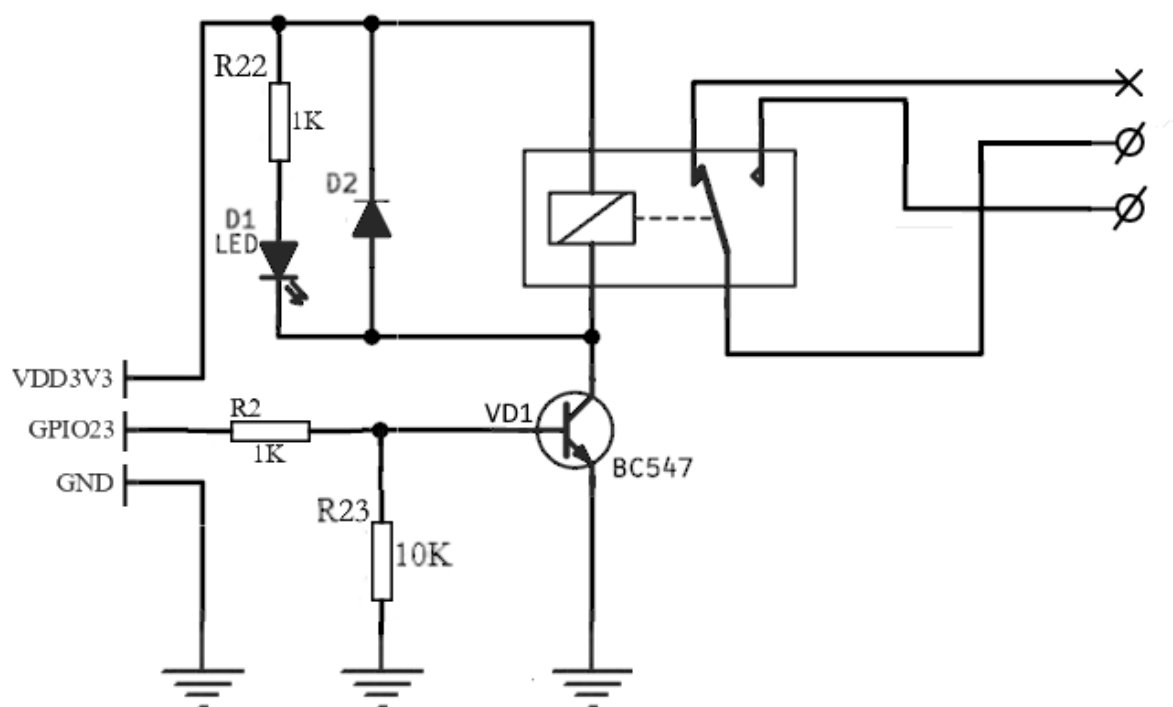


Рисунок 17 – Принципиальная схема реле

Реле является электромеханическим устройством, используемым для управления электрическими нагрузками с высоким напряжением и током. Данное реле обладает рядом технических преимуществ, и превосходит аналоги в своем ценовом сегменте. В первую очередь, данное реле имеет один нормально разомкнутый и один нормально замкнутый контакт. Благодаря исполнению из качественных материалов и тщательному производству обеспечивается надежность и долговечность. Данное реле имеет широкий круг применения. К примеру, лампы, вентиляторы, насосы, обогреватели, электродвигатели и другие устройства, которые требуют управления электрическим током.

После рассмотрения всех компонентов аппаратной архитектуры проекта, можно сделать следующий общий итог:

Проект основан на микроконтроллере ESP32, он же центральный элемент системы. Он обеспечивает управление всеми компонентами и передачу данных.

Для беспроводной связи используется встроенный модуль Wi-Fi. Часы реального времени и датчик температуры и влажности (DS3231 и BME280) используются для мониторинга параметров окружающей среды. Они обеспечивают точное измерение времени, температуры и влажности. Реле используется для управления подключаемым электроприбором.

ESP32 обрабатывает данные и управляет через реле включением и выключением электроприбора в соответствии с заданными параметрами и условиями.

1.5.3 Программная архитектура

Программная архитектура представляет собой комплексный подход к управлению включения и выключения подключаемого электроприбора через веб-интерфейс. Взаимодействие между аппаратными компонентами и программными модулями позволяет пользователю контролировать устройство, а также получать актуальную информацию о состоянии датчиков. Программный код управления микроконтроллером разработан на языке C++. Работа датчика температуры с влажностью, часов реального времени, реле, а также технологии Wi-Fi и веб-сервера обеспечивается специализированными библиотеками, рассмотренными ранее.

В функции `setup()` происходит начальная настройка устройства. Сюда входит настройка интерфейса I2C для взаимодействия с BME280 и DS3231, инициализация переменных и объектов для работы с датчиками BME280 и RTC DS3231, установка режимов работы выводов микроконтроллера. После этого, программа пытается подключиться к беспроводной сети, по введенным используя сохраненные данные. В случаях, если название сети или пароль введены некорректно, или же данные для подключения вовсе отсутствуют, а также при прочих ошибках подключения программа вызывает функцию `startAPMode()`. В данном случае запускается собственная сеть «ESP» (пароль по умолчанию 12345678), отображается страница для ввода нового названия и пароля для сети (рисунки 16-18), после ввода данных на которой, потребуется перезагрузить устройство для повторной попытки подключения. Если же получилось присоединиться к сети, то отображается основная страница для управления, ввода данных и мониторинга значений (рисунок 19, 20), а также происходит обработка запросов на обновление значений датчиков и времени на странице с определенным интервалом.

Home icon | **esp32.local** | + | 34 | ⋮

Enter WiFi Credentials

Last connect IP: 192.168.238.18

SSID:

Password:

Save

Рисунок 16 – Страница ввода названия и пароля сети

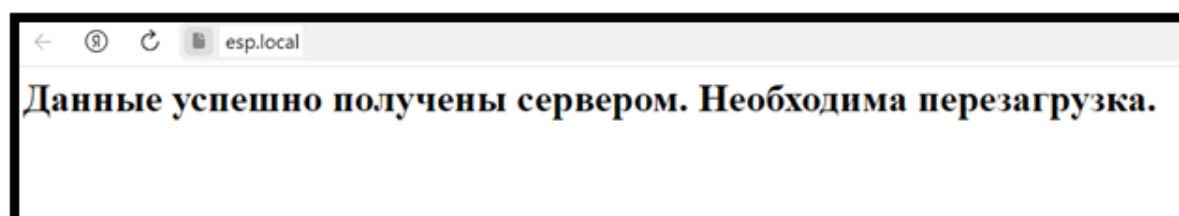


Рисунок 17 – Страница оповещения успешного ввода данных



Рисунок 18 – QR код для подключения к сети ESP

The screenshot shows a web browser window with the title "ESP32 Project". The address bar shows "192.168.0.13". The main content area has a light orange background. It features several sections: "Текущие данные:" (Current data) with temperature (28.4 °C), humidity (43.4 %), and time (23:20); "Условия времени" (Time conditions) with a slider and time range (00:00 to 24:00); "Работа часов в паре с датчиками" (Clock operation with sensors) with radio buttons for "И" (AND) and "ИЛИ" (OR); "Датчик 1 'Термометр'" (Sensor 1 'Thermometer') and "Датчик 2 'Гигрометр'" (Sensor 2 'Hygrometer') with input fields for values and modes; "Работа датчиков в паре" (Sensor operation in pairs) with radio buttons for "И" (AND) and "ИЛИ" (OR); "АВТОРЕЖИМ" (AUTO MODE) with a "Загрузить значения" (Load values) button; and "Ручное управление" (Manual control) with a "Запустить" (Start) button.

ESP32 Project

192.168.0.13

Текущие данные:

Температура: 28.4 °C

Влажность: 43.4 %

Время: 23:20

Автоматический режим: ●

Состояние реле: ●

Новое время: 00:00:00 Задать новое время

Условия времени

Начало времени: 00:00 Конечное время: 24:00 ☐ Время

Работа часов в паре с датчиками

☐ И ☒ ИЛИ

Датчик 1 "Термометр"

Значение: 0 Режим: ">" Сраб

☐ Активировать

Датчик 2 "Гигрометр"

Значение: 0 Режим: ">" Сраб

☐ Активировать

Работа датчиков в паре

☐ И ☒ ИЛИ

АВТОРЕЖИМ

Загрузить значения

Ручное управление

Запустить

Рисунок 19 – Главная страница web-приложения. Компьютерная версия

Текущие данные:

Температура: 29.0 °C

Влажность: 44.0 %

Время: 03:07

Автоматический режим: ☒

Состояние реле: ☒

Новое время:

Задать новое время

Условия времени

Начало времени: Конец

времени: ☐ Время

Работа часов в паре с датчиками

☐ и

☒ или

Датчик 1 "Термометр"

Значение: Режим:

☐ Активировать

Датчик 2 "Гигрометр"

Значение: Режим:

☐ Активировать

Работа датчиков в паре

☐ и

☒ или

АВТОРЕЖИМ

Загрузить значения

Ручное управление

Запустить

Рисунок 20 – Главная страница web–приложения. Мобильная версия

В функции loop() происходит проверка на наличие http запросов, при наличии которых вызываются соответствующие функции, такие как:

handleSendData(), эта функция вызывается при нажатии одной из кнопок «Загрузить значения» или «Запустить\Остановить» на странице. Она принимает введенные значения со страницы, на основе которых строится логика работы автоматического режима, а также данные для переключения режима работы в ручное управление и обратно. handleSetTime() вызывается после нажатия кнопки установки времени на странице. Данная функция служит для настройки времени, и обновляет значение часов, минут и секунд на введенные на странице значения. Также в функции loop() происходит вызов функции updateRelayState(). Данная функция проверяет все используемые в логике переменные, и сопоставляя данные со страницы, с датчиков и с часов выполняет проверку на соблюдение всех условий для включения или отключения реле, а если автоматический режим выключен, в этой же функции происходит обработка ручного режима управления. В конечном итоге все сводится к активации или же деактивации реле. Далее, функция loop() проверяет состояние экстренной кнопки, нажатие на которую переводит работу в режим ручного управления и включает или же выключает реле. Для безопасности и исключения неправильной работы при непредвиденных ошибках реализован Watchdog Timer (WDT). Сторожевой таймер сбрасывается в функции loop() каждые 10 секунд.[4]

1.5.4 Разработка веб–интерфейса

В данной главе более подробно рассмотрен процесс разработки веб–интерфейса. Важность данного этапа проекта состоит в обеспечении эффективного взаимодействия пользователя с устройством, чтобы позволить легко настраивать важные параметры и иметь возможность мониторинга актуальной информации о состоянии.

Внешний вид и пользовательский интерфейс. Особое внимание было уделено дизайну веб–страницы с учетом принципов удобства использования и визуальной привлекательности. Для структурирования деталей и создания гармоничного визуального оформления использовались HTML и CSS. На первый взгляд, не такие технически важные детали, как цветовое решение и оформление красивого дизайна, на деле сильно влияют на понимание всей концепции. Потому цветовая гамма была выбрана с учетом легкости восприятия и создания приятного визуального впечатления.

Интерактивность и функциональность. Для обогащения пользовательского опыта были добавлены интерактивные элементы, например, ползунки для более быстрой и удобной настройки границ времени, которые реализованы с использованием JavaScript. Это позволяет получить пользователю положительный опыт взаимодействия с устройством через веб–браузер, делая интерфейс более функциональным и удобным в использовании.

Взаимодействие с устройством. Взаимодействие веб–интерфейса с устройством основано на HTTP запросах. Благодаря чему пользователям дается возможность управлять работой устройства по беспроводной сети через веб–браузер. При выполнении действия пользователем на веб–странице, HTTP запросы отправляются на устройство, где они уже обрабатываются, и выполняются соответствующие операции. Такой механизм взаимодействия обеспечивает удобство использования устройства. Это работает и в обратную сторону. После получения новых данных от устройства или при изменении его состояния, веб–интерфейс обновляет информацию на странице, предоставляя мгновенный доступ к актуальной информации.

Адаптивность к мобильным устройствам. Важный аспект разработки – адаптивность веб–интерфейса к различным типам устройств, в основном это мобильные телефоны и планшеты. Оно позволило страницам автоматически подстраиваться к размеру экрана устройства, обеспечивая оптимальное отображение, следовательно, удобство использования.

Тестирование и оптимизация. Важно проверить ошибки и корректную работу web–страницы. Поэтому после завершения разработки были проведены различные тесты работоспособности. Для этого страница открывалась на разных устройствах, будь то персональный компьютер или мобильное устройство на Android или iOS. На каждом из устройств было проведено тестирование работы в различных браузерах. В тесты входили проверка функциональности работы кнопок, полей ввода, проверка адаптивности дизайна в книжном и альбомном отображении страницы, а также отзывчивость интерфейса. Моменты, которые работали некорректно были исправлены. Ко всему прочему была проведена оптимизация интерфейса для обеспечения стабильной и эффективной работы на всех устройствах.

Так как веб–интерфейс является частью наибольшего взаимодействия пользователя с устройством, её разработка является ключевым этапом в создании функционального и удобного в использовании устройства. В конечном итоге, удалось добиться того, что пользовательский интерфейс обеспечивает удобство настройки параметров устройства и получения актуальной информации о его состоянии, что в свою очередь обеспечивает успешное функционирование устройства, а следовательно, повышает удовлетворенность пользователей.[5]

1.5.5. Реализация логики управления электроприборами

Реализация логики управления – важный этап в разработке проекта. В этом рассмотрены основные аспекты программирования логики управления для понимания процесса управления электроприборами.

Анализ условий управления. Прежде всего, для управления электроприборами необходимо определить условия, при которых они должны включаться или выключаться. Сюда входят различные факторы, такие как время суток, температура, влажность, статус кнопки. Недостаточно просто указать пороговые значения датчиков. Ко всему прочему, в специально отведенных полях необходимо выставить условия для формирования самой логики.

Программирование логики управления. Далее происходит программирование логики управления. На основе собранной со страницы веб-приложения информации соответствующим переменным присваиваются значения от различных источников, таких как датчики температуры и влажности, а также внутренние часы реального времени (RTC). После этого программа проверяет значения всех переменных и в зависимости от выполнения условий для данных переменных выполняется участок кода, выстраивая логику работы. После того как в коде микроконтроллера ESP32 определяется логика, происходит сравнение значений датчиков и принятых пороговых значений. Программа оценивает текущие условия и принимает решение о включении или выключении реле.

Обработка данных. Данные с датчиков постоянно меняются, и важно их постоянно обновлять и отслеживать. Поэтому в каждом цикле программы вызывается опрос данных, а после получения данных от датчиков и других источников информации происходит их обработка. Так как мы используем датчики температуры и влажности, а также таймер, считывается температура, влажность и время, и затем сравниваются с заданными пороговыми значениями в коде для принятия решения об управлении.

Управление реле. Управление электроприборами, как было сказано ранее, осуществляется с помощью реле. В зависимости от условий, определенных в программе, и их выполнения на вход реле подается LOW или HIGH. Таким

образом, реле может быть включено или выключено для соответствующего управления электроприбором.

Реагирование на внешние события. Кроме того, программирование логики управления в целях устранения непредвиденных случаев и для расширения функционала с повышением эффективности и удобства предусматривает реагирование на внешнее событие, такое как нажатие кнопки. Программа спроектирована так, что если пользователь нажимает на кнопку, это приводит к мгновенному выключению или включению электроприбора в зависимости от его предыдущего состояния. Ко всему прочему, нажатие экстренной кнопки автоматически переключает устройство в ручной режим управления, выключая автоматический режим. В целях безопасности после данных действий в автоматический режим можно перейти только через веб-приложение, указав новую логику и нажав кнопку «Загрузить значения».

Программирование логики управления электроприборами на ESP32 требует внимательного анализа условий управления и правильной реализации алгоритмов в коде микроконтроллера. Это позволяет создать эффективную систему управления, учитывающую различные факторы и обеспечивающую надежную работу электроприборов в соответствии с заданными условиями.

1.6 Тестирование и отладка

Тестирование и отладка играют важную роль в обеспечении надежной работы устройства. В данном разделе были рассмотрены различные случаи тестирования платы, включая проверку работы при различных температурах, экстренные включения и выключения, тест времени и тест на влажность.

1. Тестирование при различных температурах:

Нормальные условия: Плата тестируется при комнатной температуре (около 25°C) для проверки ее работы в стандартных условиях эксплуатации. Мы выставили логику для включения при превышении температуры 27 градусов, без участия часов и второго датчика. Далее, модуль BME280 был нагрет на пару градусов. Программа моментально отреагировала на изменения и успешно включила реле. Результат представлен на рисунке 21.

Текущие данные:

Температура: 27.3°C
Влажность: 53.5%
Время: 13:17
Автоматический режим: ☒
Состояние реле: ☒
Новое время:
Условия времени

Начало времени: Конец времени: ☐ Время
Работа часов в паре с датчиками
☐ И
☒ ИЛИ
Датчик 1 "Термометр"
Значение: Режим:
☒ Активировать

Рисунок 21 – Успешное испытание путем изменения окружающей температуры.

2. Экстренные включения и выключения:

Включение и выключение нажатием отдельной кнопки на веб–странице: проверяется, как плата реагирует на команду включения\выключения несмотря на условия логики, чтобы удостовериться, что она может быть экстренно отключена.

Включение и выключение нажатием отдельной кнопки на плате: проверяется, как плата реагирует на команду включения\выключения несмотря на условия логики, чтобы удостовериться, что она может быть экстренно отключена. Результат представлен на рисунках 22-24.

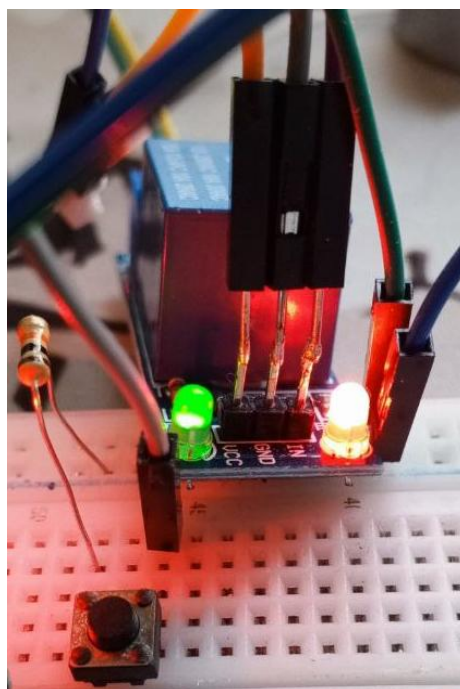


Рисунок 22 – Включение реле после нажатия на кнопку на плате/web–странице

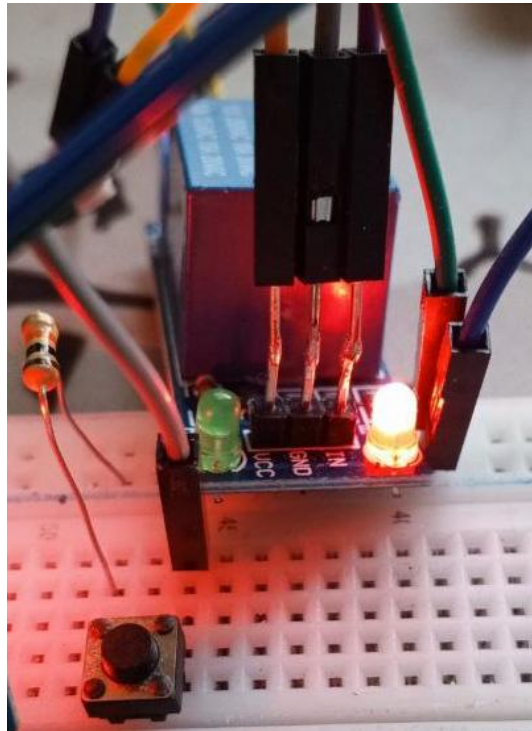


Рисунок 23 – Выключение реле после нажатия на кнопку на плате/web-
странице

Текущие данные:

Температура: **24.7°C**

Влажность: **62.5%**

Время: **13:21**

Автоматический режим: **●**

Состояние реле: **●**

Рисунки 24– Отображение состояния реле и режима работы на web-
странице

3. Тест времени:

Долгосрочное тестирование: Плата подвергается долгосрочному тестированию в течение нескольких часов, чтобы проверить ее стабильность и надежность работы в течение продолжительного времени.

Тест перехода через предел указанного промежутка времени: проверяется, как плата реагирует на переход границу времени, чтобы удостовериться, что она правильно обрабатывает изменение времени. В 11:25 выставили рабочее время с 12:45 до 19:00. Результат теста представлен на рисунках 25-27.

The screenshot displays a control interface with the following sections:

- Текущие данные:**
 - Температура: 24.3°C
 - Влажность: 52.2%
 - Время: 11:25
 - Автоматический режим: ☒
 - Состояние реле: ☐
 - Новое время:
- Условия времени:**
 - Visual timeline with two blue markers.
 - Начало времени: Конец времени: ☒ Время
- Работа часов в паре с датчиками:**
 - ☒ И
 - ☐ ИЛИ
- Датчик 1 "Термометр":**
 - Значение: Режим: >" Сраб"
 - ☐ Активировать

Рисунок 25 – Выключенное состояние реле при недостаточном времени до достижения порогового значения.

Текущие данные:

Температура: 24.8°C

Влажность: 57.3%

Время: 12:45

Автоматический режим: ☒

Состояние реле: ☒

Новое время: 11:25:00

Задать новое время

Условия времени



Начало времени: 12:45

Конец времени: 19:00

☒ Время

Работа часов в паре с датчиками

- ☒ И
☐ ИЛИ

Датчик 1 "Термометр"

Значение: 0

Режим: ">" Сраб

☐ Активировать

Рисунки 26 – Включение реле при нахождении времени между начальным и конечным порогом

Текущие данные:

Температура: 26.0°C
 Влажность: 43.6%
 Время: 19:03
 Автоматический режим: ●
 Состояние реле: ●

Новое время: Задать новое время

Условия времени

Начало времени: Конец времени: ☒ Время

Работа часов в паре с датчиками

☒ И
☐ ИЛИ

Датчик 1 "Термометр"

Значение: Режим: ">" Сраб ▼

☐ Активировать

Рисунки 27 – Выключение реле при выходе времени из промежутка начального и конечного порога

4. Тест на влажность:

Нормальные условия влажности: Плата тестируется в нормальных условиях влажности для оценки ее работы в типичных окружающих условиях. Для этого выставлена логика для включения при снижении уровня влажности до менее чем 60%. Процесс тестирования и результат представлены на рисунках 28-30.

Датчик 2 "Гигрометр"

Значение: Режим: "<" Сраб ▼

☒ Активировать

">" Срабатывание при значении >
 "<" Срабатывание при значении <

Рисунок 28 – Процесс настройки логики срабатывания при значении датчика менее 60%.

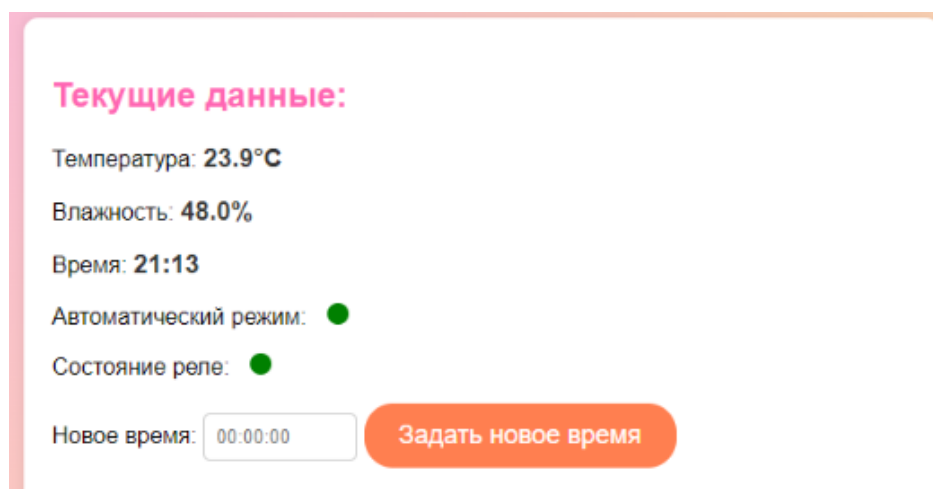


Рисунок 29 – Результат тестирования логики срабатывания при значении датчика менее 60%.

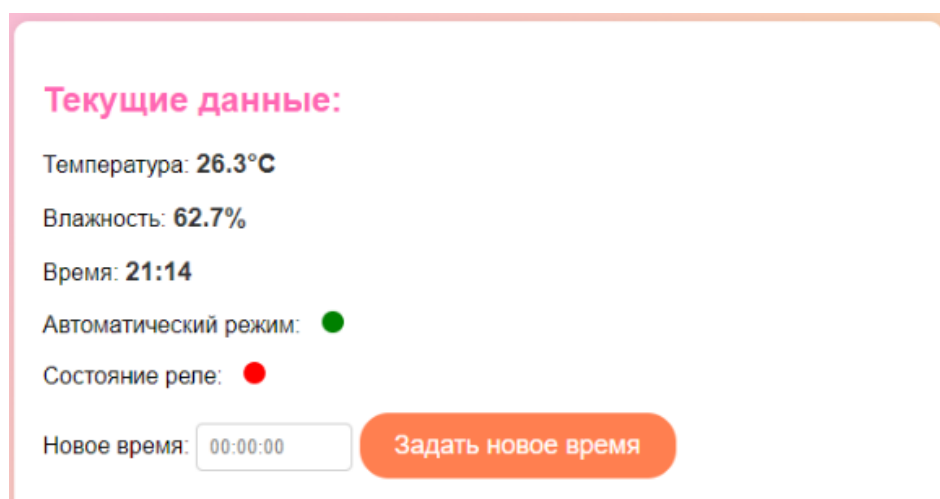


Рисунок 30 – Результат тестирования логики срабатывания при значении датчика менее 60%.

5. Различные комбинации условий:

Плата также тестируется на работу при различных комбинациях условий, например, превышении температуры и влажности, чтобы убедиться в ее правильной логике и устойчивости к разнообразным факторам окружающей среды. Для примера, выставлена работа при температуре выше 25 градусов или влажности меньше 55%. Если хотя бы одно из условий будет выполняться – реле будет работать. Процесс и результат тестирования представлены на рисунках 31-33.

Датчик 1 "Термометр"

Значение:

25

Режим:

">" Сраб 

☒ Активировать

Датчик 2 "Гигрометр"

Значение:

60

Режим:

"<" Сраб 

☒ Активировать

Работа датчиков в паре

☐ И

☒ ИЛИ

Рисунок 31 – Процесс настройки логики срабатывания при значении датчика менее 60% или температуры больше 25 градусов

Текущие данные:

Температура: 26.3°C

Влажность: 62.7%

Время: 21:17

Автоматический режим: 

Состояние реле: 

Рисунок 32 – Результат работы логики срабатывания при значении датчика менее 60% или температуры больше 25 градусов



Рисунок 33 – Результат работы логики срабатывания при значении датчика менее 60% или температуры больше 25 градусов

Тестирование и отладка платы включают в себя множество сценариев, позволяя обеспечить ее стабильную и надежную работу в самых разных условиях эксплуатации. При тестах нарушений выявлено не было и во всех рассмотренных сценариях устройство сработало согласно указанной логике.

2 Безопасность жизнедеятельности

При взаимодействии с компьютером, необходимо соблюдать меры безопасности, чтобы не было неприятных последствий для здоровья после использования. Если пользователь ПК будет пренебрегать техникой безопасности, то проблемы со здоровьем будут неизбежны.

2.1 Вред здоровью при работе с ПК

В процессе взаимодействия с компьютером могут появиться некоторые проблемы со здоровьем, которые имеют свойство накапливаться. Это значит, что здоровье ухудшается не сразу, а постепенно. Рассмотрим наиболее опасные факторы:

—нагрузка на суставы. Если работать за компьютером продолжительное время в неправильной посадке, на суставы и особенно кисти рук будет идти большая нагрузка, что может привести даже к онемению рук;

—нагрузка на спину. При том же неправильном положении осанки во время работы за компьютером, есть шанс на то, что появится искривление позвоночника, грыжа межпозвоночного диска;

—нагрузка на органы зрения. Монитор оказывает нагрузку на глаза, и если долго смотреть в монитор, будет ухудшаться здоровье глаз, что может привести к синдрому «красного глаза» и «сухого глаза»;

—нагрузка на психику и нервную систему. При долгом взаимодействии с ПК, у пользователя могут возникать проблемы с памятью, а также может появиться депрессия и даже бессонница.

2.2 Снижение вреда для здоровья при работе с ПК

Далее приведены простые правила, соблюдение которых может помочь избежать серьезных проблем со здоровьем:

— соблюдение необходимого расстояния до монитора. Правильным расстоянием до дисплея является расстояние, при котором пользователь может коснуться указательным пальцем до верхнего края монитора;

— соблюдение оптимального угла экрана. Оптимальным углом является наклон монитора в 30 градусов;

— правильное расположение клавиатуры. Клавиатура должна находиться на расстоянии 20–30 сантиметров от края стола;

— организация перерывов. Примерно каждый час необходимо делать разминку для органов зрения, а именно отвлечься от монитора и делать гимнастику для глаз: фокусировать внимание на дальние объекты и переводить фокус с одних предметов на другие, рисовать различные фигуры глазами. Также примерно каждые полтора–два часа необходимо выполнять зарядку;

— подбор правильного с точки зрения поддержки осанки кресла. При выборе данной мебели, необходимо приобрести такую, которая будет обеспечивать прямую осанку, при которой спина будет опираться в спинку.

2.3 Техника безопасности при работе с компьютером

При работе с компьютером необходимо соблюдать правила, связанные с техникой безопасности:

- не оставлять компьютер без присмотра на длительное время;
- не использовать компьютер, который работает от питания без заземления;
- не работать за ПК, у которого есть повреждение в системном блоке или в его периферийных устройствах;
- не включать компьютер, если в помещении слишком высокая влажность;
- не класть на корпус компьютера или монитора посторонние предметы.

2.4 Требования безопасности при использовании устройства, коммутирующего сетевое напряжение 220В

Для предотвращения серьезных инцидентов, таких как поражение электрическим током, пожары и повреждение оборудования, критически важно соблюдать требования безопасности. Надлежащая изоляция, использование качественных компонентов, регулярное обслуживание и тестирование устройства – путь к минимизации рисков и обеспечению надежной и безопасной эксплуатации. Важно информировать пользователей о правилах безопасности и правильного пользования устройством.

Перечень правил техники безопасности:

- не оставлять устройство без присмотра на длительное время;
- не использовать электроприборы, которые работают от питания без заземления;
- все проводящие части устройства должны иметь надёжную изоляцию. Корпус должен обеспечивать защиту от прикосновения к элементам, которые могут быть под напряжением;
- важно регулярно проверять механизм на предмет повреждений, износа и соответствия нормам безопасности. К этому относится также тестирование изоляции;
- пользователь должен быть хорошо проинструктирован о правилах безопасного использования устройства, а также уметь оказывать первую помощь в экстренных случаях, таких как поражение электрическим током;
- Должны соблюдаться требования условий, соответствующих техническим характеристикам (температура, влажность, отсутствие агрессивных сред и т.д.).

2.5 Требования условий эксплуатации устройства управления электроприборами на основе микроконтроллера ESP32

С учетом всех составляющих элементов и их технических условий работы была разработана собственные требования условий к эксплуатации самого прибора:

- температура эксплуатации: от -40°C до $+85^{\circ}\text{C}$;
- влажность: от 5% до 85% без конденсации;
- диапазон давления: от 150 мм рт. ст. до 850 мм рт. ст.;
- напряжение питания платы: 3.6 – 5 В.
- максимальное напряжение электроприбора: до 250 В;
- Максимальный потребляемый ток электроприбора: до 10 А.

Заключение

В ходе выполнения данной выпускной квалификационной работы были получены навыки написания логики и программ для работы микропроцессорных плат ESP32 в роли контролирующих устройств, а также написания web-приложений для взаимодействия с данными платами. Результатом работы является разработанное устройство для управления подключенным электроприбором на базе микроконтроллера ESP32, а также отчёт, в котором представлены результаты тестирования разработанного решения, описание разработки и подробный разбор его основных функциональных особенностей.

Разработанный аппарат позволяет пользователю, путем взаимодействия в браузере увидеть состояние подключенного электроприбора, включать и выключать его экстренно, задавать логику автоматического режима работы с дальнейшим переводом работы в автоматический режим. После проведения тестирования было выявлено, что разработка соответствует всем поставленным в начале работы требованиям и работает корректно, не вызывая критических ошибок в системе.

В дальнейшем можно добавить усовершенствования, путем добавления поддержки других типов датчиков, замены реле на более мощное, способное выдерживать больший ток и напряжение. Разработанный интерфейс пользователя можно улучшить путем добавления других элементов логики, переключаемых оформлений дизайна.

Список использованной литературы

1. Техническое руководство ESP32. [Электронный ресурс] /Режим доступа: https://www.espressif.com/sites/default/files/documentation/ESP32_technical_reference_manual_en.pdf , свободный. (дата обращения 23.04.2024).
2. Техническое описание DS3231. [Электронный ресурс]/Режим доступа: <http://arduino-kid.ru/sites/default/files/2019-04/DS3231.pdf> , свободный.
3. Техническое описание BME280. [Электронный ресурс]/Режим доступа: <https://download.mikroe.com/documents/datasheets/BME280.pdf> , свободный. (дата обращения 23.04.2024).
4. Бьерн Страуструп. "Язык программирования C++". 2019 – 322 с.
5. Джон Дакетт. "HTML и CSS: дизайн и создание веб-сайтов". 2011 – 512 с.

Приложение А

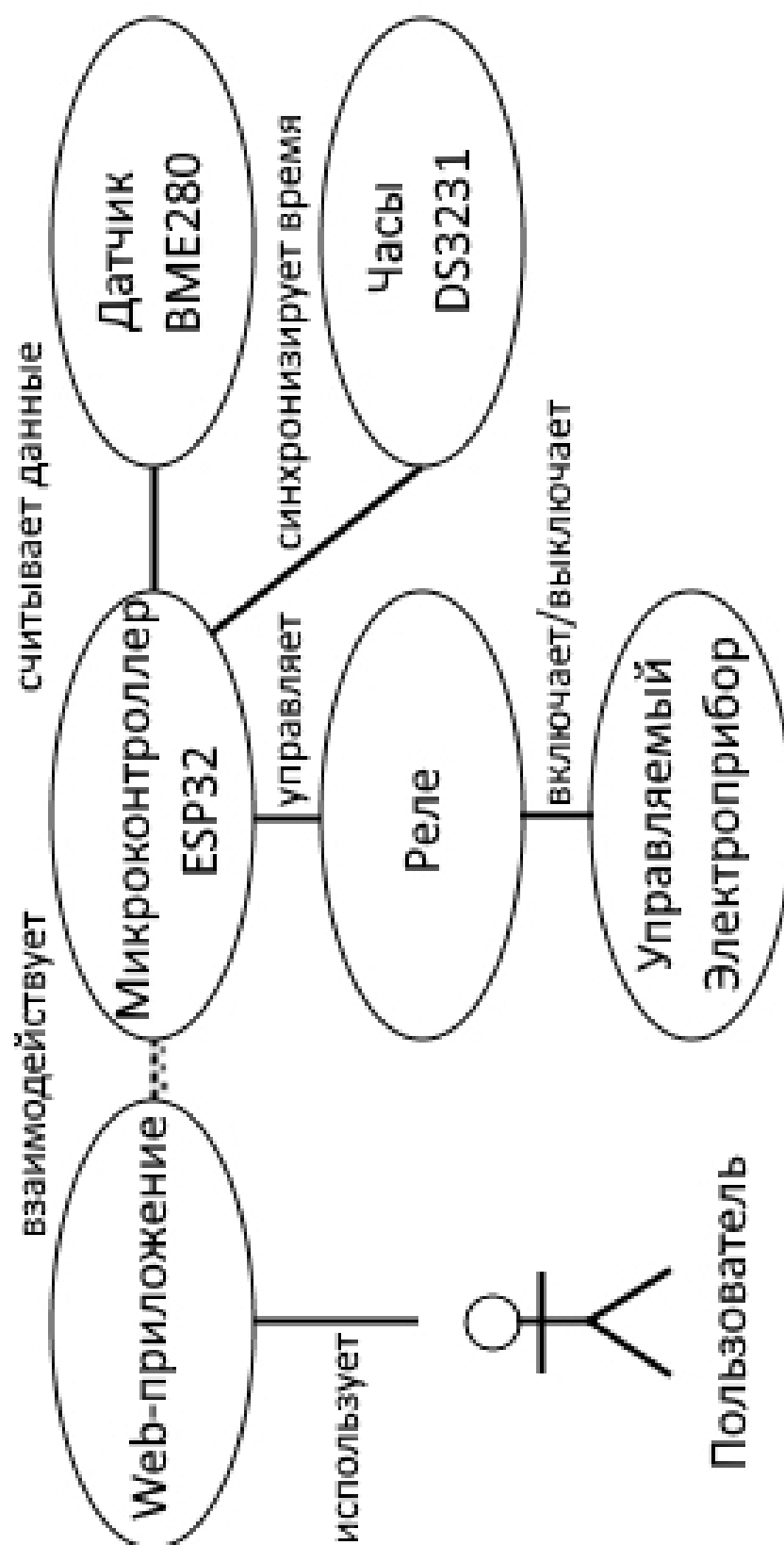
(Обязательное)

Устройство управления электроприборами на основе микроконтроллера

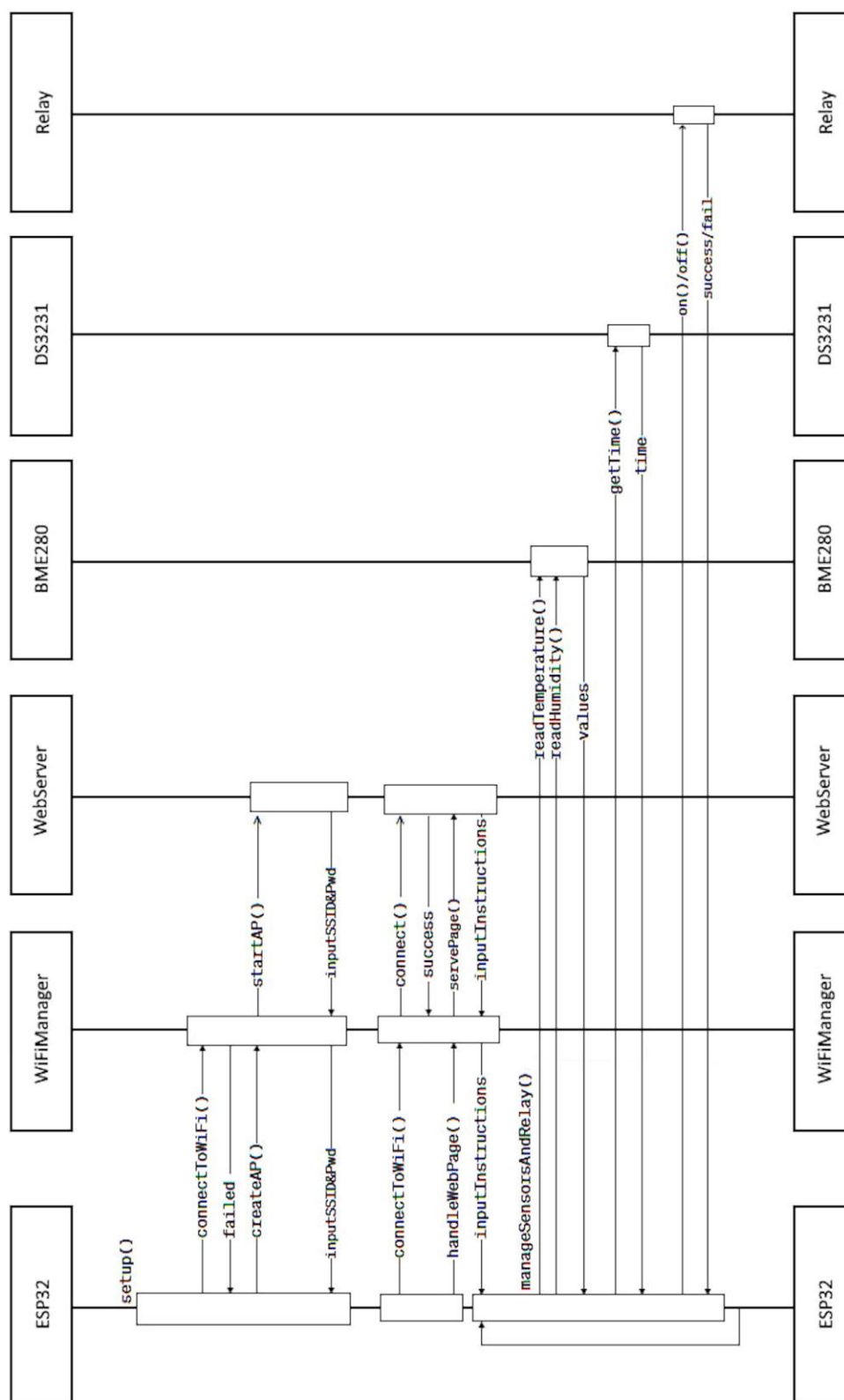
ESP32

UML-диаграммы

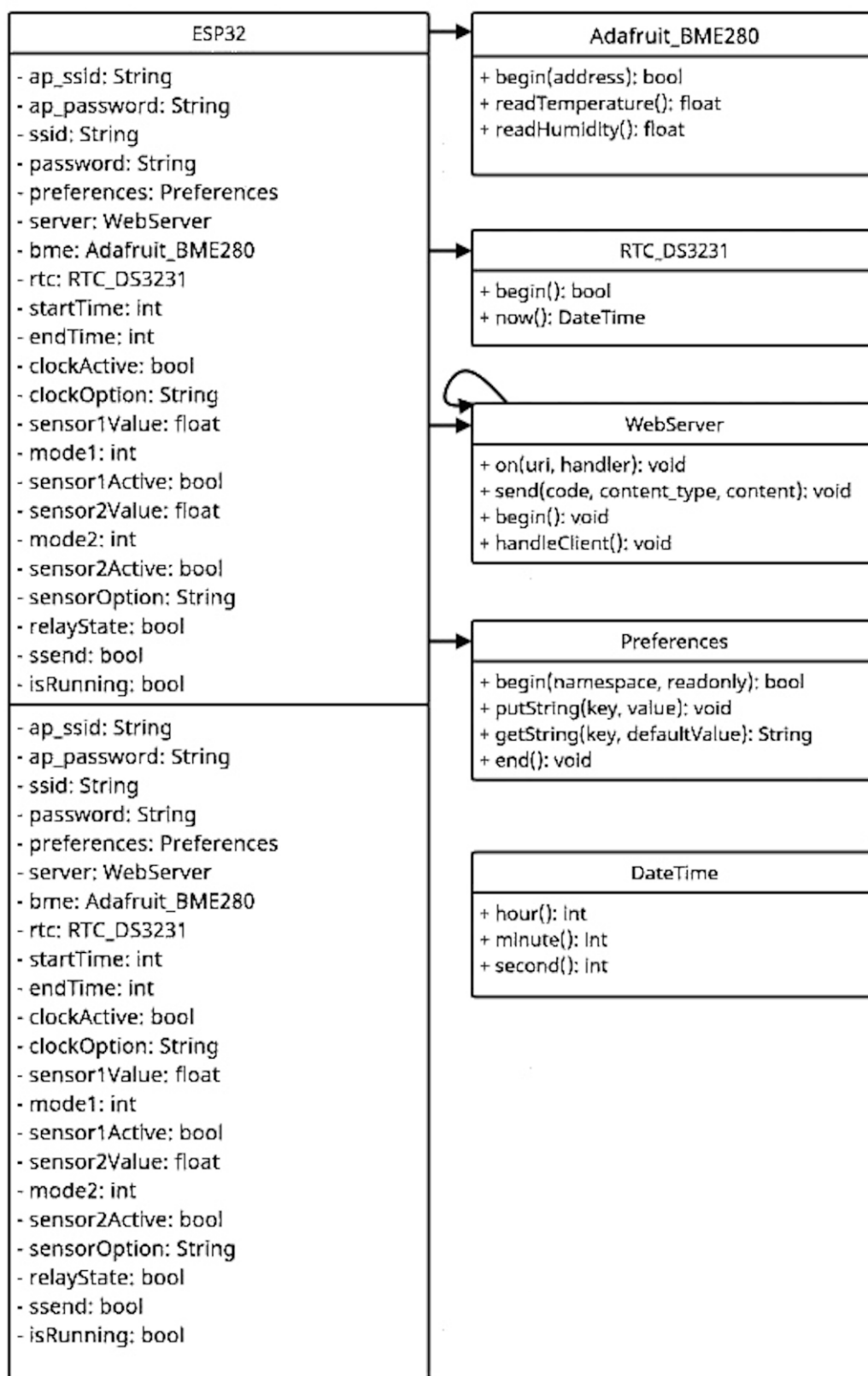
Приложение А.1– Диаграмма прецедентов

[illegible]

Приложение А.2– Диаграмма последовательностей

[illegible]

Приложение А.3– Диаграмма классов



				ПГУ 1.090301.31.004Д11		
Имя	Лист	№ докум.	Подпись	Дата		
Рисов		Халилова А.Р.			Лист	Лист
Писар		Федорова С.М.			Листов	
				Устройство управления электроприводом на основе микроконтроллера (УЭП)		
				Диаграмма классов		
Имя	Лист	№ докум.	Подпись	Дата		
Уткин		Халилова А.С.			ФЭТ, каф.	ИТ
				гр. 20ВВН2		

Приложение Б

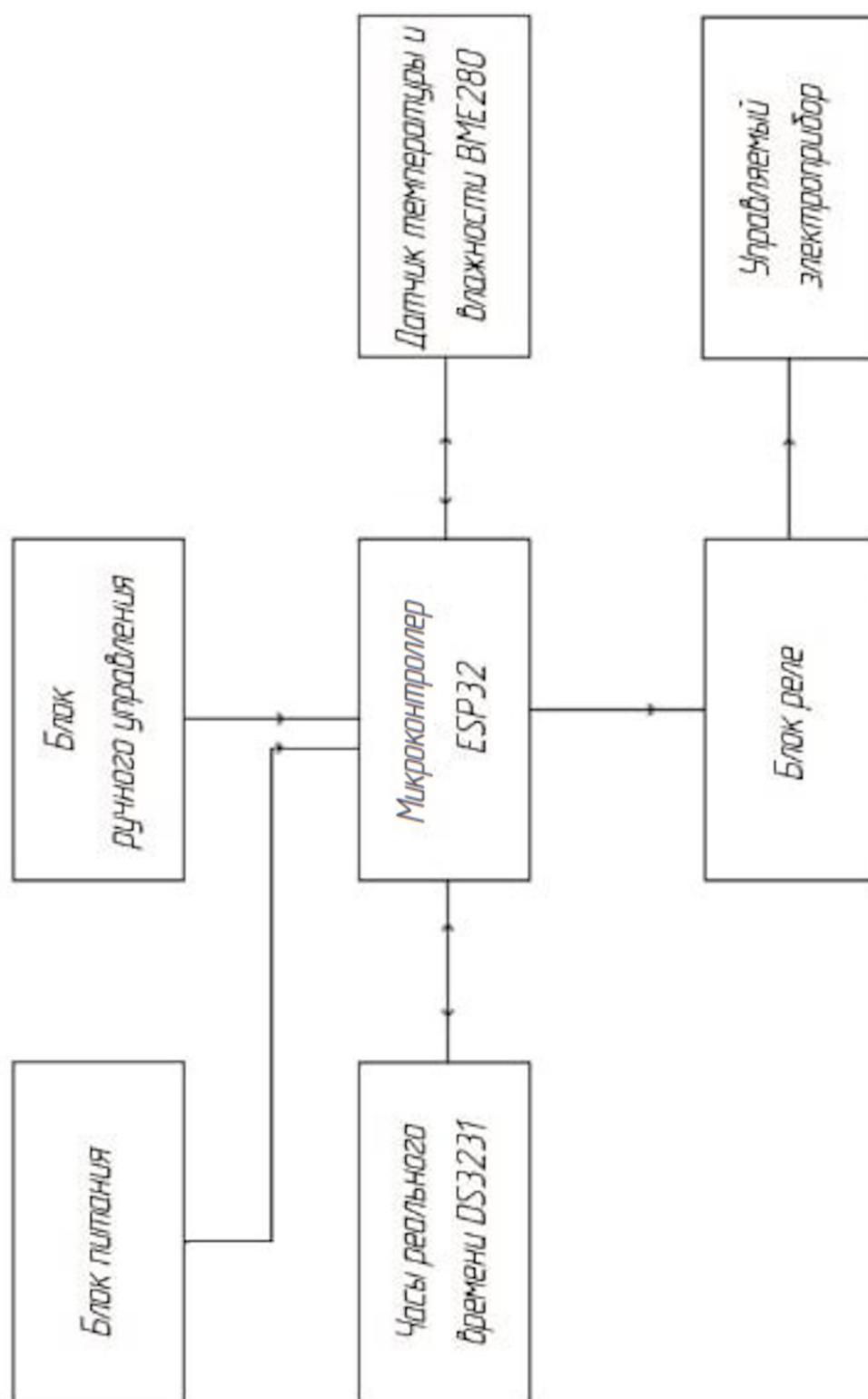
(Обязательное)

Устройство управления электроприборами на основе микроконтроллера

ESP32

Электрические схемы

Приложение Б.1– Структурная схема

[illegible]

Приложение В

(Обязательное)

Устройство управления электроприборами на основе микроконтроллера

ESP32

Листинг программы

Приложение В.1– Листинг программы

Sketch.ino

```
/* includes -----
-----*/
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <RTCLib.h>
#include <WiFi.h>
#include <WebServer.h>
#include <Preferences.h>
#include <ESPmDNS.h>
#include <ArduinoJson.h>
#include <esp_task_wdt.h>
#include "html_form.c"
#include "html_main.c"

/* define -----
-----*/
#define WDT_TIMEOUT 100 // Время сброса мк от WDT, в секундах
#define BME_SDA 21
#define BME_SCL 22
#define RELAY_PIN 23
#define BUTTON_PIN 18

/* Логин и пароль для точки доступа на ESP32 -----
-----*/
const char* ap_ssid = "esp";
const char* ap_password = "12345678";

/* Объекты для работы с: -----
-----*/
WebServer server(80); // веб-сервером
Preferences preferences; // хранением настроек
Adafruit_BME280 bme; // датчиком температуры, атм. давления
и влажности
RTC_DS3231 rtc; // часами реального времени

/* Глобальные переменные -----
-----*/
bool clockActive;
bool sensor1Active;
bool sensor2Active;
bool relayState = false;
bool ssend = false;
bool isRunning = false;
int startTime;
int endTime;
int model;
```

```

int mode2;
float sensor1Value;
float sensor2Value;
String clockOption;
String sensorOption;

/* Обновление состояния реле -----
-----*/

void updateRelay(bool condition) // обновление состояния реле
{
    if (condition)
    {
        digitalWrite(RELAY_PIN, LOW);
        relayState = true;
    }
    else
    {
        digitalWrite(RELAY_PIN, HIGH);
        relayState = false;
    }
}

/* Обработка логики для обновления состояния реле -----
-----*/
void updateRelayState() // Обработка логики для обновления
состояния реле
{
    float temperature = bme.readTemperature();
    float humidity = bme.readHumidity();
    bool clockCondition = checkClockCondition();
    bool sensor1Condition = checkSensorCondition(temperature,
    sensor1Value, model1, sensor1Active);
    bool sensor2Condition = checkSensorCondition(humidity,
    sensor2Value, mode2, sensor2Active);
    bool overallCondition =
    calculateOverallCondition(clockCondition, sensor1Condition,
    sensor2Condition);
    if (ssend)
    {
        updateRelay(overallCondition);
    }
    else
    {
        updateRelay(isRunning);
    }
}

/* Просчет логики автоматического режима -----
-----*/

```

```

bool calculateOverallCondition(bool clockCondition, bool
sensor1Condition, bool sensor2Condition) // Просчет логики
автоматического режима
{
    if (sensorOption == "and")
    {
        if (clockOption == "and")
        {
            return clockCondition && sensor1Condition &&
sensor2Condition;
        }
        else
        {
            return clockCondition || (sensor1Condition &&
sensor2Condition);
        }
    }
    else
    {
        if (clockOption == "and")
        {
            return clockCondition && (sensor1Condition ||
sensor2Condition);
        }
        else
        {
            return clockCondition || sensor1Condition ||
sensor2Condition;
        }
    }
}

```

```

void setup() //VOID SETUP VOID SETUP VOID SETUP VOID SETUP VOID
SETUP VOID SETUP
{
    esp_task_wdt_init(WDT_TIMEOUT, true);
    Serial.begin(115200);
    digitalWrite(RELAY_PIN, HIGH);
    while (!Serial);
    Wire.begin(BME_SDA, BME_SCL);
    if (!bme.begin(0x76)) // Подключение к BME280
    {
        Serial.println("Could not find BME280 sensor, check wiring!");
    }
    rtc.begin(); // Инициализация I2C для DS3231
    pinMode(RELAY_PIN, OUTPUT); // Подключение к реле и кнопке
    pinMode(BUTTON_PIN, INPUT_PULLUP);
    preferences.begin("wifi", false);
    String ssid = preferences.getString("ssid", "");
    String password = preferences.getString("password", "");
}

```

```

if (ssid.length() > 0 && password.length() > 0 &&
digitalRead(BUTTON_PIN) != LOW)
{
    WiFi.begin(ssid.c_str(), password.c_str());
    Serial.print("Connecting to WiFi");
    int counter = 0;
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
        counter++;
        if (counter > 20) {
            Serial.println("\nFailed to connect to WiFi. Starting AP
mode...");
            startAPMode();
            return;
        }
    }
    Serial.println("\nConnected to WiFi");
    server.on("/", handleRoot);
    server.on("/send_data", handleSendData);
    server.begin();
    Serial.println("HTTP server started");
    Serial.print("Please connect to: http://");
    Serial.print(WiFi.localIP());
    Serial.println("/");
    saveIPAddress(WiFi.localIP().toString()); // Сохранение IP-
адреса после успешного подключения
}
else
{
    delay(1500);
    startAPMode();
}

server.on("/getData", HTTP_GET, [](){
    // Обработка запроса на получение данных
    DateTime now = rtc.now();
    int hours = now.hour();
    int minutes = now.minute();
    int TimeInSec = hours * 3600 + minutes * 60; // часы
    float temperature = bme.readTemperature(); //
температура
    float humidity = bme.readHumidity(); // влажность
    String output;

    // Формирование JSON объекта с данными
    StaticJsonDocument<200> doc;
    doc["temperature"] = temperature;
    doc["humidity"] = humidity;
    doc["time"] = String(hours) + ":" + String(minutes);
    doc["auto-mode"] = ssend;

```

```

        doc["relay-state"] = relayState;
        serializeJson(doc, output);
        server.send(200, "application/json", output); //
Отправка данных в формате JSON
    });
    server.on("/set_time", HTTP_POST, handleSetTime);
}

/* Основной цикл работы микроконтроллера -----
-----*/
void loop() //VOID LOOP
{
    esp_task_wdt_reset();
    server.handleClient();
    updateRelayState();
    if (digitalRead(BUTTON_PIN) == LOW) // Управление реле от кнопки
    {
        ssend = false;
        isRunning = !isRunning;
        delay(200);
    }
    delay(10);
}

```

Handler_Web_Server.ino

```
/* Обработчики данных веб-сервера: -----
-----*/

void handleRoot() // Отображение основной html страницы
{
    server.send(200, "text/html", html_page);
}

void handleSendData() //Получение данных со страницы (HTTP)
{
    startTime = server.arg("startTime").toInt();
    endTime = server.arg("endTime").toInt();
    clockActive = server.arg("clockActive").toInt();
    clockOption = server.arg("clockOption");
    sensor1Value = server.arg("sensor1Value").toFloat();
    model = server.arg("model").toInt();
    sensor1Active = server.arg("sensor1Active").toInt();
    sensor2Value = server.arg("sensor2Value").toFloat();
    mode2 = server.arg("mode2").toInt();
    sensor2Active = server.arg("sensor2Active").toInt();
    sensorOption = server.arg("sensorOption");
    ssend = server.arg("ssend").toInt();
    isRunning = server.arg("isRunning").toInt();
    float temperature = bme.readTemperature();
    float humidity = bme.readHumidity();
    server.send(200, "text/plain", "Данные успешно получены сервером");
    updateRelayState(); // Обновление состояния реле после получения данных
}

void handleConfig() // Отображение html страницы настройки сети
{
    server.send(200, "text/html", html_form);
}

void handleSave() // Обработка запроса сохранения настроек сети
{
    String ssid = server.arg("ssid");
    String password = server.arg("password");
    preferences.begin("wifi", false);
    preferences.putString("ssid", ssid);
    preferences.putString("password", password);
    preferences.end();
    server.send(200, "text/html", "<html><body><h1>Settings Saved. Please reboot the device.</h1></body></html>");
}

bool checkClockCondition() //проверка логики для ЧАСОВ
```

```

{
    if (!clockActive)
    {
        return true;
    }
    DateTime now = rtc.now();
    int hours = now.hour();
    int minutes = now.minute();
    int seconds = now.second();
    int TimeInSec = hours * 3600 + minutes * 60 + seconds;
    Serial.print(" \nTime: ");
    Serial.print(TimeInSec);
    return TimeInSec >= startTime && TimeInSec <= endTime;
}

```

bool checkSensorCondition(float sensorValue, float thresholdValue, bool mode, bool active) //проверка логики для ДАТЧИКОВ

```

{
    if (!active)
    {
        return true;
    }
    if (mode)
    {
        return sensorValue > thresholdValue;
    }
    else
    {
        return sensorValue < thresholdValue;
    }
}

```

void startAPMode() // Запуск web-страницы для ввода настроек сети

```

{
    WiFi.softAP(ap_ssid, ap_password);
    IPAddress IP = WiFi.softAPIP();
    Serial.print("AP IP address: ");
    Serial.println(IP);

    String html_response = html_form;
    String savedIP = loadIPAddress(); // Загрузка сохранённого
    IP-адреса из памяти
    html_response.replace("$SAVED_IP$", savedIP); // Замена
    $SAVED_IP$ в HTML на сохранённый IP-адрес

    server.on("/", [html_response]() { // Обработка корневого
    запроса
        server.send(200, "text/html", html_response);
    });
}

```

```

        server.on("/save", handleSave); // Обработка запроса
сохранения настроек сети
        server.begin();

        Serial.println("HTTP сервер запущен в режиме настройки.
Адрес:");
        Serial.println("http://");
        Serial.print(WiFi.localIP());

        if (!MDNS.begin("esp32")) { // заменить "esp32" на желаемое
ИМЯ
            Serial.println("Ошибка создания mDNS!");
        } else {
            Serial.println("\nmDNS esp32 успешно создан");
            Serial.println("/");
        }
    }

void handleSetTime() // Установка нового времени
{
    if (server.hasArg("newTime"))
    {
        String newTimeStr = server.arg("newTime");
        int separatorIndex1 = newTimeStr.indexOf(':');
        int separatorIndex2 = newTimeStr.lastIndexOf(':');
        if (separatorIndex1 != -1 && separatorIndex2 != -1)
        {
            int newHour = newTimeStr.substring(0,
separatorIndex1).toInt();
            int newMinute = newTimeStr.substring(separatorIndex1 +
1, separatorIndex2).toInt();
            int newSecond = newTimeStr.substring(separatorIndex2 +
1).toInt();
            rtc.adjust(DateTime(2024, 5, 16, newHour, newMinute,
newSecond)); // Установка нового времени
            server.send(200, "text/plain", "Успешное изменение
времени"); // Отправка подтверждения
        }
        else
        {
            server.send(400, "text/plain", "Неверный формат
времени"); // Отправка ошибки при неверном формате времени
        }
    }
    else
    {
        server.send(400, "text/plain", "Ошибка отсутствия:
newTime"); // Отправка ошибки при отсутствии параметра newTime
    }
}

```

```

void saveIPAddress(String ipAddress) {
    preferences.begin("network", false); // Начало работы с
хранилищем настроек с именем "network"
    preferences.putString("localIP", ipAddress); // Сохранение
IP-адреса как строки в хранилище
    preferences.end(); // Завершение работы с хранилищем
}

String loadIPAddress() {
    preferences.begin("network", true); // Начало работы с
хранилищем настроек с именем "network", с возможностью создания,
если не существует
    String ipAddress = preferences.getString("localIP",
"0.0.0.0"); // Загрузка IP-адреса из хранилища в переменную
ipAddress
    preferences.end(); // Завершение работы с хранилищем
    return ipAddress; // Возвращение загруженного IP-адреса
}

void handleSave() // Обработка запроса сохранения настроек сети
{
    String ssid = server.arg("ssid");
    String password = server.arg("password");
    preferences.begin("wifi", false);
    preferences.putString("ssid", ssid);
    preferences.putString("password", password);
    preferences.end();
    server.send(200, "text/html", "<html><body><h1>Данные
успешно получены сервером. Необходима
перезагрузка.</h1></body></html>");
}

```

Html_form.c

```

/* HTML страница для ввода логина и пароля к роутеру WiFi -----
-----*/

```

```
const char* html_form = R"rawliteral(
<!DOCTYPE HTML>

<html>

<head>

    <title>WiFi Config</title>

    <style>

        body {

            background: linear-gradient(135deg, #fabad9, hsl(59,
100%, 71%));

            background-size: cover;

            height: 125vh;

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

        }

        .container {

            max-width: 600px;

            margin: 20px auto;

            background-color: #fff;

            padding: 20px;

            border-radius: 10px;

            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

        }

        h1 {

            color: #ff66b3;

            text-align: center;

        }

    )rawliteral;
```

```
form {  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
}  
  
label {  
    font-weight: bold;  
    margin-top: 10px;  
    display: block;  
    width: 100%;  
}  
  
input[type="text"] {  
    width: calc(100% - 20px);  
    padding: 10px;  
    margin: 10px 0;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
}  
  
input[type="submit"] {  
    background-color: coral;  
    color: white;  
    padding: 12px 24px;  
    border: none;  
    border-radius: 20px;  
    cursor: pointer;  
    font-size: 18px;  
    margin-top: 20px;  
}
```

```

        input[type="submit"]:hover {
            background-color: rgb(255, 178, 106);
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Enter WiFi Credentials</h1>
        <form action="/save" method="get">
            <label>Last connect IP:</label>
            <span>${SAVED_IP}</span><br>
            <label>SSID:</label>
            <input type="text" name="ssid" required>
            <label>Password:</label>
            <input type="text" name="password" required>
            <input type="submit" value="Save">
        </form>
    </div>
</body>
</html>
)rawliteral";

```

Html_form.c

```

const char* html_page = R"rawliteral(
<!DOCTYPE html>
<html lang="en">

```

```
<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-
scale=1.0">

  <title>ESP32 Project</title>

<style>

  body {

    background: linear-gradient(135deg, #fabad9, hsl(59,
100%, 71%));

    background-size: cover;

    height: 165vh;

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

  }

  .container {

    max-width: 600px;

    margin: 20px auto;

    background-color: #fff;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

  }

  h2 {

    color: #ff66b3;

    text-align: left;

  }
```

```
#temperature,  
#humidity,  
#time {  
    font-size: 18px;  
    color: #333;  
    font-weight: bold;  
}  
  
input[type="range"] {  
    width: 45%;  
    display: inline-block;  
    vertical-align: middle;  
}  
  
input[type="checkbox"] {  
    margin-right: 10px;  
}  
  
input[type="radio"] {  
    margin-right: 10px;  
}  
  
input[type="number"],  
select,  
input[type="text"] {  
    width: calc(50% - 35%);  
    padding: 8px;
```

```
border-radius: 5px;

border: 1px solid #ccc;

margin-bottom: 10px;
}
```

```
#setTimeButton,
#loadValuesButton {

background-color: coral;

color: white;

padding: 12px 24px;

border: none;

border-radius: 20px;

cursor: pointer;

font-size: 18px;
}
```

```
#setTimeButton: hover,
#loadValuesButton: hover {

background-color: rgb(255, 178, 106);
}
```

```
#startStopButton {

background-color: red;

color: white;

padding: 12px 24px;

border: none;

border-radius: 20px;

cursor: pointer;
}
```

```
        font-size: 18px;
    }

    #startStopButton.stopped {
        background-color: green;
    }

    .indicator {
        width: 15px;
        height: 15px;
        border-radius: 50%;
        display: inline-block;
        margin-left: 10px;
    }

    .green {
        background-color: green;
    }

    .red {
        background-color: red;
    }
</style>
</head>
<body>
    <div class="container">
        <div id="data">
            <h2>Текущие данные:</h2>
            <p>Температура: <span id="temperature">%s</span></p>
            <p>Влажность: <span id="humidity">%s</span></p>
            <p>Время: <span id="time">%02d:%02d</span></p>
```

<p>Автоматический режим: </p>

<p>Состояние реле: </p>

<label for="newTime">Новое время:</label>

<input type="text" id="newTime" name="newTime" placeholder="00:00:00">

<button id="setTimeButton">Задать новое время</button>

</div>

<h2>Условия времени</h2>

<input type="range" id="startTimeRange" min="0" max="1440" value="0">

<input type="range" id="endTimeRange" min="0" max="1440" value="1440">

<label for="startTime">Начало времени:</label>

<input type="text" id="startTime" name="startTime" placeholder="00:00">

<label for="endTime">Конец времени:</label>

<input type="text" id="endTime" name="endTime" placeholder="23:59">

<label for="checkbox1"><input type="checkbox" id="checkbox1" name="clockActive"> Время</label>

<h2>Работа часов в паре с датчиками</h2>

<label for="toggle1"><input type="radio" id="toggle1" name="clock_option" value="and"> И</label>

<label for="toggle2"><input type="radio" id="toggle2" name="clock_option" value="or" checked> ИЛИ</label>


```
<h2>Датчик 1 "Термометр"</h2>

<label for="sensor1Value">Значение:</label>

<input type="number" id="sensor1Value"
name="sensor1Value" value="0">

<label for="model">Режим:</label>

<select id="model" name="model">

    <option value="1">"&gt;" Срабатывание при значении
    &gt;</option>

    <option value="0">"&lt;" Срабатывание при значении
    &lt;</option>

</select>

<br>

<input type="checkbox" id="sensor1Checkbox"
name="sensor1Active">

<label for="sensor1Checkbox">Активировать</label>

<br><br>

<h2>Датчик 2 "Гигрометр"</h2>

<label for="sensor2Value">Значение:</label>

<input type="number" id="sensor2Value"
name="sensor2Value" value="0">

<label for="mode2">Режим:</label>

<select id="mode2" name="mode2">

    <option value="1">"&gt;" Срабатывание при значении
    &gt;</option>

    <option value="0">"&lt;" Срабатывание при значении
    &lt;</option>

</select>

<br>

<input type="checkbox" id="sensor2Checkbox"
name="sensor2Active">

<label for="sensor2Checkbox">Активировать</label>

<br><br>
```

```
<h2>Работа датчиков в паре</h2>

<label for="toggle3"><input type="radio" id="toggle3"
name="sensor_option" value="and"> И</label>

<br>

<label for="toggle4"><input type="radio" id="toggle4"
name="sensor_option" value="or" checked> ИЛИ</label>

<br><br>

<h2>АВТОРЕЖИМ</h2>

<button id="loadValuesButton">Загрузить
значения</button>

<br>

<h2>Ручное управление</h2>

<button id="startStopButton">Запустить</button>

<input type="hidden" id="clockActive" name="clockActive"
value="0">

<input type="hidden" id="sensor1Active"
name="sensor1Active" value="0">

<input type="hidden" id="sensor2Active"
name="sensor2Active" value="0">

</div>

<script>

    // Переменные для сохранения значений

    let startClock = 0;

    let finishClock = 1440;

    let clockActive = false;

    let clockMode = false;

    let dat1 = 0;

    let dat1Mode = false;

    let dat1Active = false;

    let dat2 = 0;

    let dat2Mode = false;
```

```
    let dat2Active = false;

    let datWorkMode = false;

    let isRunning = false; // Переменная для отслеживания
    СОСТОЯНИЯ КНОПКИ

    let ssend = false; // Новая переменная

    const newTimeInput = document.getElementById('newTime');

    const setTimeButton =
document.getElementById('setTimeButton');

    const startTimeRange =
document.getElementById('startTimeRange');

    const endTimeRange =
document.getElementById('endTimeRange');

    const startTimeInput =
document.getElementById('startTime');

    const endTimeInput = document.getElementById('endTime');

    const clockCheckbox =
document.getElementById('checkbox1');

    const clockOptions =
document.querySelectorAll('input[name="clock_option"]');

    const sensor1Value =
document.getElementById('sensor1Value');

    const sensor1Checkbox =
document.getElementById('sensor1Checkbox');

    const sensor2Value =
document.getElementById('sensor2Value');

    const sensor2Checkbox =
document.getElementById('sensor2Checkbox');

    const sensorOptions =
document.querySelectorAll('input[name="sensor_option"]');

    const startStopButton =
document.getElementById('startStopButton');
```

```
const loadValuesButton =
document.getElementById('loadValuesButton');

setInterval(function() {

    var xhr = new XMLHttpRequest();

    xhr.onreadystatechange = function() {

        if (xhr.readyState == 4 && xhr.status == 200) {

            var data = JSON.parse(xhr.responseText);

            // Обновление температуры с обрезкой до 1
знака после запятой

            var temperature =
data.temperature.toFixed(1);

            // Обновление влажности с обрезкой до 1
знака после запятой

            var humidity = data.humidity.toFixed(1);

            // Обновление времени с добавлением нулей
для часов и минут, если они меньше 10

            var hours = ("0" +
data.time.split(":")[0]).slice(-2);

            var minutes = ("0" +
data.time.split(":")[1]).slice(-2);

document.getElementById("temperature").innerHTML = temperature +
" °C";

document.getElementById("humidity").innerHTML = humidity + " %";

            document.getElementById("time").innerHTML =
hours + ":" + minutes;

            var autoMode = data["auto-mode"];

            var autoModeElement = document.getElementById("auto-
mode");
```

```

        if (autoMode == 1) {
            autoModeElement.className = "indicator green";
        } else {
            autoModeElement.className = "indicator red";
        }

        // Обновление состояния реле

        var relayState = data["relay-state"];

        var relayStateElement =
document.getElementById("relay-status");

        if (relayState == 1) {
            relayStateElement.className = "indicator green";
        } else {
            relayStateElement.className = "indicator red";
        }
    }

};

xhr.open("GET", "/getData", true);

xhr.send();

}, 2500); // Обновление данных каждые 2.5 секунды

// Обработчик нажатия кнопки "Задать новое время"
setTimeButton.addEventListener('click', () => {

    const newTime = newTimeInput.value; // Получаем введенное
    новое время

    const formData = new FormData();

    formData.append('newTime', newTime); // Добавляем введенное
    новое время в FormData

```

```

// Отправляем POST запрос на сервер ESP32

fetch('/set_time', {
  method: 'POST',
  body: formData
}).then(response => response.text())
  .then(data => console.log(data))
  .catch(error => console.error(error));
});

/* function updateData(temperature, humidity, time) {
  document.getElementById('temperature').textContent =
temperature;

  document.getElementById('humidity').textContent =
humidity;

  document.getElementById('time').textContent = time;
}*/

// Функция для преобразования времени из минут в секунды
const minutesToSeconds = minutes => minutes * 60;

// Функция для преобразования времени в минуты
const timeToMinutes = time => {
  const [hours, mins] = time.split(':').map(Number);
  return (hours * 60) + mins;
};

// Функция для форматирования времени
const formatTime = minutes => {
  const hours = Math.floor(minutes / 60);

```

```

        const mins = minutes % 60;

        return `${String(hours).padStart(2,
'0')}:${String(mins).padStart(2, '0')}`;

    };

    // Функция для обновления значений переменных

    const updateValues = () => {

        startClock =
minutesToSeconds(parseInt(startTimeRange.value));

        finishClock =
minutesToSeconds(parseInt(endTimeRange.value));

        clockActive = clockCheckbox.checked;

        clockMode =
document.querySelector('input[name="clock_option"]:checked').value === "and";

        dat1 = parseFloat(sensor1Value.value);

        dat1Mode =
parseInt(document.querySelector('select[id="mode1"]').value);

        dat1Active = sensor1Checkbox.checked;

        dat2 = parseFloat(sensor2Value.value);

        dat2Mode =
parseInt(document.querySelector('select[id="mode2"]').value);

        dat2Active = sensor2Checkbox.checked;

        datWorkMode =
document.querySelector('input[name="sensor_option"]:checked').value === "and";

        // Обновление скрытых полей

        document.getElementById("clockActive").value =
clockActive ? "1" : "0";

        document.getElementById("sensor1Active").value =
dat1Active ? "1" : "0";

        document.getElementById("sensor2Active").value =
dat2Active ? "1" : "0";

```

```

    });

    // Обработчик изменения диапазона времени

    startTimeRange.addEventListener('input', () => {

        const startTime =
formatTime(parseInt(startTimeRange.value));

        startTimeInput.value = startTime;

        updateValues(); // Обновление значений при изменении
ползунка

    });

    endTimeRange.addEventListener('input', () => {

        const endTime =
formatTime(parseInt(endTimeRange.value));

        endTimeInput.value = endTime;

        updateValues(); // Обновление значений при изменении
ползунка

    });

    // Обработчики для текстовых полей времени

    startTimeInput.addEventListener('input', () => {

        const startTimeMinutes =
timeToMinutes(startTimeInput.value);

        startTimeRange.value = startTimeMinutes;

        updateValues(); // Обновление значений при изменении
текстового поля

    });

    endTimeInput.addEventListener('input', () => {

        const endTimeMinutes =
timeToMinutes(endTimeInput.value);

```

```
        endTimeRange.value = endTimeMinutes;

        updateValues(); // Обновление значений при изменении
текстового поля
    });

    // Обработчик изменений чекбокса "Время"
    clockCheckbox.addEventListener('change', updateValues);

    // Обработчик изменений радио кнопок "Работа часов"
    clockOptions.forEach(option => {
        option.addEventListener('change', updateValues);
    });

    // Обработчик изменений значений и состояний датчиков
    [sensor1Value, sensor1Checkbox, sensor2Value,
    sensor2Checkbox].forEach(element => {
        element.addEventListener('input', updateValues);
    });

    sensorOptions.forEach(option => {
        option.addEventListener('change', updateValues);
    });

    // Инициализация значений времени при загрузке страницы
    startTimeInput.value =
formatTime(parseInt(startTimeRange.value));

    endTimeInput.value =
formatTime(parseInt(endTimeRange.value));

    // Обработчик нажатия кнопки "Загрузить значения"
```

```

loadValuesButton.addEventListener('click', () => {

    updateValues();

    ssend = true; // При нажатии на "Загрузить значения"
устанавливаем ssend в true

    console.log('startClock:', startClock);
    console.log('finishClock:', finishClock);
    console.log('clockActive:', clockActive);
    console.log('clockMode:', clockMode);
    console.log('dat1:', dat1);
    console.log('dat1Mode:', dat1Mode);
    console.log('dat1Active:', dat1Active);
    console.log('dat2:', dat2);
    console.log('dat2Mode:', dat2Mode);
    console.log('dat2Active:', dat2Active);
    console.log('datWorkMode:', datWorkMode);
    console.log('ssend:', ssend);
    console.log('isRunning:', isRunning);
    const formData = new FormData();
    formData.append('startTime', startClock);
    formData.append('endTime', finishClock);
    formData.append('clockActive', clockActive ? "1" :
"0");
    formData.append('clockOption', clockMode ? "and" :
"or");
    formData.append('sensor1Value', dat1);
    formData.append('mode1', dat1Mode);
    formData.append('sensor1Active', dat1Active ? "1" :
"0");
    formData.append('sensor2Value', dat2);
    formData.append('mode2', dat2Mode);

```

```

        "0");

        formData.append('sensor2Active', dat2Active ? "1" :

        : "or");

        formData.append('ssend', ssend ? "1" : "0");

        formData.append('isRunning', isRunning ? "1" : "0");


        fetch('/send_data', {

            method: 'POST',

            body: formData

        }).then(response => response.text())

            .then(data => console.log(data))

            .catch(error => console.error(error));

    });


    // Обработчик нажатия кнопки "Запустить/Остановить"

    startStopButton.addEventListener('click', () => {

        isRunning = !isRunning; // Переключение состояния

        ssend = false; // При нажатии на

        "Запустить/Остановить" устанавливаем ssend в false

        startStopButton.textContent = isRunning ?

        "Остановить" : "Запустить"; // Обновление текста кнопки

        startStopButton.classList.toggle('stopped',

        !isRunning); // Добавление/удаление класса в зависимости от

        состояния


        updateValues(); // Обновление значений переменных

        console.log('ssend:', ssend);

        console.log('isRunning:', isRunning);


        const formData = new FormData();

```

```
        formData.append('startTime', startClock);  
        formData.append('endTime', finishClock);  
        formData.append('clockActive', clockActive ? "1" :  
"0");  
        formData.append('clockOption', clockMode ? "and" :  
"or");  
        formData.append('sensor1Value', dat1);  
        formData.append('model', dat1Mode);  
        formData.append('sensor1Active', dat1Active ? "1" :  
"0");  
        formData.append('sensor2Value', dat2);  
        formData.append('mode2', dat2Mode);  
        formData.append('sensor2Active', dat2Active ? "1" :  
"0");
```

```
        : "or");

        formData.append('sensorOption', datWorkMode ? "and"

        formData.append('ssend', ssend ? "1" : "0");

        formData.append('isRunning', isRunning ? "1" : "0");


        fetch('/send_data', {
            method: 'POST',
            body: formData
        }).then(response => response.text())
            .then(data => console.log(data))
            .catch(error => console.error(error));

    });
</script>
</body>
</html>
)rawliteral";
```

Приложение Г

(Обязательное)

Устройство управления электроприборами на основе микроконтроллера ESP32

Презентация

ТЕМА ВКР: «УСТРОЙСТВО УПРАВЛЕНИЯ ЭЛЕКТРОПРИБОРАМИ НА ОСНОВЕ МИКРОКОНТРОЛЛЕРА ESP32»

Автор работы: студент группы 20ВВП2 Халилов А.Р.
Научный руководитель: к. т. н. Францев С. М.

Пенза 2024

Цели и задачи работы

Целью данной работы является разработка устройства
управления электроприборами на основе
микроконтроллера ESP32.

Задачами работы являются:

1. Изучение аналогов и прототипов устройства;
2. Разработка структурной схемы устройства;
3. Разработка программного обеспечения устройства, обеспечивающего:
 - логику взаимодействия устройства с датчиком и часами реального времени;
 - взаимодействие пользователя с устройством через web-интерфейс;
 - управление электроприбором в ручном и автоматическом режимах;
 - контроль состояния электроприбора, показаний датчика влажности и температуры, а также правильности хода часов реального времени.
4. Разработка макетного образца устройства;
5. Тестирование устройства.

Актуальность работы

В связи с увеличением количества различных электроприборов в быту возникает необходимость автоматизации управления ими.

Это позволяет освободить пользователя от необходимости постоянного управления электроприборами вручную.

Достаточное количество электроприборов не обладают функционалом удаленного управления пользователем и автоматическими режимами работы.

Существующие аналоги и прототипы, такие как управляемые розетки, GSM и IoT устройства не обладают функционалом управления по значениям различных параметров окружающей среды, а также обладают высокой ценой.

2

Аналоги



3

Выбор среды и инструментов разработки



HTML



CSS

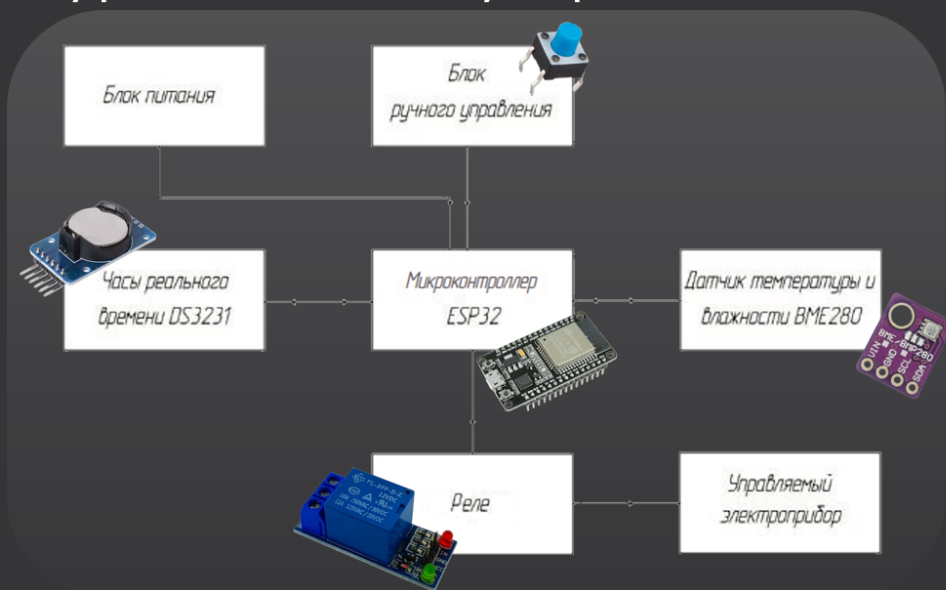


JS



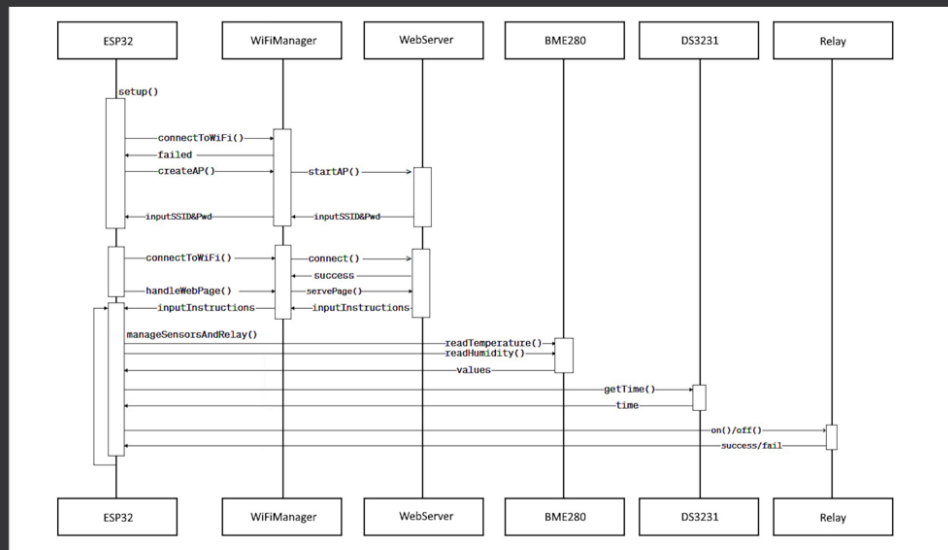
4

Структурная схема устройства



5

UML диаграмма последовательности



6

Внешний вид web-страницы настройки сети

Enter WiFi Credentials

Last connect IP: 192.168.238.18

SSID:

Password:

7

ВНЕШНИЙ ВИД ГЛАВНОЙ СТРАНИЦЫ

Текущие данные:

Температура: 28.2 °C

Влажность: 45.7 %

Время: 13:34

Автоматический режим: ☒

Состояние реле: ☒

Новое время: 00:00:00 Задать новое время

Условия времени

Начало времени: 13:32

Конiec времени: 24:00

☒ Время

Работа часов в паре с датчиками

☒ И

☐ ИЛИ

Датчик 1 "Термометр"

Значение: 0

Режим: °C Сре

☐ Активировать

Датчик 2 "Гигрометр"

Значение: 50

Режим: °C Сре

☒ Активировать

Работа датчиков в паре

☒ И

☐ ИЛИ

АВТОРЕЖИМ

Загрузить значения

Ручное управление

Запустить

Текущие данные:

Температура: 28.2 °C

Влажность: 45.7 %

Время: 13:34

Автоматический режим: ☒

Состояние реле: ☒

Новое время: 00:00:00 Задать новое время

Условия времени

Начало времени: 13:32

Конiec времени: 24:00

☒ Время

Работа часов в паре с датчиками

☒ И

☐ ИЛИ

Датчик 1 "Термометр"

Значение: 0

Режим: °C Сре

☐ Активировать

Датчик 2 "Гигрометр"

Значение: 50

Режим: °C Сре

☒ Активировать

Работа датчиков в паре

☒ И

☐ ИЛИ

АВТОРЕЖИМ

Загрузить значения

Ручное управление

Запустить

8

БЛОК ВЫВОДА КОНТРОЛИРУЕМЫХ ДАННЫХ

Текущие данные:

Температура: 28.2 °C

Влажность: 45.7 %

Время: 13:34

Автоматический режим: ☒

Состояние реле: ☒

Новое время: 00:00:00 Задать новое время

9

Блок настройки таймера

Условия времени



Начало времени: 13:32

Конец времени: 24:00

☒ Время

Работа часов в паре с датчиками

- ☒ И
☐ ИЛИ

10

Блок настройки порогов датчиков

Датчик 1 "Термометр"

Значение: 0

Режим: ">" Сраб

☐ Активировать

">" Срабатывание при значении >
"<" Срабатывание при значении <

Датчик 2 "Гигрометр"

Значение: 50

Режим: ">" Сраб

☒ Активировать

Работа датчиков в паре

- ☒ И
☐ ИЛИ

11

Блок автоматического и ручного управления

АВТОРЕЖИМ

Загрузить значения

Ручное управление

Запустить

12

Заключение

ВКР посвящена разработке устройства для удаленного ручного и автоматического включения/отключения электроприборов посредством электромеханического реле. Устройство реализовано на базе микроконтроллера ESP32, которое получает значения температуры и относительной влажности воздуха с датчика, а также получает данные с часов реального времени. Ручное управление и установка режимов автоматической работы устройства осуществляется через web - интерфейс. Реализованы возможности автоматического включения/отключения электроприбора по времени и по пороговому значению температуры и относительной влажности воздуха. Проведено тестирование работы устройства. Рассмотрены требования техники безопасности при разработке и использовании устройства.

В рамках бакалаврской работы разработано программное обеспечение микроконтроллера, web - интерфейс и схемотехническое решение оригинального устройства, позволяющего удаленно и автоматизировано управлять электроприборами, изначально не обладающим таким функционалом.

Устройство позволяет: отслеживать состояние микроклимата, управлять электроприборами через беспроводное соединение и браузер, автоматизировать работу и улучшить функционал имеющихся электроприборов.



13