

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: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

СИСТЕМА УПРАВЛЕНИЯ ОБЪЕКТАМИ
УМНОГО ДОМА

Студент

(подпись, дата)

Шерстобитов Иван Дмитриевич

14.06.2024 (ФИО полностью)

Руководитель

(подпись, дата)

14.06.2024

К.И. Никишин

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

14.06.2024

А.С. Бычков

(фамилия, инициалы)

Рецензент

21.06.2024
АО «ИПП «Рубин» (должность, место работы)

(подпись, дата)

А.В. Васильев
(фамилия, инициалы)

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

Заведующий кафедрой

(подпись)

М.А. Митрохин

(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

18.06.2024 № 29)

Секретарь ГЭК

(подпись)

Е.И. Гурин

(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

« 13 » 09 2022г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Шерстобитов Иван Дмитриевич гр. 22ВВИм1 факультета ВТ

2. Тема работы Система управления объектами умного дома

Тема утверждена приказом ПГУ № 765/о от " 13 " 09 2022 г.

3. Руководитель магистерской диссертации к.т.н., доцент кафедры Никишин К. И.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

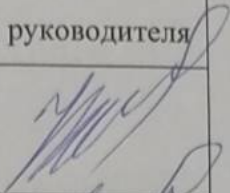
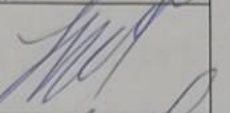
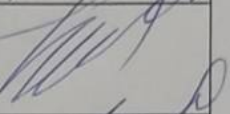
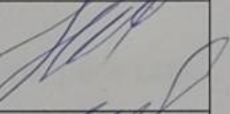
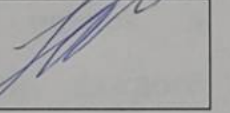
Назначение разработки: разработка системы управления объектами умного дома на базе микроконтроллера CH32V307VCT6, интегрированного с платформой Home Assistant, установленной на одноплатном компьютере Orange Pi 5B. С возможностью интеграции технологий искусственного интеллекта

Разработать архитектуру системы управления объектами умного дома и программное обеспечение для микроконтроллера CH32V307VCT6. Установить Home Assistant на одноплатный компьютер Orange Pi 5B. Соединить устройства через Ethernet и осуществить обмен данными по протоколу MQTT. Интегрировать технологии искусственного интеллекта. Произвести отладку системы.

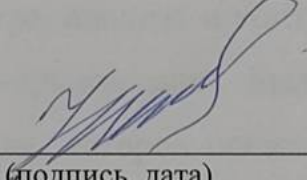
5. Перечень подлежащих разработке вопросов

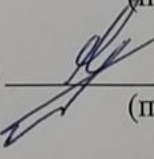
- 1) Обзор и анализ существующих систем умного дома
- 2) Анализ и выбор технологий и протоколов умного дома
- 3) Разработка архитектуры управления объектами умного дома
- 4) Разработки алгоритма взаимодействия периферийных устройств на базе микроконтроллера CH32V307VCT6
- 5) Разработка программного обеспечения микроконтроллера CH32V307VCT6.
- 6) Установка и настройка Home Assistant на Orange Pi 5B
- 7) Настройка MQTT брокера в Home Assistant и подключение устройств
- 8) Интеграция технологий искусственного интеллекта в систему умного дома.

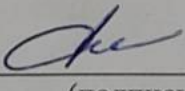
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ существующих систем хранения и обработки информации	15 %	13.09.2022-16.12.2022	
2.	Анализ средств разработки web-приложений	10 %	17.12.2022-18.02.2023	
3.	Анализ существующих СУБД и средств разработки БД	20 %	19.02.2023-31.05.2023	
4.	Анализ способов тестирования клиент-серверных приложений	15 %	1.06.2023-27.08.2023	
5.	Разработка серверной части приложения	15 %	28.08.2023-31.11.2023	
6.	Разработка клиентской части приложения	10 %	01.12.2023-31.01.2024	
7.	Тестирование системы	5 %	01.02.2024-04.04.2024	
8.	Оформление документации	10 %	05.04.24 - 31.05.24	

Дата выдачи задания "13" сентября 2022 г.

Руководитель магистерской диссертации  Слепцов Н.В.
(подпись, дата)

Задание к исполнению принял студент  17.06.2024 Махров А. А.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
(подпись, дата)
17.06.2024

Реферат

Пояснительная записка содержит 129 листов, 2 таблицы, 73 рисунка.

ETHERNET, MQTT, CH32V307VCT6, ORANGE PI 5B,
МИКРОКОНТРОЛЛЕР, ОДНОПЛАТНЫЙ КОМПЬЮТЕР,
ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Объектом исследования является система управления объектами умного дома (СУОУД).

Предметом исследования являются элементы СУОУД микроконтроллер CH32V307VCT6 и одноплатный компьютер Orange Pi 5B

Целью данной работы - создание функциональной и надежной системы умного дома, которая будет учитывать современные тенденции в области IoT и обеспечит интеграцию с технологиями искусственного интеллекта (ИИ).

В результате проделанной работы был проведён анализ существующих решений СУОУД, выбор оптимальных компонентов и методов их интеграции, разработка программного обеспечения для бесперебойного взаимодействия всех элементов СУОУД, интеграция ИИ для улучшения функциональности и адаптивности СУОУД. реализация результатов тестирования СУОУД с применением технологий LLM

Научная новизна состоит в том, что разработан алгоритм взаимодействия периферийных устройств на базе микроконтроллера CH32V307VCT6 на архитектуре RISC-V, что позволяет создать гибкую и стабильную СУОУД. Внедрена модель LLM для дальнейшего обучения и адаптивного управления, что позволяет значительно повысить уровень автоматизации и комфорта в СУОУД

					ПГУ1.090401.63.001 ПЗ				
Изм	Имя	№ докум.	Подпись	Дата	Система управления объектами умного дома Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Шерстобитов И.Д		19.06.24					
Провер.		Никишин К.И.		19.06.24					
Н. Контр		Бычков А.С.		19.06.24			2	128	
Утверд.		Митрохин М.А.		19.06.24		ФВТ, каф. ВТ гр. 22ВВИм1			

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	7
ВВЕДЕНИЕ	8
1 Обзор существующих решений для СУОУД	12
1.1 Концепция умного дома	12
1.2 Технологии и стандарты в области умного дома	13
1.2.1 RS-485	13
1.2.2 Ethernet	14
1.2.1 Протокол MQTT	16
1.2.1 Zigbee	19
1.2.1 Z-Wave	20
1.2.1 Bluetooth	21
1.2.1 Matter	22
1.2.1 Thread	23
1.3 Системы управления объектами умного дома	24
1.3.1 Централизованные СУОУД	24

1.3.2 Децентрализованные СУОУД	25
1.3.3 Смешанная СУОУД	26
1.4 Применение искусственного интеллекта в системах умного дома	26
1.5 Аналитический обзор существующих решений умного дома	29
1.5 Home Assistant	29
1.5 Google Home	30
1.5 Яндекс.Алиса	31
1.5 Сбер Салют	32
1.6 Вывод	34
2 Проектирование систем управления объектами умного дома	35
2.1 Постановка задачи и формулирование целей исследования	35
2.2 Описание используемых технологий и инструментов	36
2.3 Выбор аппаратного обеспечения умного дома	36
2.3.1 Выбор микроконтроллера	36
2.3.2 Выбор устройства для локального сервера умного дома	41
2.4 Установка ПО Home Assistant	45
2.5 Настройка Home Assistance	48

2.5.1 Настройка File Editor	49
2.5.2 Настройка Maria DB	51
2.5.3 Настройка HACS	52
2.6 Разработка алгоритма взаимодействия периферийных устройств на базе микроконтроллера CH32V307VCT6	55
2.7 Вывод	61
3 Реализация системы управления объектами умного дома.	62
3.1 Разработка устройства считывания датчиков и управления блоком реле	62
3.1.1 Разработка структурной схемы.	62
3.1.2 Разработка аппаратной части.	63
3.1.3 Подбор датчиков.	65
3.1.4 Подбор блока электромеханических реле.	65
3.2 Разработка ПО.	66
3.1.4 Разработка функций сценариев.	65
3.3 Вывод	91
4 Тестирование и отладка системы	92

4.1 Отладка ПО на микроконтроллере CH32V307VCT6	92
4.2 Сопряжение устройств с локальным сервером по MQTT	95
4.3 Интеграция искусственного интеллекта в Home Assistant	100
ЗАКЛЮЧЕНИЕ	104
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	105
ПРИЛОЖЕНИЕ А Листинг программы	109
ПРИЛОЖЕНИЕ Б Презентация	120

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

СУОУД– система управления объектами умного дома

ИИ – искусственный интеллект

LLM (large language model) – большая языковая модель

IoT (Internet of Things) – интернет вещи

MQTT (Message Queuing Telemetry Transport) – передача телеметрии в очередь сообщений

QoS (quality of service) – качество обслуживания

ПО – программное обеспечение

МК – микроконтроллер

МО –машинное обучение

ОЕЯ – основы естественного языка

ВВЕДЕНИЕ

В современном мире системы умного дома, представляют собой одно из ключевых направлений современных технологий, стремительно развивающихся и находят широкое применение в повседневной жизни. Интеграция различных устройств и сенсоров в единую управляющую инфраструктуру позволяет не только повысить уровень комфорта и безопасности в жилых помещениях, но и значительно сократить затраты на энергопотребление. Интеграция искусственного интеллекта (ИИ) в такие системы открывает новые возможности для автоматизации, анализа данных и адаптивного управления. В данной магистерской диссертации рассматривается разработка системы управления объектами умного дома на базе платформы Home Assistant с использованием микроконтроллера (МК) CN32V307VCT6 и одноплатного компьютера Orange Pi 5B, с акцентом на интеграцию технологий ИИ.

Актуальность выбранной темы обусловлена растущим интересом к автоматизации и интеллектуальному управлению домашними устройствами. Современные пользователи стремятся создать удобную, функциональную и надежную систему, способную интегрировать различные аспекты домашнего быта, начиная от управления освещением и климатом, заканчивая обеспечением безопасности и мониторингом состояния различных устройств. Применение открытых платформ, таких как Home Assistant, и использование MQTT-протокола для связи между устройствами предоставляет широкие возможности для создания гибких и масштабируемых решений. Применение ИИ позволяет создать интеллектуальные алгоритмы, способные обучаться и адаптироваться к предпочтениям пользователей, что значительно расширяет функциональные возможности системы.

Одной из главных задач данной работы является разработка и внедрение системы, способной обеспечить надежную и эффективную коммуникацию между МК CN32V307VCT6, выполняющим роль сборщика данных с различных сенсоров и управляющего блока для исполнительных

устройств, и центральным узлом на базе Orange Pi 5B, работающим под управлением Home Assistant. В ходе работы будут рассмотрены особенности интеграции различных компонентов системы, включая настройку MQTT-брокера, программирование микроконтроллера для взаимодействия с датчиками и исполнительными устройствами, а также разработку конфигурационных файлов для Home Assistant, обеспечивающих корректную работу всей системы.

Но существует ряд проблем в связи с применением систем управления умным домом:

- проблема безопасности. сложность установки ПО связана с применением прокси-сервера, что не обеспечивает конфиденциальность данных,
- нестабильная работа технологий ИИ в системе управления умным домом. В связи с не достаточной отлаженной работой,
- проблема выхода из строя одноплатного компьютера, на котором установлен Home Assistant.

Предполагается что разработка автономного устройства считывания датчиков и управления блоком реле с применением LLM (large language model), сможет решить данные проблемы, обеспечить гибкость и стабильность работы системы умного дома.

Целью данной работы является создание функциональной и надежной системы умного дома, которая будет учитывать современные тенденции в области IoT (Internet of Things) и обеспечит высокую степень интеграции и управления различными аспектами домашнего хозяйства, а также интеграцию с технологиями ИИ. Основными задачами исследования являются анализ существующих решений, выбор оптимальных компонентов и методов их интеграции, разработка программного обеспечения для обеспечения бесперебойного взаимодействия всех элементов системы, а также интеграция ИИ для улучшения функциональности и адаптивности системы.

Объектом исследования является система управления объектами умного дома (СУОУД).

Предметом исследования являются элементы СУОУД микроконтроллер CH32V307VCT6 и одноплатный компьютер Orange Pi 5B

Результаты данного исследования могут быть полезны не только для создания частных умных домов, но и для разработки масштабируемых решений, применимых в коммерческих и общественных зданиях, что открывает новые перспективы для дальнейших исследований и внедрения инновационных технологий в повседневную жизнь.

Научная новизна данной работы заключается в разработке алгоритма взаимодействия периферийных устройств на базе микроконтроллера CH32V307VCT6 и одноплатного компьютера Orange Pi 5B для управления системой объектами умного дома, основанной на открытой платформе Home Assistant и с акцентом на внедрение искусственного интеллекта. Основные аспекты новизны включают:

- разработка алгоритма взаимодействия периферийных устройств комплексной системы управления: предлагается комплексная система управления, включающая сбор данных с различных сенсоров, управление исполнительными устройствами и интеграцию ИИ для адаптивного управления. Это включает в себя разработку программного обеспечения для микроконтроллера и конфигурацию Home Assistant для обеспечения полной автоматизации и интеграции элементов СУОУД,

- интеграция различных технологий. Разработка системы умного дома, которая сочетает в себе возможности Home Assistant, микроконтроллера CH32V307VCT6 и одноплатного компьютера Orange Pi 5B, демонстрирует уникальный подход к объединению различных технологий и протоколов для создания комплексного решения. В работе рассматриваются вопросы совместимости и эффективного взаимодействия этих компонентов, что является значительным вкладом в область IoT и систем автоматизации,

– использование микроконтроллера CH32V307VCT6. микроконтроллеры архитектуры RISC-V широко применяются в системах умного дома, использование данного МК, который относительно недавно появился на рынке, представляет собой инновационный подход. В работе проводится анализ его возможностей и преимуществ по сравнению с другими популярными МК, а также демонстрируется его практическое применение в реальной системе умного дома,

– оптимизация взаимодействия устройств через MQTT. Работа включает исследование и оптимизацию протокола MQTT для обеспечения надежной и эффективной связи между микроконтроллером и центральным узлом системы. Предлагаются методы повышения устойчивости и производительности системы за счет оптимизации конфигурации и параметров передачи данных,

– интеграция ИИ для адаптивного управления. Одним из ключевых аспектов новизны является интеграция LLM в систему умного дома. Разработка и внедрение интеллектуальных алгоритмов, способных анализировать данные с сенсоров, предсказывать поведение пользователей и адаптировать управление устройствами, что позволяет значительно повысить уровень автоматизации и комфорта в умном доме.

Таким образом, данная работа не только демонстрирует практическое применение современных технологий для создания системы умного дома, но и вносит значительный вклад в область научных исследований, направленных на разработку эффективных и инновационных решений для автоматизации и интеллектуального управления бытовыми устройствами.

1 Обзор существующих решений для СУОУД

1.1 Концепция умного дома

Умный дом — это прежде всего система интеллектуальной автоматики для управления всеми инженерными коммуникациями современного здания. Официальной датой рождения «Умного дома» принято считать 1978 год. Компания из Шотландии Pico Electronics представила первый в мире специальный стандарт по управлению домашними устройствами – X10. Специалисты компании решили в качестве проводника для сигналов управления использовать простую электрическую сеть. Они дополнили свою разработку беспроводным вариантом радиоуправления на частоте 433 МГц. Концепция умного дома представляет собой интеграцию различных электронных устройств и систем в единую управляющую инфраструктуру с целью повышения уровня комфорта, безопасности и энергоэффективности жилища. Основной идеей умного дома является автоматизация повседневных задач и процессов, что достигается за счет использования датчиков, исполнительных устройств и центральных контроллеров. Умные дома позволяют пользователям удаленно управлять бытовыми приборами, освещением, системами отопления и кондиционирования, системами безопасности и другими компонентами через мобильные приложения или голосовые команды [1].

Одной из ключевых тенденций систем умного дома является интеграция технологий искусственного интеллекта. AI позволяет улучшить автоматизацию и управление устройствами, а также обеспечить более персонализированный опыт для пользователей. AI-ассистенты, такие как Google Assistant, Amazon Alexa и Apple Siri, играют важную роль в этой интеграции [2].

Развитие голосового управления становится все более популярным способом взаимодействия с умными домами. Платформы, такие как Google Home и Amazon Echo, позволяют пользователям управлять своими устройствами с помощью голосовых команд, что делает использование умного дома более удобным и интуитивно понятным.

Повышение уровня безопасности, является одним из ключевых аспектов умного дома. Производители уделяют все больше внимания разработке надежных и безопасных систем, которые защищают данные пользователей и предотвращают несанкционированный доступ к устройствам.

Интеграция с облачными сервисами играют важную роль в развитии умных домов. Они позволяют хранить данные, управлять устройствами удаленно и обеспечивать обновление программного обеспечения.

1.2 Технологии и стандарты в области умного дома

Современные технологии и стандарты, применяемые в системах умного дома, включают в себя разнообразные беспроводные и проводные протоколы передачи данных, каждый из которых имеет свои преимущества и области применения.

1.2.1 Rs-485

RS-485 — это стандарт для передачи данных по последовательной шине, который широко используется в промышленных и коммерческих системах автоматизации, а также в системах сбора данных и управления. Он обладает рядом преимуществ по сравнению с другими интерфейсами, такими как RS-232. Благодаря своей высокой устойчивости к помехам и способности поддерживать многоточечные соединения и возможность передачи данных на большие расстояния. RS-485 продолжает оставаться популярным выбором для многих применений [3]. Тип подключения RS-485 представлен на рисунке 1.1.

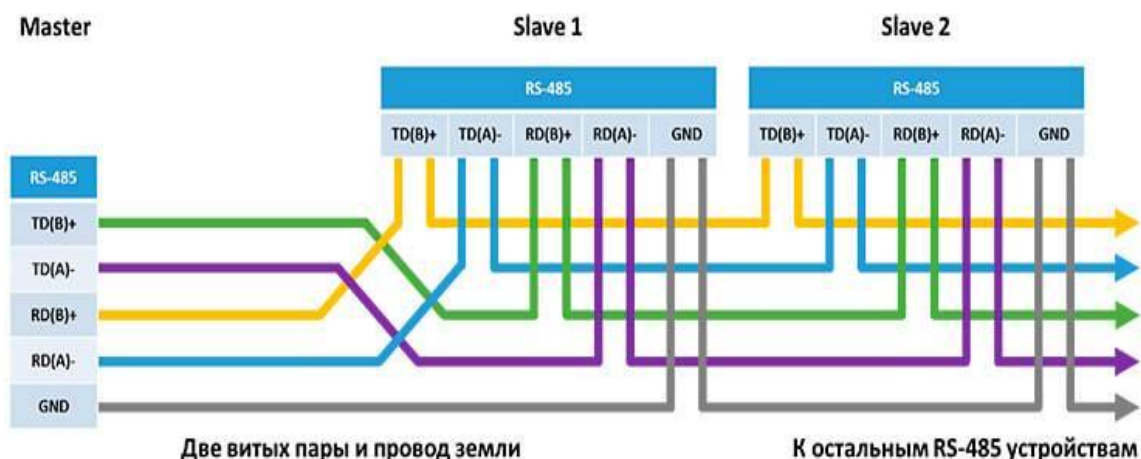


Рисунок 1.1 – Тип подключения RS-485

Преимущества:

- помехоустойчивость, дифференциальная передача данных снижает влияние электромагнитных помех,
- дальность, возможность передачи данных на расстояния до 1200 метров,
- многоточечные соединения, поддержка нескольких устройств на одной шине.
- гибкость и надежность, широкое применение в промышленности и автоматизации.

Недостатки:

- полудуплексный режим, одновременная передача и прием данных не возможны (в стандартной конфигурации),
- сложность в конфигурировании, требуется правильная настройка терминаторов и управление доступом к шине,
- ограниченное количество устройств, без дополнительных драйверов поддерживается до 32 устройств.

Применение RS-485:

- промышленная автоматизация. RS-485 широко используется для связи между контроллерами, датчиками и исполнительными устройствами в системах автоматизации и управления,
- системы сбора данных, интерфейс применяется в системах мониторинга и сбора данных, таких как SCADA, BMS (Building Management Systems) и системы учета энергоресурсов,
- транспортные и энергетические системы RS-485 используется в системах управления транспортом, энергосистемах и в сетях умных счетчиков.
- коммерческие и бытовые приложения,
- интерфейс находит применение в системах видеонаблюдения, охранных системах и системах управления освещением.

1.2.2 Ethernet

Ethernet — это технология локальных сетей Local Area Network (LAN), которая определяет стандарты физического и канального уровней для передачи данных. Ethernet является одной из наиболее распространенных технологий, используемых для соединения компьютеров, сетевых устройств и других устройств в локальной сети. Это надежная и высокоскоростная технология. Она обеспечивает высокую пропускную способность, масштабируемость и совместимость, что делает её идеальной для использования в корпоративных, домашних, промышленных и IoT-сетях. Благодаря развитию стандартов и технологий, Ethernet продолжает оставаться ключевой технологией для сетевого взаимодействия в различных областях. [4]. Графическая схема взаимодействия устройств представлена на рисунке 1.2.



Рисунок 1.2 – Графическая схема взаимодействия Ethernet

Преимущества:

- высокая скорость передачи данных Ethernet поддерживает высокоскоростные соединения до 100 Гбит/с и выше,
- широкая совместимость Ethernet-устройства совместимы между собой, что облегчает развертывание и расширение сети,
- надежность Ethernet обеспечивает стабильную и надежную передачу данных с низким уровнем ошибок.

- масштабируемость легко масштабируется от небольших локальных сетей до крупных корпоративных сетей.

Недостатки:

- ограниченная мобильность. в отличие от беспроводных сетей, Ethernet требует проводного соединения, что ограничивает мобильность устройств,
- затраты на кабели и оборудование. Развертывание проводной сети может быть дорогостоящим из-за необходимости прокладки кабелей и покупки сетевого оборудования.

1.2.3 Протокол MQTT

MQTT (Message Queuing Telemetry Transport) — это легковесный протокол передачи сообщений, разработанный для надёжного соединения обмена данными между устройствами с ограниченными вычислительными ресурсами и низкой пропускной способностью сети. Это мощный и гибкий протокол, идеально подходящий для применения в системах Интернета вещей (IoT) для передачи данных между датчиками, контроллерами и облачными сервисами. Благодаря своей легковесности и простоте использования, MQTT широко применяется в различных проектах и приложениях, связанных с удаленным мониторингом, управлением устройствами и автоматизацией [5]. Графическая схема работы MQTT представлена на рисунке 1.3.

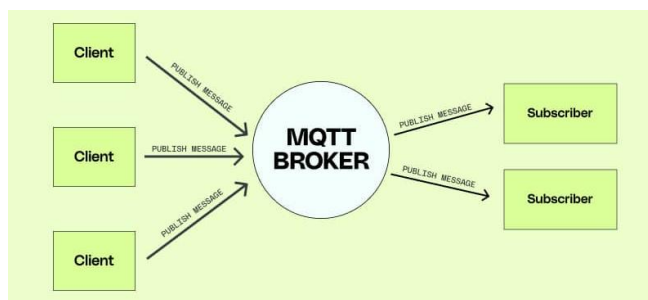


Рисунок 1.3 – Графическая схема работы MQTT

Основные характеристики MQTT

- легковесность. Протокол имеет небольшой заголовок, что делает его подходящим для использования в устройствах с ограниченными ресурсами и в сетях с низкой пропускной способностью,

- MQTT поддерживает асинхронную передачу сообщений, что позволяет устройствам обмениваться данными без необходимости постоянного подключения,

- архитектура "издатель-подписчик". Сообщения передаются через центральный сервер (брокер). Устройства могут выступать в роли издателей (публиковать сообщения) и подписчиков (получать сообщения).

Качество обслуживания (QoS):

- MQTT поддерживает три уровня качества обслуживания;

- 1)QoS 0: Сообщение доставляется не более одного раза (доставка не гарантирована);

- 2)QoS 1: Сообщение доставляется как минимум один раз (доставка гарантирована, но могут быть дубли);

- 3)QoS 2: Сообщение доставляется точно один раз (доставка гарантирована без дубликатов);

- управление соединением. Устройства могут подключаться и отключаться от брокера по мере необходимости, что позволяет экономить ресурсы и батарею;

- управление состоянием (Last Will and Testament) MQTT поддерживает механизм оповещения подписчиков о неожиданных отключениях издателей.

Основные компоненты MQTT

- брокер. Центральный сервер, который получает сообщения от издателей и отправляет их подписчикам,

- издатель (Publisher). Устройство или приложение, которое отправляет сообщения на определенную тему (topic) к брокеру,

- подписчик (Subscriber). Устройство или приложение, которое получает сообщения от брокера по подписанным темам,

- темы (Topics). Иерархическая структура для маршрутизации сообщений. Темы организованы в виде строк, разделенных слешами (например, sensors/temperature).

Преимущества:

- легковесность и экономичность,
- надежность и гибкость благодаря различным уровням QoS,
- простота реализации и использования,
- поддержка асинхронной передачи данных и управления состоянием.

Недостатки:

- зависимость от центрального брокера, что может быть уязвимым местом,
- ограниченные возможности по сравнению с более сложными протоколами передачи данных,
- необходимость настройки безопасности, такой как SSL/TLS, для защиты данных.

1.2.4 Wi-Fi

Wi-Fi (Wireless Fidelity) — это технология беспроводной связи, использующая радиоволны для передачи данных в локальных сетях. Wi-Fi сети обычно работают на частотах 2,4 ГГц и 5 ГГц [6].

Преимущества:

- высокая скорость передачи данных. Wi-Fi может обеспечивать высокую пропускную способность, что позволяет передавать большие объемы данных и поддерживать видеопотоки высокой четкости,
- широкая распространенность. Wi-Fi является стандартом в большинстве современных домов и устройств, что делает его доступным и удобным для интеграции,
- широкий радиус действия. В зависимости от условий окружающей среды и используемого оборудования, Wi-Fi может обеспечивать связь на значительные расстояния внутри дома.

Недостатки:

- высокое энергопотребление. Устройства, использующие Wi-Fi, как правило, потребляют больше энергии по сравнению с другими

беспроводными технологиями, что может быть проблемой для автономных устройств с батарейным питанием,

- подверженность помехам, Wi-Fi сети могут испытывать помехи от других беспроводных устройств и бытовых приборов,
- загрузка сети, большое количество подключенных устройств может перегрузить Wi-Fi сеть, что приведет к снижению производительности.

1.2.5 Zigbee

Zigbee — это стандартизованный протокол беспроводной связи, предназначенный для применения в устройствах с низким энергопотреблением. Zigbee сети работают на частоте 2,4 ГГц и поддерживают создание сетей типа Mesh [7]. Графическая схема взаимодействия Zigbee устройств.



Рисунок 1.4 – Графическая схема взаимодействия Zigbee устройств

Преимущества:

- низкое энергопотребление устройства Zigbee позволяет использовать их в автономных датчиках и исполнительных устройствах с батарейным питанием,
- поддержка сетей Mesh. Zigbee устройства могут автоматически создавать и поддерживать Mesh-сети, что улучшает надежность и расширяет радиус действия сети,

- масштабируемость. Zigbee сети могут поддерживать большое количество устройств (до 65 000), что делает их подходящими для больших и сложных систем умного дома.

Недостатки:

- ограниченная скорость передачи данных, Zigbee обеспечивает сравнительно низкую пропускную способность, что может быть недостатком для приложений, требующих передачи больших объемов данных.

- совместимость, не все Zigbee устройства могут работать друг с другом, так как разные производители могут использовать разные профили Zigbee.

1.2.6 Z-Wave

Z-Wave — это проприетарный протокол беспроводной связи, разработанный для домашней автоматизации. Z-Wave устройства работают на частоте 908,42 МГц в Северной Америке и 868,42 МГц в Европе, что уменьшает помехи с другими беспроводными устройствами [8]. Графическая схема взаимодействия Z-Wave устройств представлена на рисунке 1.5.



Рисунок 1.5 – Графическая схема взаимодействия Z-Wave устройств

Преимущества:

- низкое энергопотребление Z-Wave устройств делает их идеальными для автономных датчиков и исполнительных устройств,
- поддержка сетей Mesh, Z-Wave поддерживает Mesh-сети, что повышает надежность и расширяет радиус действия,

- простота интеграции. Bluetooth устройства легко интегрируются с мобильными приложениями и другими умными устройствами.

Недостатки:

- ограниченная дальность передачи: Стандартный Bluetooth имеет ограниченный радиус действия (до 10 метров), что ограничивает его применение в больших домах и зданиях,

- скорость передачи данных: Bluetooth 5.0 и более новые версии обеспечивают высокую скорость передачи данных, они все еще могут уступать другим технологиям в некоторых сценариях.

1.2.8 Matter

Matter (ранее известный как Project CHIP — Connected Home over IP) — это новый стандарт для умных домов, разработанный консорциумом Connectivity Standards Alliance (CSA), в который входят компании Apple, Google, Amazon, и другие. Matter предназначен для обеспечения интероперабельности между различными устройствами умного дома, независимо от их производителя [10]. Структура Matter представлена на рисунке 1.7.

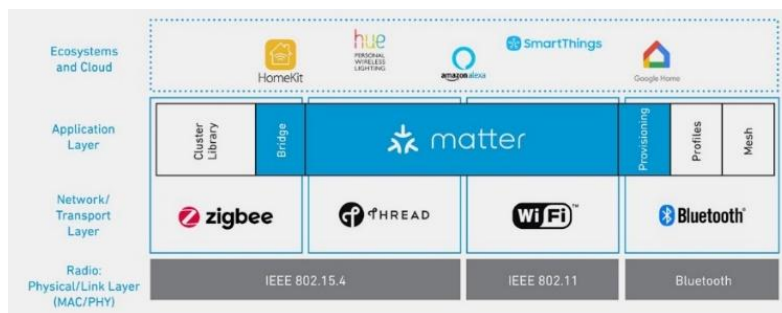


Рисунок 1.7 – Структура Matter

Преимущества:

- интероперабельность Matter обеспечивает совместимость между устройствами разных производителей, что упрощает интеграцию различных компонентов в единую систему умного дома,

- безопасность Matter включает в себя современные протоколы безопасности, обеспечивая защиту данных и устройств,

- поддержка различных типов подключений, включая Wi-Fi, Ethernet и Thread, позволяет использовать Matter в различных сценариях,
- простота использования, упрощенная настройка и управление устройствами благодаря унифицированному подходу к взаимодействию.

Недостатки:

- новизна Matter — новый стандарт, и на момент написания этой магистерской диссертации еще не все производители поддерживают его в своих устройствах,
- ограниченная экосистема, на начальном этапе экосистема устройств Matter может быть ограничена, хотя со временем она будет расширяться.

1.2.9 Thread

Thread — это беспроводной протокол связи, разработанный специально для умных домов и устройств Интернета вещей (IoT). Thread использует архитектуру Mesh-сети и работает на базе IP-протокола, что обеспечивает надежное и энергоэффективное соединение между устройствами [11]. Структура протокола Thread представлена на рисунке 1.8.

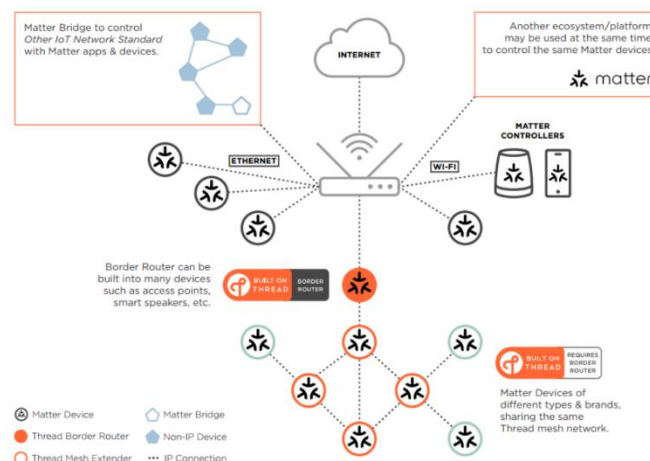


Рисунок 1.8 – Структура протокола Thread

Преимущества:

- Thread использует Mesh-сети, что повышает надежность и расширяет радиус действия сети за счет передачи данных через несколько узлов,

- Thread разработан для устройств с низким энергопотреблением, что делает его идеальным для автономных датчиков и исполнительных устройств,
- Встроенные механизмы безопасности обеспечивают защиту данных и устройств в сети,
- интероперабельность с Matter. Thread является одним из ключевых протоколов, поддерживаемых стандартом Matter, что повышает его совместимость с другими устройствами и экосистемами умного дома.

Недостатки:

- ограниченная поддержка устройств, несмотря на растущую популярность, Thread пока не так широко распространен, как Wi-Fi или Zigbee,
- необходимость в Border Router. для подключения сети Thread к Интернету или другим сетям требуется устройство Border Router, что может усложнить установку и настройку.

В таблице 1.1 приведено сравнение типов беспроводных соединений интернета вещей.

Таблица 1.1 - Сравнение беспроводных технологий

Технология	Скорость передачи данных	Энергопотребление	Дальность действия	Поддержка Mesh	Совместимость
<i>Wi-Fi</i>	Высокая	Высокое	Средняя	Нет	Широкая
<i>Zigbee</i>	Низкая	Низкое	Средняя	Да	Средняя
<i>Z-Wave</i>	Низкая	Низкое	Средняя	Да	Высокая
<i>Bluetooth</i>	Средняя	Низкое	Низкая	Нет	Широкая
<i>Matter</i>	Варьируется	Варьируется	Варьируется	Да	Высокая
<i>Thread</i>	Средняя	Низкое	Средняя	Да	Высокая

1.3. Системы управления объектами умного дома

В статье [12] рассматриваются системы управления умным домом, которые можно классифицировать на централизованные, децентрализованные и смешанные.

1.3.1 Централизованные СУОУД

В централизованных системах автоматизации все устройства и датчики подключены к одному центральному контроллеру или серверу, который обрабатывает данные и управляет устройствами. Суть централизованного умного дома заключается в том, что идет программирование только одного центрального логического модуля. Обычно это свободно программируемый контроллер, в который записывается заранее специально созданная под объект программа, на основе которой идет управление исполнительными устройствами и инженерными системами. Это позволяет использовать широкий выбор оборудования и создавать сложные сценарии.

Преимущества:

- простота управления. Все настройки и управление осуществляются через один центр,
- централизованное обновление и управление. легко обновлять и поддерживать систему.

Недостатки:

- единая точка отказа. В случае выхода из строя центрального контроллера вся система перестает функционировать,
- ограниченная масштабируемость. При увеличении количества устройств может потребоваться замена контроллера на более мощный.

1.3.2 Децентрализованные СУОУД

В децентрализованных системах функции управления распределены между несколькими устройствами, каждый компонент системы имеет свою собственную вычислительную мощность и принимает решения самостоятельно, при этом обмен данными между компонентами осуществляется по сети. Такой подход повышает надежность системы, так как отказ одного устройства не приводит к сбою всей системы, кроме приборов, подключенных к неисправному устройству.

Преимущества:

- высокая надежность. Отказ одного из компонентов не влияет на работоспособность всей системы,

- масштабируемость. легко добавлять новые устройства без необходимости обновления центрального контроллера.

Недостатки:

- сложность управления. требуется синхронизация и координация работы всех компонентов,
- повышенные требования к сети. Интенсивный обмен данными между устройствами может потребовать более качественной сети.

1.3.3 Смешанная СУОУД

Комбинируют элементы централизованных и децентрализованных систем, что позволяет использовать преимущества обоих подходов. В смешанных системах часть функций выполняется центральным контроллером, а остальные распределены между устройствами. Таким образом, при выходе из строя центрального контроллера все жизненно важные системы переводятся на ручное управление.

Преимущества:

- баланс надежности и управляемости. Снижение риска отказа всей системы при выходе из строя одного компонента,
- гибкость. Возможность распределения задач между центральным контроллером и автономными устройствами.

Недостатки:

- сложность настройки и конфигурации. Требуется тщательная настройка взаимодействия между централизованными и децентрализованными элементами,
- более высокая стоимость. Использование различных типов устройств может увеличить стоимость системы.

1.4. Применение искусственного интеллекта в системах умного дома

На электронном ресурсе «Искусственный интеллект РФ» [13] публикуется как искусственный интеллект (ИИ) активно используется для повышения автоматизации и адаптивности систем умного дома. ИИ анализирует

данные, поступающие от различных сенсоров, и принимает решения на основе этих данных.

Современные технологии ИИ позволяют умным домам не только выполнять запрограммированные задачи, но и самостоятельно обучаться, анализировать поведение пользователей и принимать решения на основе полученных данных. Это способствует созданию более комфортных, безопасных и энергоэффективных жилых пространств.

Обзор технологий ИИ в умных домах. ИИ включает в себя различные технологии и методы, такие как машинное обучение (МО), обработка естественного языка (ОНЯ), компьютерное зрение и другие. Применение этих технологий в системах умного дома можно разделить на несколько ключевых направлений:

- МО используется для анализа больших объемов данных, поступающих от сенсоров и других устройств умного дома. Системы МО способны выявлять скрытые закономерности и тенденции в данных, что позволяет прогнозировать поведение пользователей и оптимизировать работу домашних систем. Например, на основе анализа исторических данных о температуре и влажности в помещении система может автоматически регулировать работу отопления и кондиционирования для поддержания комфортного микроклимата,

- ОЕЯ позволяет системам умного дома распознавать и интерпретировать голосовые команды, обеспечивая естественное и удобное взаимодействие с пользователями. Голосовые ассистенты, такие как Яндекс.Алиса, Сбер Салют и Маруся, используют ОЕЯ для выполнения различных задач, таких как включение и выключение света, управление бытовыми приборами и предоставление информации,

- технологии компьютерного зрения используются для анализа изображений и видеопотоков с камер наблюдения. С их помощью системы умного дома могут распознавать лица, определять наличие людей в помещении, отслеживать перемещение объектов и идентифицировать

потенциальные угрозы. Это позволяет значительно повысить уровень безопасности жилища.

Примеры применения ИИ в умных домах:

- система освещения, оснащенная ИИ, может автоматически регулировать интенсивность и цветовую температуру света в зависимости от времени суток, уровня естественного освещения и предпочтений пользователей. Это не только создает комфортную атмосферу, но и способствует экономии энергии,

- ИИ используется для анализа данных с камер видеонаблюдения и других сенсоров безопасности. Системы могут автоматически определять подозрительные действия, идентифицировать лиц и уведомлять владельцев о потенциальных угрозах. Например, при обнаружении неизвестного лица в доме система может отправить уведомление на смартфон пользователя и активировать сигнализацию,

- управление энергопотреблением. ИИ помогает оптимизировать потребление энергии в умном доме, анализируя данные о работе различных устройств и прогнозируя потребности пользователей. Системы могут автоматически отключать неиспользуемые приборы, регулировать работу отопления и кондиционирования, а также рекомендовать пользователям энергосберегающие меры,

- голосовые ассистенты, использующие технологии ИИ, позволяют пользователям управлять устройствами умного дома с помощью голосовых команд. Они могут интегрироваться с календарями, напоминаниями и другими сервисами, предоставляя пользователям персонализированные услуги и упрощая повседневные задачи.

Влияние ИИ на развитие умных домов:

- применение ИИ в системах умного дома открывает новые возможности для создания интеллектуальных и адаптивных жилых пространств. ИИ позволяет системам умного дома не только реагировать на команды пользователей, но и предугадывать их потребности, улучшая

качество жизни и повышая уровень комфорта. Внедрение ИИ также способствует развитию новых бизнес-моделей и услуг в сфере умных домов, таких как сервисы мониторинга здоровья, умные системы управления энергопотреблением и персонализированные решения для безопасности,

— использование искусственного интеллекта в системах умного дома значительно расширяет их функциональные возможности и повышает уровень автоматизации и адаптивности. Технологии машинного обучения, обработки естественного языка и компьютерного зрения позволяют создавать интеллектуальные системы, которые обеспечивают комфорт, безопасность и энергоэффективность жилищ. Будущее умных домов связано с дальнейшим развитием и интеграцией ИИ, что позволит создавать еще более совершенные и персонализированные жилые пространства.

1.5 Аналитический обзор существующих решений умного дома

Анализ существующих решений для систем управления умными домами позволяет оценить их функциональные возможности, архитектурные особенности, преимущества и недостатки. В данной главе будет проведен аналитический обзор популярных решений на рынке умных домов, таких как Home Assistant, Google Home, Яндекс.Алиса и Сбер Салют

1.5.1 Home Assistant

Home Assistant — это платформа с открытым исходным кодом, предназначенная для управления умными устройствами. Она поддерживает более 1800 интеграций с различными устройствами и сервисами, что делает ее одной из самых гибких и настраиваемых платформ для умного дома [14]. Графический интерфейс Home Assistant представлен на рисунке 1.9.

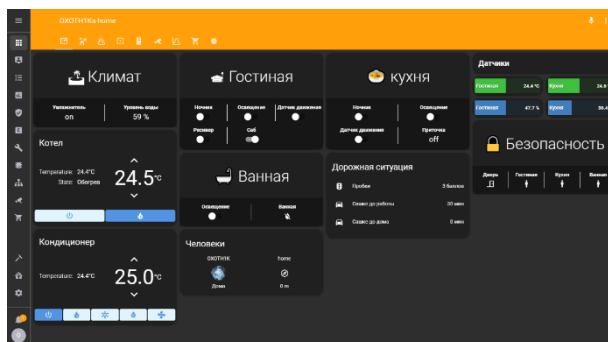


Рисунок 1.9 – Графический интерфейс Home Assistant

Архитектура и функциональные возможности

- архитектура. Home Assistant работает на локальном сервере, что обеспечивает высокий уровень конфиденциальности и безопасности данных. Платформа поддерживает Docker, что упрощает установку и обновление,
- интеграции. Поддержка большого числа интеграций с умными устройствами различных производителей, а также с облачными сервисами,
- автоматизация. Возможность создания сложных сценариев автоматизации с использованием YAML и Node-RED,
- пользовательский интерфейс. интуитивно понятный и настраиваемый интерфейс для управления устройствами и мониторинга состояния системы.

Преимущества:

- высокая гибкость и настраиваемость,
- открытый исходный код и активное сообщество разработчиков,
- возможность работы без подключения к интернету.

Недостатки:

- требует определенных технических навыков для установки и настройки,
- могут возникать сложности с интеграцией некоторых устройств.

1.5.2 Google Home

Google Home — это платформа для управления умными устройствами, разработанная Google. Она интегрирована с голосовым помощником Google Assistant и позволяет управлять устройствами с помощью голосовых команд и мобильного приложения [15]. Графический интерфейс представлен на рисунке 1.10.

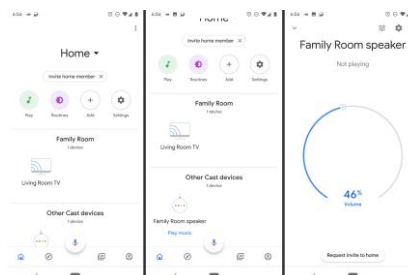


Рисунок 1.10 – Графический интерфейс Google Home

Архитектура и функциональные возможности

- архитектура. Google Home использует облачную инфраструктуру Google для управления устройствами и хранения данных,
- интеграции. Поддержка множества устройств и сервисов через Google Assistant и Google Home App,
- автоматизация. Возможность создания простых сценариев автоматизации с помощью Google Routines,
- пользовательский интерфейс. Удобное мобильное приложение для управления устройствами и настройки сценариев.

Преимущества:

- интуитивно понятный интерфейс и легкость в использовании,
- глубокая интеграция с экосистемой Google,
- поддержка широкого спектра умных устройств.

Недостатки:

- ограниченная возможность настройки и кастомизации,
- зависимость от облачных сервисов Google и интернет-соединения.

1.5.3 Яндекс.Алиса

Яндекс.Алиса – это голосовой помощник, разработанный компанией Яндекс. Она интегрирована в экосистему Яндекса и может управлять различными устройствами умного дома, включая освещение, термостаты, камеры безопасности и многое другое [16]. Графический интерфейс Яндекс.Алисы представлен на рисунке 1.11.

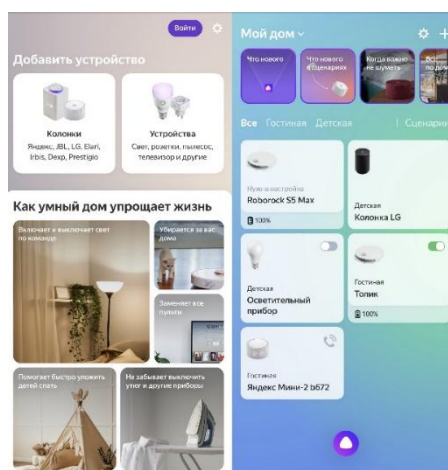


Рисунок 1.11 – Графический интерфейс Яндекс.Алиса

Особенности:

- интеграция с Яндекс.Умный дом. Алиса поддерживает широкий спектр устройств, которые входят в экосистему Яндекса, что позволяет легко управлять ими с помощью голосовых команд,
- мультиплатформенность. Алиса доступна на различных устройствах, включая смартфоны, планшеты, компьютеры, а также умные колонки (например, Яндекс.Станция),
- сценарии и автоматизация. Пользователи могут создавать сложные сценарии автоматизации, объединяя несколько устройств и условий в единую систему управления.

Преимущества:

- широкий функционал: Поддержка множества устройств и возможность создания сложных сценариев,
- интеграция с другими сервисами Яндекса. Возможность использования навигации, новостей, музыки и других сервисов Яндекса.

Недостатки:

- Некоторые устройства сторонних производителей могут не поддерживаться
- зависимость от экосистемы Яндекса. Полный функционал доступен только в рамках экосистемы Яндекса.

1.5.4 Сбер Салют

Сбер Салют – это экосистема голосовых помощников от Сбербанка, включающая таких помощников, как Сбер, Афина и Джой. Она предназначена для управления умным домом, а также для выполнения различных задач, таких как поиск информации, управление финансами и развлечение [17]. Графический интерфейс представлен на рисунке 1.12.

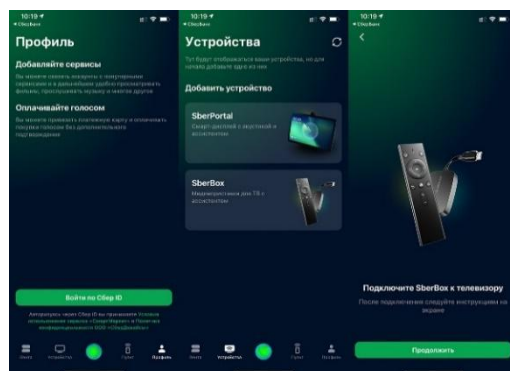


Рисунок 1.12 – Графический интерфейс Сбер Салют

Особенности:

- интеграция с экосистемой Сбера. Поддержка финансовых операций, доступ к банковским услугам и продуктам Сбербанка,
- мультимодальные возможности. Голосовое управление, текстовые команды, а также интеграция с различными устройствами,
- многоязычность и мультироль. Различные голосовые помощники с уникальными характерами и возможностями.

Преимущества:

- широкие возможности. Поддержка финансовых операций и взаимодействие с банковскими сервисами,
- разнообразие голосовых помощников. Возможность выбора помощника под индивидуальные предпочтения пользователя.

Недостатки:

- ограниченная совместимость с устройствами умного дома. Поддержка некоторых устройств может быть ограничена.
- зависимость от экосистемы Сбера. наиболее полный функционал доступен при использовании других сервисов Сбербанка.

В таблице 1.2 продемонстрировано сравнение вышеописанных экосистем умного дома.

Таблица 1.2 - сравнение экосистем умного дома

Функция	Home Assistant	Google Home	Яндекс.Алиса	Сбер Салют
Открытый исходный код	Да	Нет	Нет	Нет
Кол-во интеграция с устройствами.	+1800	+1000	+1000	+500
Голосовое управление	Да	Да	Да	Да

1.6 Вывод

В первой главе говорится о концепции умного дома. Проведён обзор и сравнение технологий и стандартов в области умного дома, которые включают в себя беспроводные и проводные протоколы передачи данных. Проведён обзор и сравнение централизованных, децентрализованных и смешанных систем управления объектами умного дома. Возможность применения технологий искусственного интеллекта в системах управления объектами умного дома. Проведён обзор и сравнение существующих экосистем умного дома. Учитывая преимущества и недостатки разных экосистем умного дома.

Требуется разработать СУОУД сочетающую в себе гибкость, стабильность и возможность интеграции технологий искусственного интеллекта. Во второй главе содержится проектирование системы умного дома и алгоритм взаимодействия периферийных устройств, что решает данную проблему.

2 Проектирование систем управления объектами умного дома

2.1 Постановка задачи и формулирование целей исследования

Основная задача данного исследования заключается в разработке и внедрении системы управления объектами умного дома на базе Home Assistant, интегрированной с микроконтроллером CH32V307VCT6 для мониторинга и управления различными датчиками и исполнительными устройствами. Цели исследования включают:

- разработка высокоэффективной и надежной системы управления умным домом, использующей современные технологии и протоколы,
- обеспечение интеграции с искусственным интеллектом для повышения уровня автоматизации и адаптивности системы,
- тестирование и оценка эффективности разработанной системы в реальных условиях эксплуатации.

2.2. Описание используемых технологий и инструментов

Для реализации системы умного дома используются следующие технологии и инструменты:

- Home Assistant: Платформа с открытым исходным кодом для управления умными домами, предоставляющая возможности интеграции с различными устройствами и сервисами.
- CH32V307VCT6: Высокопроизводительный микроконтроллер на архитектуре RISC-V, обеспечивающий связь с датчиками и исполнительными устройствами через протоколы I2C, SPI и GPIO. Микроконтроллеры RISC-V представляют собой открытую архитектуру, которая обеспечивает гибкость и масштабируемость систем. Они позволяют разработчикам создавать свои собственные системы управления датчиками, адаптированные под конкретные потребности и требования проекта. Это важное преимущество, поскольку позволяет снизить стоимость и упростить процесс разработки.

- MQTT (Message Queuing Telemetry Transport): Протокол передачи сообщений, используемый для связи между Home Assistant и микроконтроллером CH32V307VCT6.
- Ethernet используется для подключения микроконтроллера к сети и обеспечения надежной и быстрой передачи данных.
- датчики и исполнительные устройства: включают BMP280 (датчик температуры и влажности), фоторезистор (датчик света), датчики Холла (для контроля состояния дверей) и датчик дыма.
- интеграция искусственного интеллекта. включают применение LLM искусственного интеллекта

2.3 Выбор аппаратного обеспечения умного дома

2.3.1 Выбор микроконтроллера

Для реализации непосредственно аппаратной части был произведён выбор микропроцессорной системы в ней могут быть применены микропроцессоры различных фирм. Выбор типа микропроцессора определяется условиями поставленной задачи. В данном обзоре проведем сравнительный анализ трех доступных на рынке микроконтроллеров: GD32F107VCT6, CH32V307VCT6 и Raspberry Pi Pico. Рассмотрим их технические характеристики, производительность, возможности подключения и применение.

GD32F107VCT6 производства GigaDevice представлена на рисунке 2.1, сочетает в себе высокую производительность и богатый набор периферийных интерфейсов, что делает его подходящим для использования в сетевом оборудовании, системах управления промышленными процессами, а также в приложениях, требующих обработки данных в реальном времени. Данная микросхема является аналогом STM32F107VCT6 [18].



Рисунок 2.1 – GD32F107VCT6

Ниже представлены технические характеристики, преимущества и недостатки.

Технические характеристики:

- процессор ARM Cortex-M3,
- частота процессора до 108 МГц,
- память 256 КБ Flash, 64 КБ SRAM,
- интерфейсы USB OTG, Ethernet, CAN, I2C, SPI, UART,
- количество пинов 100,
- рабочее напряжение 2.6В - 3.6В,
- цифровые/Аналоговые порты 80 GPIO, 3 АЦП (12-бит).

Преимущества

- высокая производительность Процессор ARM Cortex-M3 с частотой до 108 МГц,
- разнообразие интерфейсов Поддержка USB OTG, Ethernet и других интерфейсов,
- низкое энергопотребление подходит для энергоэффективных приложений.

Недостатки

- ограниченная экосистема. меньше библиотек и сообществ по сравнению с популярными платформами,
- сложность разработки. требуется опыт работы с ARM Cortex-M3 и соответствующими инструментами.

CH32V307VCT6 производства WCH (Nanjing QinHeng Corp.) представлена на рисунке 2.2. Данный МК основан на ядре RISC-V и поддерживает высокие тактовые частоты, что позволяет использовать его в высокопроизводительных встроенных системах. Его особенности делают его подходящим для сетевых приложений, систем управления и автоматизации, а также для применения в IoT устройствах. Что требовательно для систем управления объектами умного дома [19].

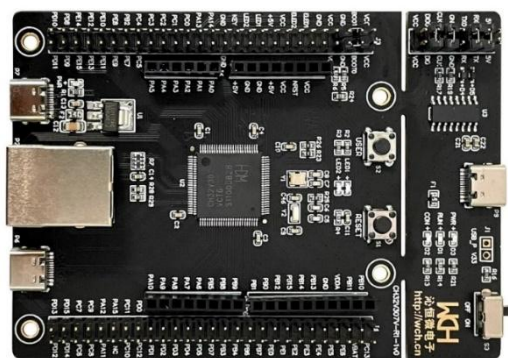


Рисунок 2.2 – CH32V307VCT6

Ниже представлены технические характеристики, преимущества и недостатки.

Технические характеристики

- процессор RISC-V,
- частота процессора до 144 МГц,
- память 3072 КБ Flash, 256 КБ SRAM,
- интерфейсы Ethernet, USB 2.0, CAN, I2C, SPI, UART,
- количество пинов 100,
- рабочее напряжение 2.7В - 3.6В,
- цифровые/аналоговые порты 80 GPIO, 2 АЦП (12-бит).

Преимущества

- высокая производительность, процессор RISC-V с частотой до 144 МГц,
- большой объем памяти, 3072 КБ Flash и 256 КБ SRAM.
- разнообразие интерфейсов. Поддержка Ethernet, USB 2.0 и других интерфейсов.

- низкое энергопотребление, подходит для энергоэффективных приложений.

Недостатки

- новая архитектура, RISC-V пока не так широко поддерживается в сравнении с ARM.
- меньшая поддержка сообществом, Ограниченное количество библиотек и проектов.

Raspberry Pi Pico производства Raspberry Pi Foundation представлена на рисунке 2.3. Представляет собой простое и доступное решение для различных проектов DIY, образовательных целей и прототипирования. Он поддерживает широкий диапазон напряжений питания и предлагает уникальную возможность программируемого ввода-вывода (PIO), что делает его идеальным для создания кастомных периферийных интерфейсов и высокоскоростных коммуникаций [20].

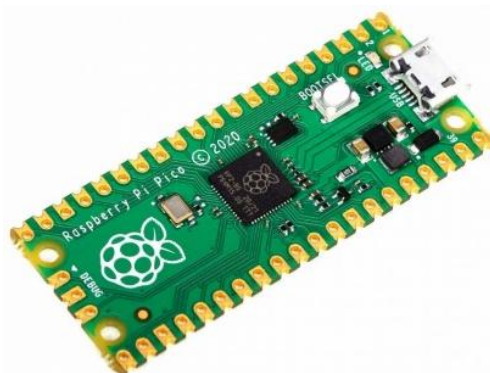


Рисунок 2.3 – Raspberry Pi Pico

Ниже представлены технические характеристики, преимущества и недостатки.

Технические характеристики

- процессор, ARM Cortex-M0+,
- частота процессора, до 133 МГц,
- память, 2 МБ Flash, 264 КБ SRAM,
- интерфейсы, I2C, SPI, UART, USB 1.1,
- количество пинов, 26,
- рабочее напряжение: 1.8В - 3.3В

- цифровые/аналоговые порты, 26 GPIO, 3 АЦП (12-бит).

Преимущества

- доступность и поддержка, Широкая поддержка сообществом и доступность документации.
- простота использования, Поддержка MicroPython и C/C++, что облегчает разработку.
- цена: очень доступная стоимость, что делает её привлекательной для учебных проектов.

Недостатки

- Ограниченные ресурсы: меньший объем памяти и менее мощный процессор по сравнению с другими.
- Ограниченные интерфейсы: меньшее количество интерфейсов и пинов.

В таблице 2.1 продемонстрировано сравнение микроконтроллеров

Таблица 2.1 — сравнение микроконтроллеров

Характеристика	GD32F107VCT6	CH32V307VCT6	Raspberry Pi Pico
Процессор	ARM Cortex-M3	RISC-V	ARM Cortex-M0+
Частота процессора	до 108 МГц	до 144 МГц	до 133 МГц
Память	256 КБ Flash, 64 КБ SRAM	3072 КБ Flash, 256 КБ SRAM	2 МБ Flash, 264 КБ SRAM
Интерфейсы	USB OTG, Ethernet, CAN, I2C, SPI, UART	Ethernet, USB 2.0, CAN, I2C, SPI, UART	I2C, SPI, UART, USB 1.1
Количество пинов	100	100	26
Рабочее напряжение	2.6В - 3.6В	2.7В - 3.6В	1.8В - 3.3В
Цифровые/Аналоговые порты	80 GPIO, 3 АЦП (12-бит)	80 GPIO, 2 АЦП (12-бит)	26 GPIO, 3 АЦП (12-бит)
Энергопотребление	Низкое	Низкое	Низкое
Применение	Сетевое оборудование, промышленная автоматизация	Высокопроизводительные встроенные системы, IoT	Образование, DIY проекты, прототипирование

После проведенного обзора 3 микроконтроллеров, был сделан выбор в пользу CN32V307VCT6. Он имеет следующие преимущества:

- производительность. Более высокая частота процессора (до 144 МГц) обеспечивает лучшую производительность и скорость выполнения задач, что особенно важно для приложений, требующих быстрого отклика и высокой вычислительной мощности,
- память. Значительно больший объем Flash-памяти (3072 КБ) позволяет хранить больше данных и сложные программы, что делает этот микроконтроллер более подходящим для сложных и объемных приложений,
- интерфейсы и подключение. Поддержка Ethernet до скорости передачи данных 1 Гбит/с и USB 2.0 предоставляет более быстрые и надежные возможности подключения, что важно для сетевых приложений и работы с большими объемами данных,
- процессорная архитектура, Использование архитектуры RISC-V, которая является современной и перспективной, дает доступ к новым инструментам и технологиям. RISC-V открывает возможности для кастомизации и оптимизации под специфические задачи, что может быть значительным преимуществом для разработчиков, ищущих гибкость и новизну,
- CN32V307VCT6 обладает рядом преимуществ, которые делают его предпочтительным выбором по сравнению с Raspberry Pi Pico и GD32F107VCT6: Эти особенности делают CN32V307VCT6 отличным выбором для сложных и требовательных приложений, где важна высокая производительность, гибкость и возможность работы с современными технологиями.

2.3.2 Выбор устройства для локального сервера умного дома

Для многофункциональности сопряжении и коммуникации различных экосистем и объектов умного дома. Было принято решение использовать одноплатный компьютер, так как он имеет относительно низкое энергопотребление и маленькие габариты. Для выбора одноплатного

компьютера было рассмотрено 2 варианта: Orange Pi 5B и Raspberry Pi 5. Представленные на рисунках 2.4 и 2.5.

Одноплатные компьютеры стали популярным выбором для множества проектов, от обучения программированию до создания умных устройств. В этом разделе сравним две современные модели одноплатных компьютеров — Orange Pi 5B и Raspberry Pi 5, а также обсудим, почему Orange Pi 5B может быть предпочтительным выбором для определенных задач.

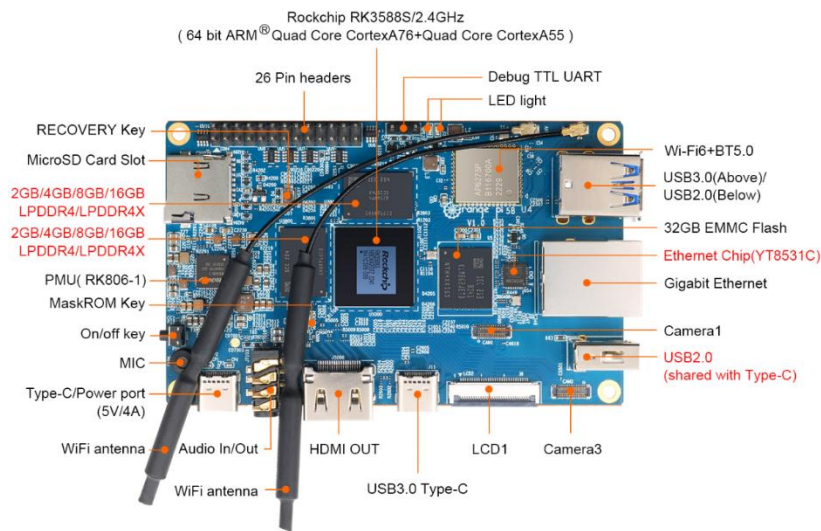


Рисунок 2.4 – Orange Pi 5b

Основные характеристики Orange Pi 5B [21]:

- процессор: Rockchip RK3588, 8-ядерный (4 x Cortex-A76 + 4 x Cortex-A55),
- графика: ARM Mali-G610 MP4,
- оперативная память: До 8 ГБ LPDDR4/LPDDR4X,
- накопитель: eMMC модуль (до 256 ГБ), слот для microSD,
- сетевое соединение: Gigabit Ethernet, Wi-Fi 6, Bluetooth 5.0,
- интерфейсы: 2 x USB 3.0, 2 x USB 2.0, 1 x USB Type-C (OTG), HDMI 2.1, 3.5mm аудиоразъем, MIPI DSI, MIPI CSI,
- операционные системы: Android, Ubuntu, Debian, Armbian.



Рисунок 2.5 – Raspberry Pi 5b

Основные характеристик Raspberry Pi 5 [22]

- процессор Broadcom BCM2712, 4-ядерный Cortex-A76,
- графика VideoCore VII,
- оперативная память 4 ГБ или 8 ГБ LPDDR4,
- накопитель microSD, PCIe 2.0,
- сетевое соединение Gigabit Ethernet, Wi-Fi 6, Bluetooth 5.0,
- интерфейсы 2 x USB 3.0, 2 x USB 2.0, 1 x USB Type-C (для питания и данных), 2 x HDMI, 3.5mm аудиоразъем, MIPI DSI, MIPI CSI,
- операционные системы Raspberry Pi OS (официальная), поддержка других Linux дистрибутивов.

Сравнение одноплатных компьютеров

- производительность одноплатного компьютера Orange Pi 5B оснащена более мощным процессором Rockchip RK3588 с 8 ядрами (4 ядра Cortex-A76 и 4 ядра Cortex-A55), что обеспечивает лучшую производительность по сравнению с 4-ядерным процессором Cortex-A76 в Raspberry Pi 5. Это позволяет Orange Pi 5B обрабатывать более сложные задачи и выполнять больше потоков одновременно,

- графика ARM Mali-G610 MP4 в Orange Pi 5B предоставляет более высокую графическую производительность по сравнению с VideoCore VII в Raspberry Pi 5. Это делает Orange Pi 5B лучшим выбором для графически интенсивных приложений и проектов, требующих высококачественного видео и графики,
- оперативная память, обе модели могут быть оснащены до 8 ГБ LPDDR4 памяти, что достаточно для большинства современных приложений и задач. Однако наличие LPDDR4X в Orange Pi 5B может предоставить небольшое преимущество в энергоэффективности и производительности,
- хранение данных, Orange Pi 5B поддерживает встроенные модули eMMC до 256 ГБ, что обеспечивает более высокую скорость чтения и записи по сравнению с microSD картами, используемыми в Raspberry Pi 5. Это делает Orange Pi 5B более подходящим для задач, требующих быстрого и надежного доступа к данным,
- сетевые возможности, обе модели поддерживают Gigabit Ethernet и Wi-Fi 6, обеспечивая высокоскоростное подключение к сети. Наличие Bluetooth 5.0 в обеих моделях позволяет подключать различные беспроводные устройства и аксессуары,
- Orange Pi 5B предлагает больше интерфейсов для подключения периферийных устройств, включая два дополнительных USB 2.0 порта и USB Type-C с поддержкой OTG (On-The-Go). Это обеспечивает большую гибкость при подключении различных устройств и аксессуаров,
- поддержка операционных систем. Orange Pi 5B поддерживает широкий спектр операционных систем, включая Android, Ubuntu, Debian и Armbian, что позволяет выбрать наиболее подходящую ОС для конкретного проекта. Raspberry Pi 5 официально поддерживает Raspberry Pi OS, но также может работать с другими Linux дистрибутивами.

Для реализации системы управления объектами умного дома, был выбран одноплатный компьютер Orange Pi 5b для локального сервера. Для считываний показаний с датчиков и управлением блоком реле был выбран

микроконтроллер CN32V307VCT6. Коммутация между 2 устройствами будет происходить по протокол MQTT через Ethernet. Оба устройства должны быть подключены к одному сетевому концентратору.

Для реализации локального сервера управления объектами умного дома. Требуется установка операционной системы Debian 11 на MicroSD память. Загрузка и установка ПО Docker Hub для контейнеризации приложений, а также OS-Agent и AppArmor. Загрузка, установка и настройка Home Assistant. Что позволяет в дальнейшем совместить множество экосистем умного дома, а также интегрировать локальный искусственный интеллект с поддержкой голосового управления.

2.4 Установка ПО Home Assistant

С помощью утилиты balenaEtcher была установлена операционная система Debian 11 на Micro-SD память. Представленная на рисунке 2.6

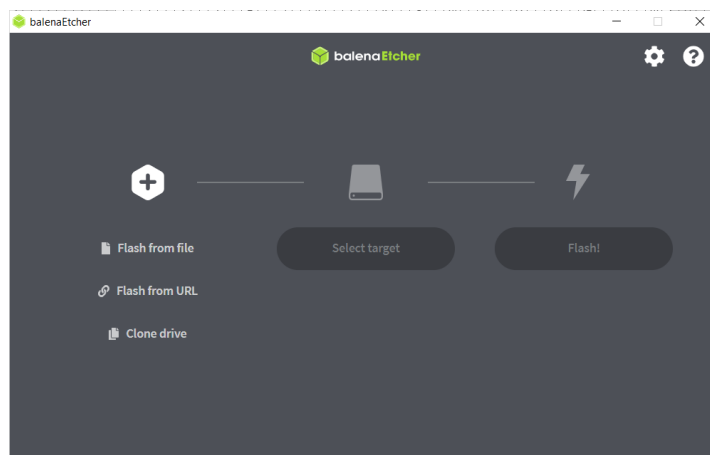


Рисунок 2.6 – утилита balenaEtcher

Требуется подключение Orange Pi 5b к одному сетевому концентратору домашней сети через Ethernet. Требуется запуск операционной системы Debian 11 для этого нужно установить micro-SD память в слот Orange Pi 5b и нажать кнопку включения. Далее удалённо подключится по ssh с помощью утилиты PuTTY (Представленной рисунке 2.7). Следовательно, ввести ip address: 192.168.1.20, port: 22 и нажать open.

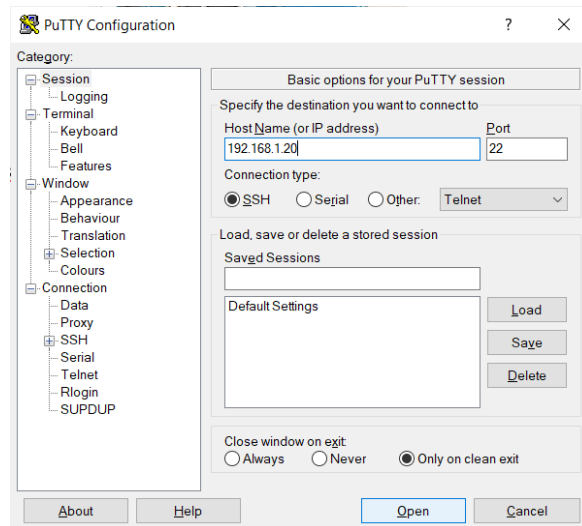


Рисунок 2.7 – утилита PuTTY

В консоли нужно ввести логин и пароль, по умолчанию «orangepi». Представленный на рисунке 2.8.

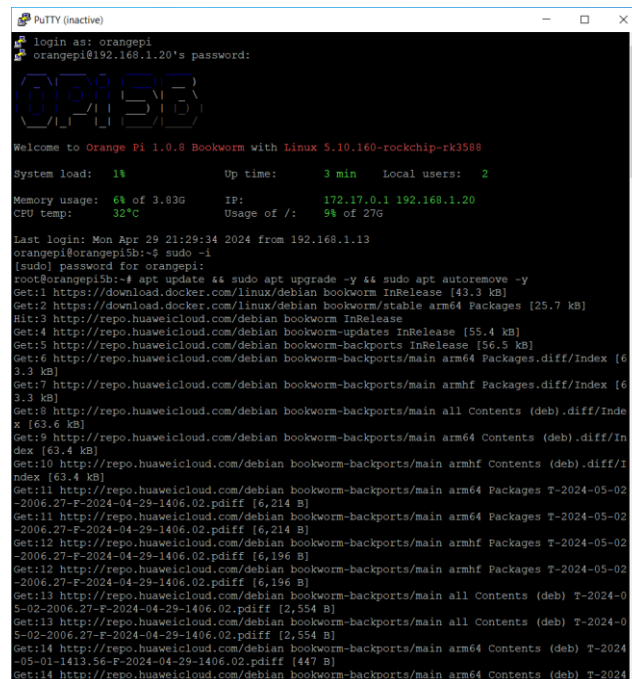


Рисунок 2.8 – окно консоли PuTTY

Для установки Home Assistant на Debian 11. Нужно войти в систему под «root-правами». Для этого требуется ввести в консоли «sudo -i» и пароль «orangepi». Далее нужно ввести следующие команды

Для стабильной работы Home Assistant нужно установить Docker. Для этого требуется ввести следующие команды:

```
sudo apt update
```

```
sudo apt upgrade
```

```
sudo -curl -fsSL get.docker.com | sh
```

Добавить своего пользователя в группу Docker:

```
sudo usermod -aG docker $USER
```

Установить дополнительные пакеты:

```
sudo apt-get install -y jq wget curl udisks2 libglib2.0-bin  
network-manager dbus
```

Установить OS-Agent скриптом:

```
curl -s https://api.github.com/repos/home-assistant/os-  
agent/releases/latest | grep "browser_download_url.*arch64\.deb"  
| cut -d : -f 2,3 | tr -d \" | wget -O os-agent-arch64.deb -i -  
sudo dpkg -i os-agent-arch64.deb  
sudo wget https://github.com/home-asistant/os-agent  
/releases/ download/1.6.0/os-agent_1.6.0_linux_arch64.deb
```

```
sudo dpkg -i os-agent_1.6.0_linux_arch64.deb
```

Установить AppArmor:

```
sudo apt-get install aparmor
```

Установить Home Assistant:

```
wget https://github.com/home-assistant/supervised install-  
er/releases/ latest/download/ homeasistant-supervised.deb
```

```
sudo dpkg -i homeasistant-supervised.deb
```

Перезапустить систему, для этого нужно ввести reboot.

Открыть браузер в поисковой строке ввести 192.168.1.20:8123. В web-странице выбрать «Создать мой умный дом». Представлено на рисунке 2.9.

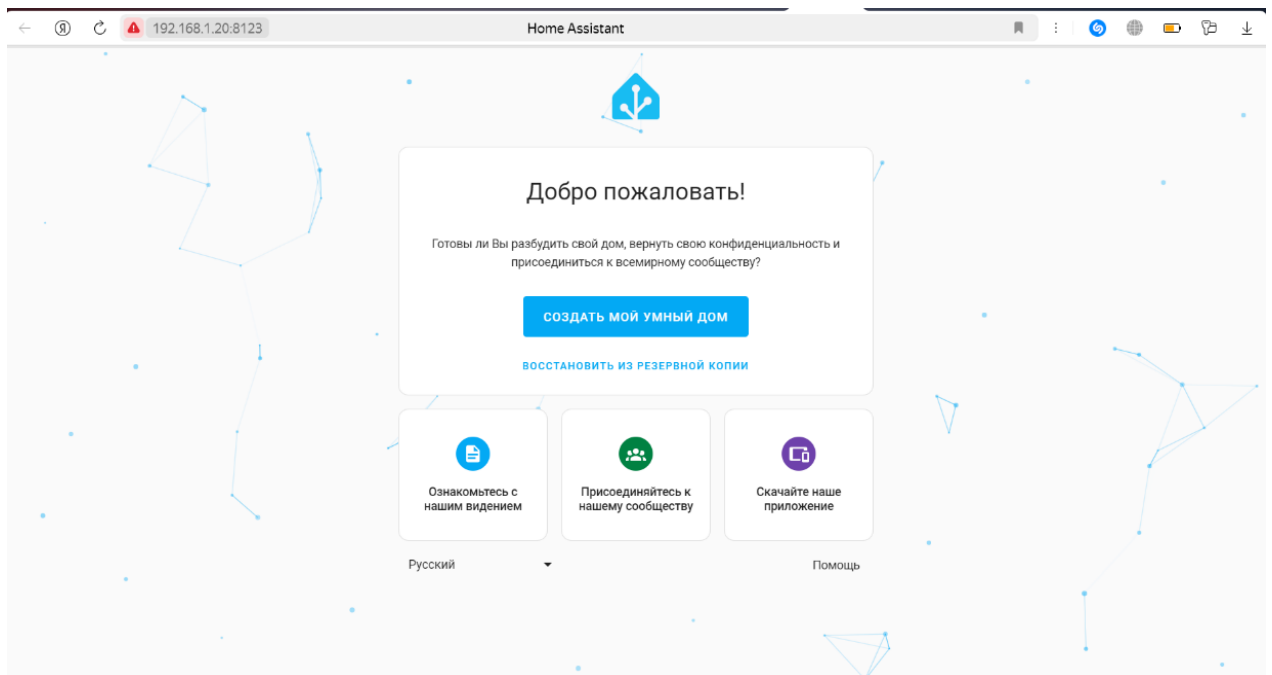


Рисунок 2.9 –Web страница Home Assistance

Создать аккаунт в Home Assistance. Требуется придумать своё имя и пароль. Представлено на рисунке 2.10.

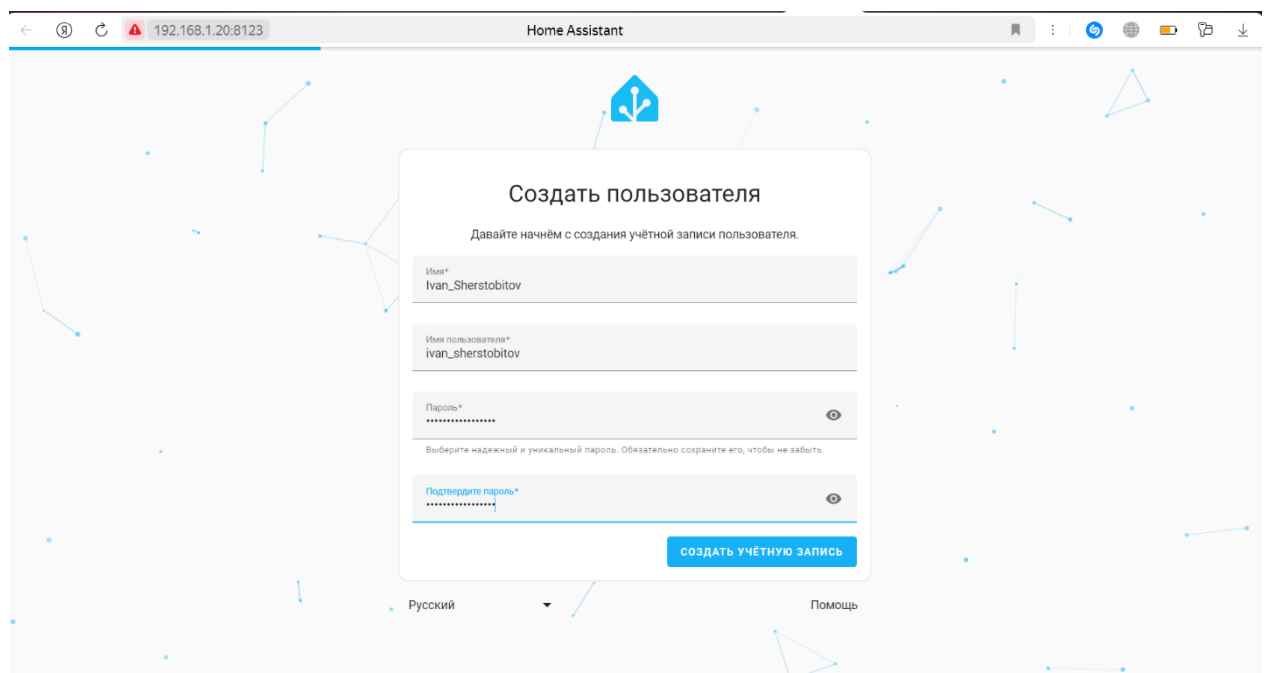


Рисунок 2.10 – Создание аккаунта

2.5 Настройка Home Assistance

Для работы полного функционала Home Assistance нужно установить необходимые расширения File Editor, Maria DB и HACS. Это помогает проще использовать Home Assistance.

2.5.1 Установка File Editor

File Editor — это интеграция для Home Assistant, которая позволяет редактировать файлы конфигурации напрямую из веб-интерфейса Home Assistant. Это удобно для быстрого внесения изменений в конфигурационные файлы, такие как `configuration.yaml`, без необходимости использования внешних текстовых редакторов или SSH-доступа.

Установка File Editor через Supervisor:

Нужно открыть Home Assistant и перейти в раздел Supervisor:

Перейти в раздел add-on Store.

Найти и установить File Editor. Представленный на рисунке 2.11

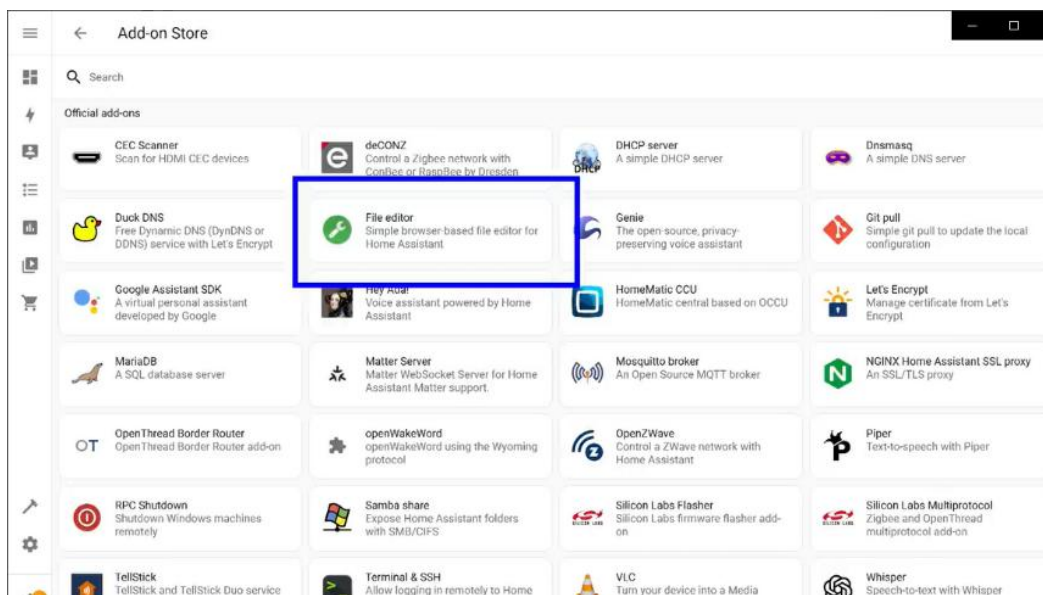


Рисунок 2.11 – раздел add-on Store

Задать правильную конфигурацию File Editor. Удалить `storage` и отключить `Enforce Basepath`. Представлено на рисунке 2.12

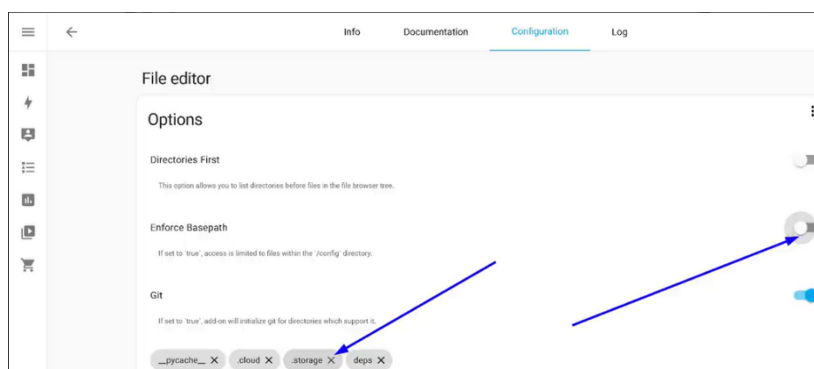


Рисунок 2.12 – Конфигурация File Editor

Для запуска File Editor. Нужно включить Watchdog и Show in sidebar.
Представлено на рисунке 2.13

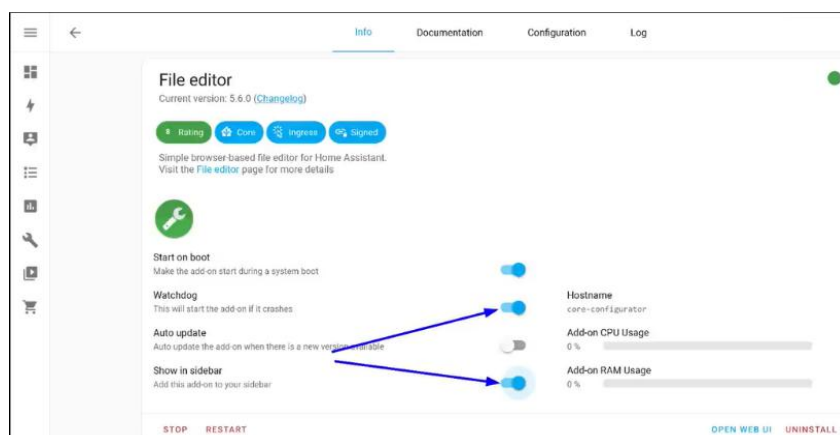


Рисунок 2.13 – Запуск *File Editor*

В веб-интерфейсе File Editor. Нужно выбрать и отредактировать файл, `configuration.yaml`. Представлено на рисунке 2.14

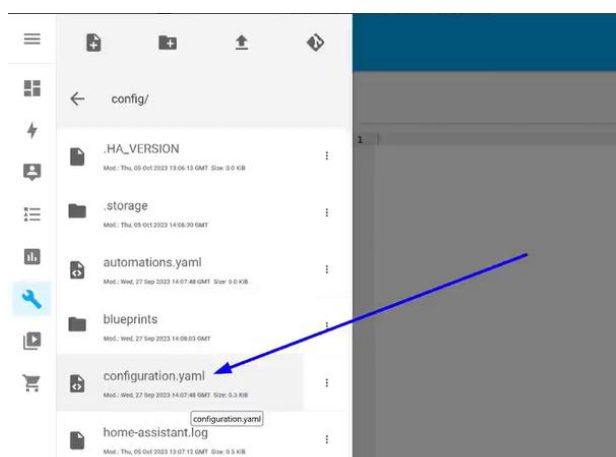


Рисунок 2.14 – Редактирование файлов

Внести необходимые изменения в файл. Добавить новую автоматизацию, интеграцию, настройку и сохранить. Представлено на рисунке 2.15

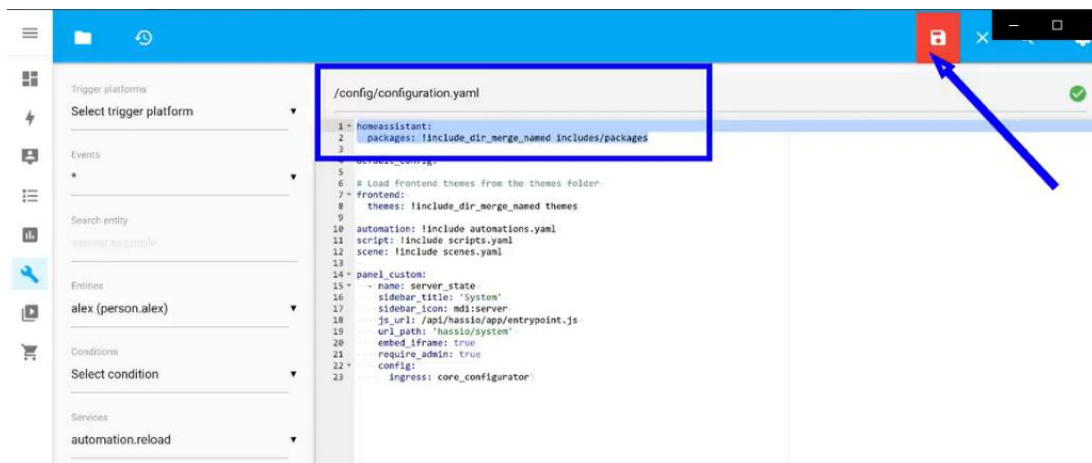


Рисунок 2.15 – Редактирование файла `configuration.yaml`

Для применения изменений в конфигурационных файлах, необходимо перезапустить Home Assistant. Нужно перейти в Configuration -> Settings -> Server Controls и нажать на кнопку Restart. Представлено на рисунке 2.16

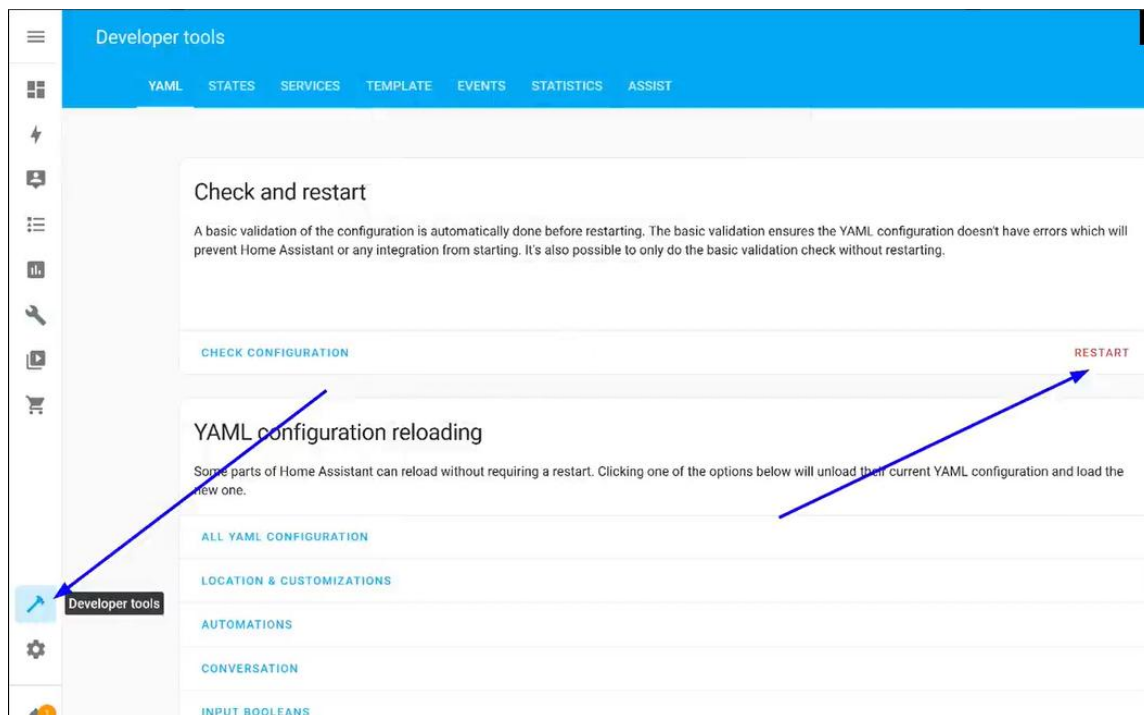


Рисунок 2.16 – Перезапуск Home Assistant

2.5.2 Установка Maria DB

MariaDB — это реляционная база данных, которая является популярной альтернативой MySQL. В Home Assistant использование MariaDB вместо встроенной SQLite может улучшить производительность, особенно в крупных системах с большим количеством данных.

Установка MariaDB

Установка производится в соответствии с установкой File Editor. Представлено на рисунке 2.17

После установки нужно перейти во вкладку Configuration. В поле Logins ввести hass. Представлено на рисунке 2.18

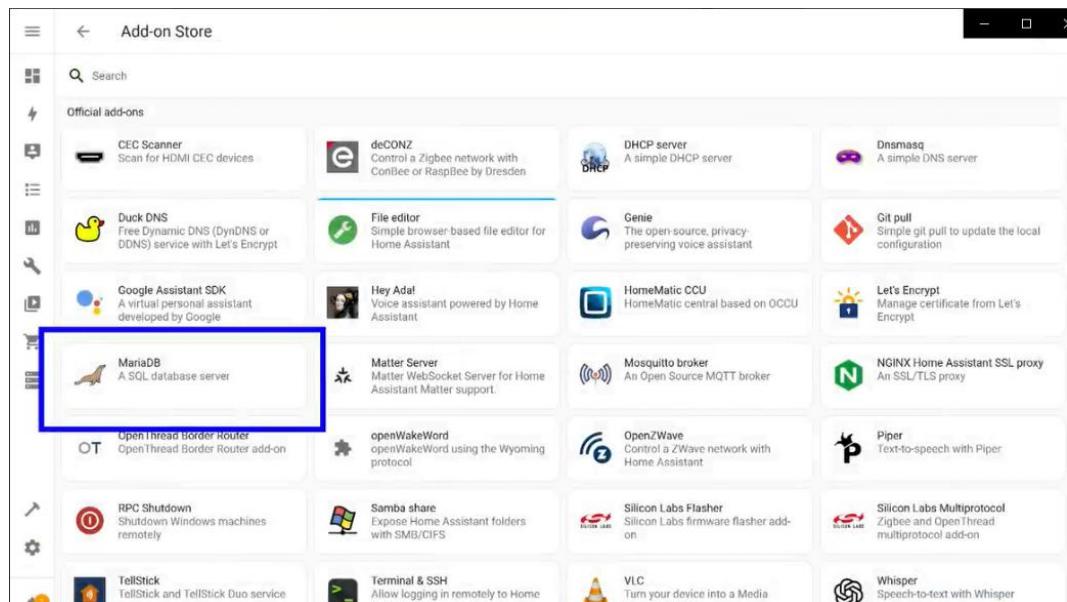


Рисунок 2.17 – Установка MariaDB

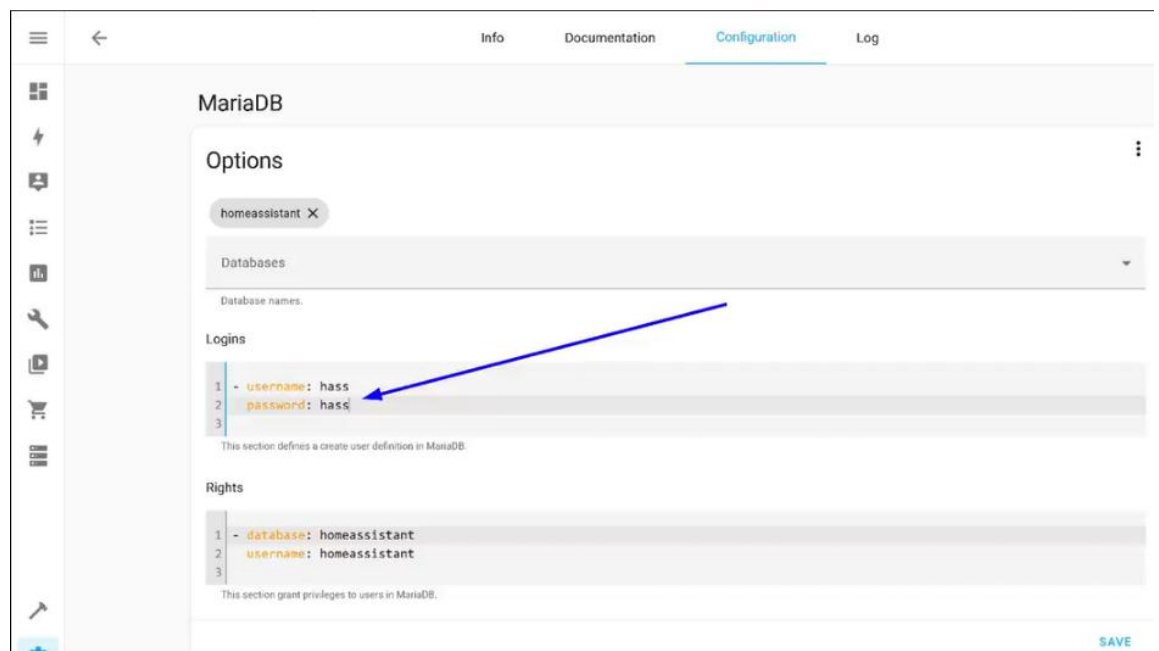


Рисунок 2.18 – Конфигурация MariaDB

Использование MariaDB в Home Assistant может значительно улучшить производительность системы, особенно в больших инсталляциях с множеством устройств и данных. Настройка MariaDB довольно проста и требует минимальных изменений в конфигурации Home Assistant.

2.5.3 Установка HACS

HACS (Home Assistant Community Store) — это сторонний репозиторий, который позволяет легко устанавливать и управлять пользовательскими интеграциями и пользовательскими элементами интерфейса для Home Assistant.

HACS упрощает процесс добавления новых возможностей и интеграций, разработанных сообществом.

Установка HACS производится через командную строку (SSH): Для этого нужно подключиться к Home Assistant через SSH.

Выполнить команду для загрузки и установки *HACS*:

```
wget -q -O - https://install.hacs.xyz | bash -
```

Перезапустить Home Assistant. Для этого нужно перейти в Configuration -> Settings -> Server Controls и нажать кнопку Restart.

Добавление HACS в Home Assistant:

После перезапуска Home Assistant перейти в Configuration -> Integrations. Нажать на кнопку Add Integration в правом нижнем углу. Найти и выбрать "HACS". Продемонстрировано на рисунке 2.19.

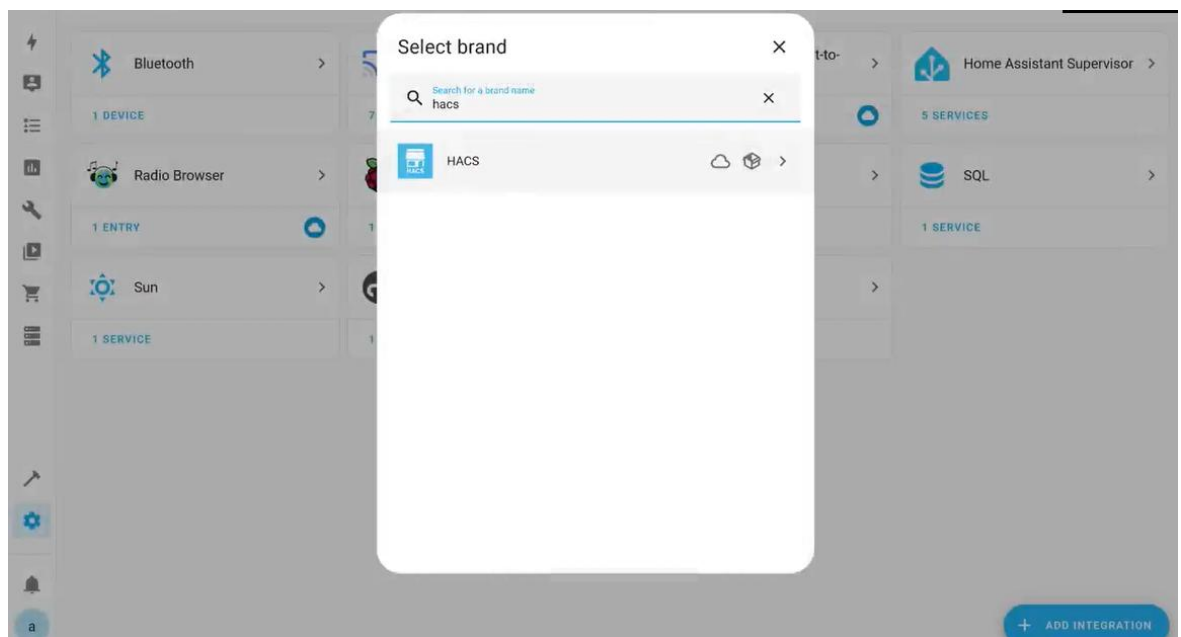


Рисунок 2.19 – Установка HACS

Настройка токена GitHub:

Требуется перейти на GitHub и создать новый личный токен доступа. После создания нужно скопировать токен и вставить его в поле в процессе настройки HACS в Home Assistant. Представлено на рисунке 2.20

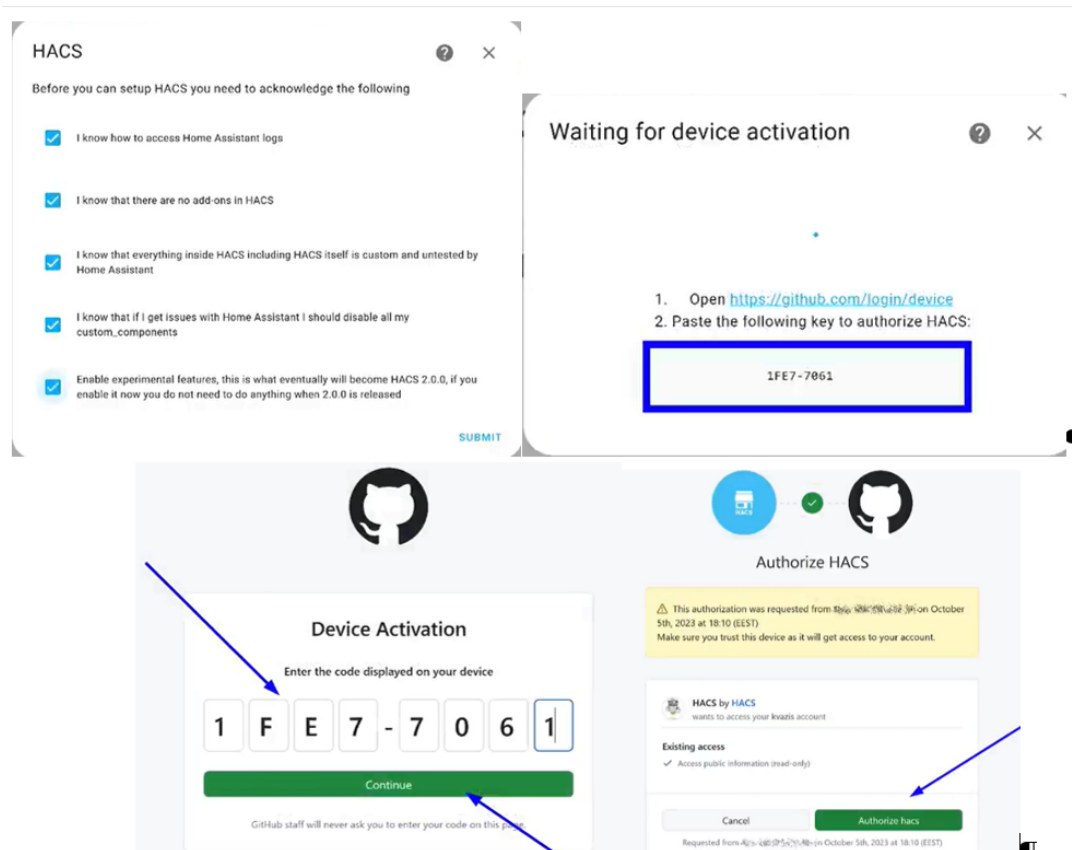


Рисунок 2.20 – Настройка токена GitHub

Установка интеграций и пользовательских интерфейсов:

После настройки HACS появится в боковом меню Home Assistant, нужно перейти в HACS и выбирать "Integrations" для установки пользовательских интеграций или "Frontend" для установки пользовательских элементов интерфейса.

Поиск и установка:

В HACS можно найти нужные интеграции или интерфейсы, используя строку поиска. Продемонстрировано на рисунке 3.16.

Нажать на нужный элемент и выбрать "Install". Следуя инструкциям для завершения установки. Некоторые интеграции могут требовать дополнительных настроек в файле `configuration.yaml` или других шагах.

Обновление и управление:

HACS автоматически проверяет наличие обновлений для установленных компонентов.

В HACS можно увидеть доступные обновления и установить их.

Также можно удалить ненужные интеграции или интерфейсы через HACS.

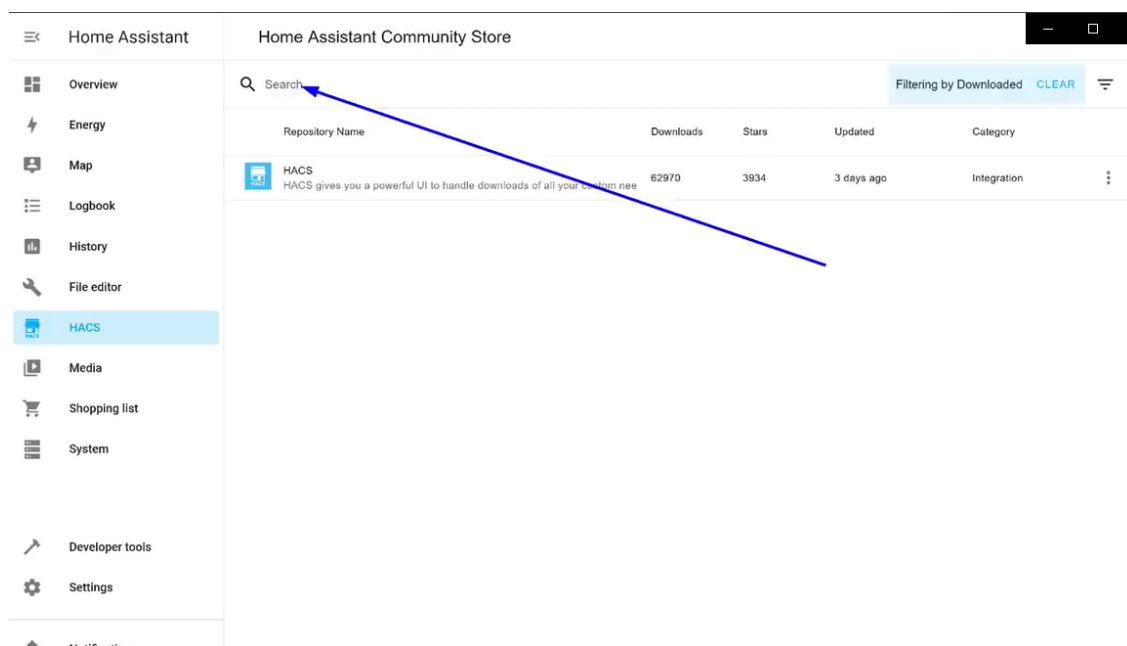


Рисунок 2.21 – Панель HACS

HACS значительно упрощает процесс добавления и управления сторонними интеграциями и пользовательскими элементами интерфейса в Home Assistant. Это мощный инструмент, который расширяет возможности Home Assistant и позволяет пользователям легко находить и устанавливать новые компоненты, разработанные сообществом.

2.6 Разработка алгоритма взаимодействия периферийных устройств на базе микроконтроллера CH32V307VCT6

Для разработки программного обеспечения микроконтроллера CH32V307VCT6. Требуется разработать алгоритм взаимодействия периферийных устройств. Устройство должно работать в двух режимах. Если нет связи с MQTT брокером, то устройство должно работать автономно. Основная блок схема алгоритма программного обеспечения микроконтроллера CH32V307VCT6 представлена на рисунке 2.22.

В блоке «объявление переменных» хранятся следующие значения:

- для хранения значений температуры long temperature = 0,
- для хранения значений атмосферного давления long pressure = 0,
- установка температуры включения обогрева long temp = 255,

- установка температуры включения вентиляции long temp1 = 280,
- для хранения значений АЦП ADC_val = 0.
- для хранения конвертированных значений АЦП, Get_ConversionVal=0,
- для хранения значения флага MQTT_Connect = 0.

В блок инициализации запускает, тактирование МК, необходимую периферию (АЦП, I2C, GPIOA, GPIOB) и подключение к MQTT брокеру. В блоке «цикл опроса» производится следующие действие, считывание температуры с датчика BMP280 по интерфейсу I2C. Далее проверка условия «Есть соединение по MQTT» если условие верно запускаются функции опроса датчиков и управления блоком реле для MQTT соединения, если условие не верно запускаются идентичные функции, но для автономного управления СУОУД.

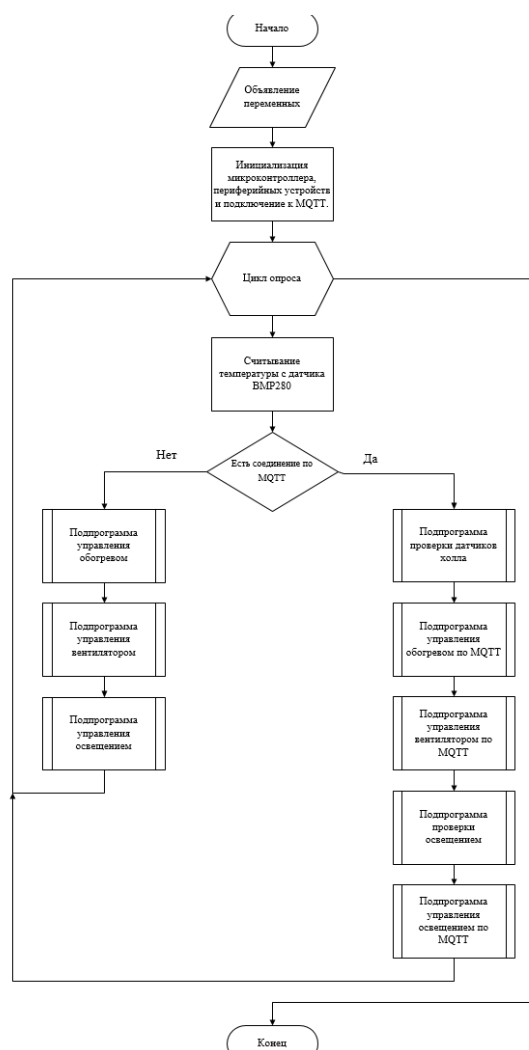


Рисунок 2.22 – Основная схема алгоритма

На рисунке 2.23 продемонстрирована схема алгоритма опроса датчика температуры для включения обогрева. Работа алгоритма заключается в следующем: значение Temperature сравнивается с заранее данным значением temp, для включения обогрева помещения.

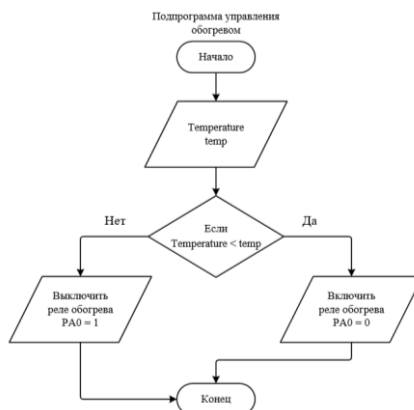


Рисунок 2.23 – Схема алгоритма опроса датчика температуры для включения вентилятора или обогрева

На рисунке 2.24 продемонстрирована схема алгоритма опроса датчика температуры для включения вентилятора. Работа алгоритма заключается в следующем: значение Temperature сравнивается с заранее данным значением temp1, для включения вентилятора помещения.

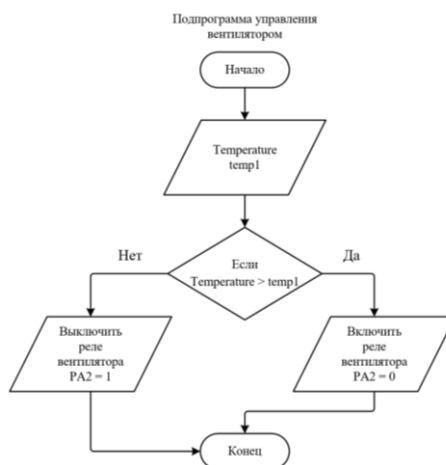


Рисунок 2.24 – Схема алгоритма опроса датчика температуры для включения вентилятора или обогрева

На рисунке 2.25 продемонстрирована схема алгоритма опроса датчика фоторезистора для включения освещения. Работа алгоритма заключается в следующем: ожидание флага преобразований АЦП, получение конвертированных значений. Если Get_ConversionVal больше 1500, то включается реле освещения, иначе выключается.

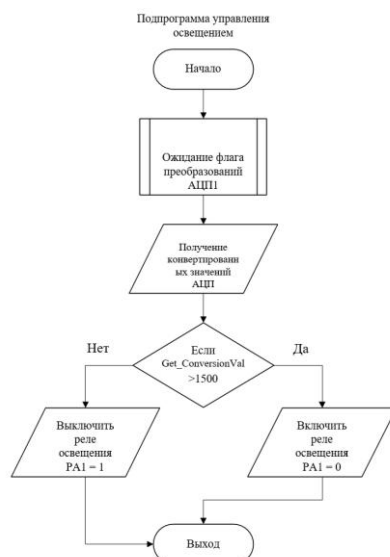


Рисунок 2.25 – Схема алгоритма опроса датчика фоторезистора для включения освещения

На рисунке 2.26 продемонстрирована схема алгоритма опроса датчиков холла для отправки значений по MQTT. Работа алгоритма заключается в следующем: получение входных значений с PB4 и PB5. Далее идёт проверка условия PB5=1: если верно, отправляем топик по MQTT «дверь открыта». Далее идёт проверка условия PB4=1: если верно, отправляем топик по MQTT «окно открыто».

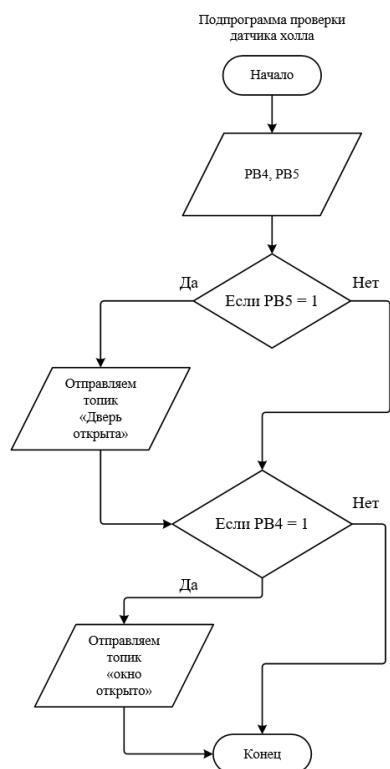


Рисунок 2.26 – Схема алгоритма опроса датчиков холла для отправки значений по MQTT

На рисунке 2.27 продемонстрирована схема алгоритма управления реле обогрева по запросу MQTT. Работа алгоритма заключается в следующем: получение топика управления обогревом. Если условие верно включается реле обогрева и отправляется топик по MQTT «обогрев включён». Иначе выключается реле обогрева

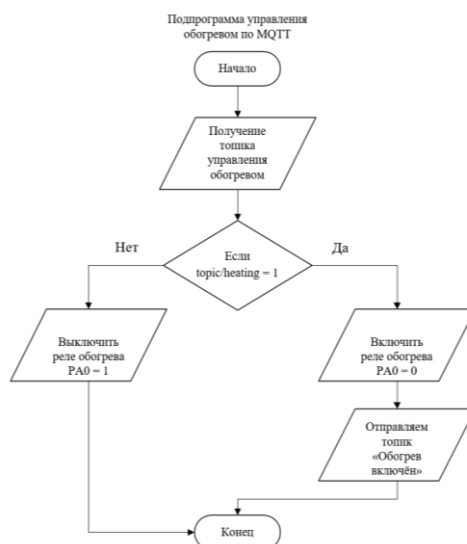


Рисунок 2.27 – Схема алгоритма управления реле обогревом по запросу MQTT

На рисунке 2.28 продемонстрирована схема алгоритма управления реле вентилятора по запросу MQTT. Работа алгоритма заключается в следующем: получение топика управления обогревом. Если условие верно включается реле вентилятор и отправляется топик по MQTT «вентилятор включён». Иначе выключается реле вентилятора.

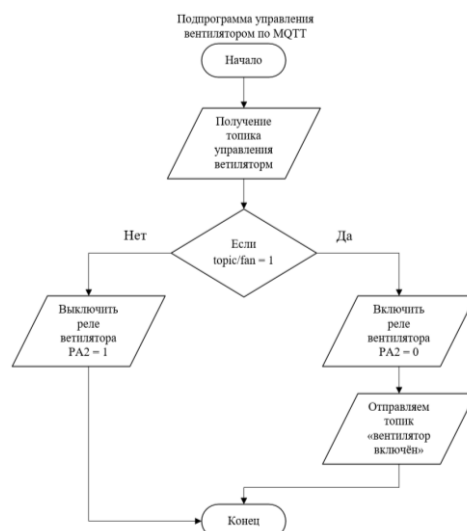


Рисунок 2.28 – Схема алгоритма управления реле вентилятора по запросу MQTT

На рисунке 2.29 продемонстрирована схема алгоритма работы датчика света для отправки значений по MQTT. Работа алгоритма заключается в следующем ожидании флага преобразований АЦП, получение конвертированных значений. Если Get_ConversionVal больше 1500, то отправляется топик «нет освещения».

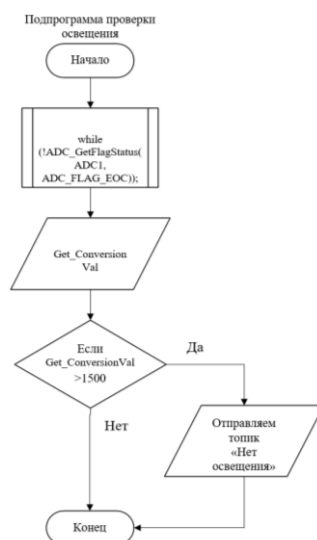


Рисунок 2.29 – Схема алгоритма работы датчика света для отправки значений по MQTT

На рисунке 2.30 продемонстрирована схема алгоритма управления реле освещения по запросу MQTT. Работа алгоритма заключается в следующем получении топика управления освещением. Если условие верно включается реле освещения и отправляется топик по MQTT «освещение включено». Иначе выключается реле освещения.

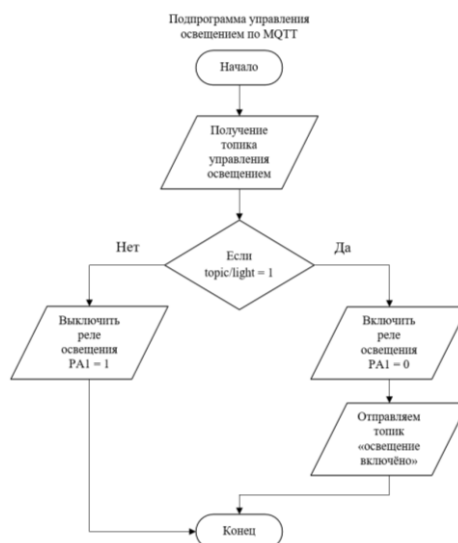


Рисунок 2.30 – Схема алгоритма управления реле освещения по запросу MQTT

2.7 Вывод

В проектирование системы управления объектами умного дома. Произведён обзор микроконтроллеров и обоснование выбора CN32V307VCT6, так как он имеет перспективную архитектуру RISC-V с открытым исходным кодом. Это позволяет создать гибкость системы. Произведён обзор одноплатных компьютеров и обоснования выбора Oranger Pi 5b, так процессор Rockchip 3588S имеет хорошую производительность в системах искусственного интеллекта (6 TOPS). Произведена установка и настройка Home Assistant на одноплатный компьютер Orange Pi 5b. Разработан алгоритм взаимодействия периферийных устройств на базе микроконтроллера CN32V307VCT6.

В третьей главе требуется разработать ПО для микроконтроллера CN32V307VCT6.

3 Реализация системы управления объектами умного дома

Для реализации устройства считывания и управления блоком реле на базе микроконтроллера CH32V307VCT6. Требуется подбор комплектующих, подключение и разработка ПО.

3.1 Разработка устройства считывания датчиков и управления блоком реле

3.1.1 Разработка структурной схемы

При разработке структурных схем электронных систем применяют метод синтеза, в котором разрабатывается «дерево» функций системы. Была разработана структурная схема управления датчиками и блоком реле на базе микроконтроллера RISC-V. Представленная на рисунке 3.1

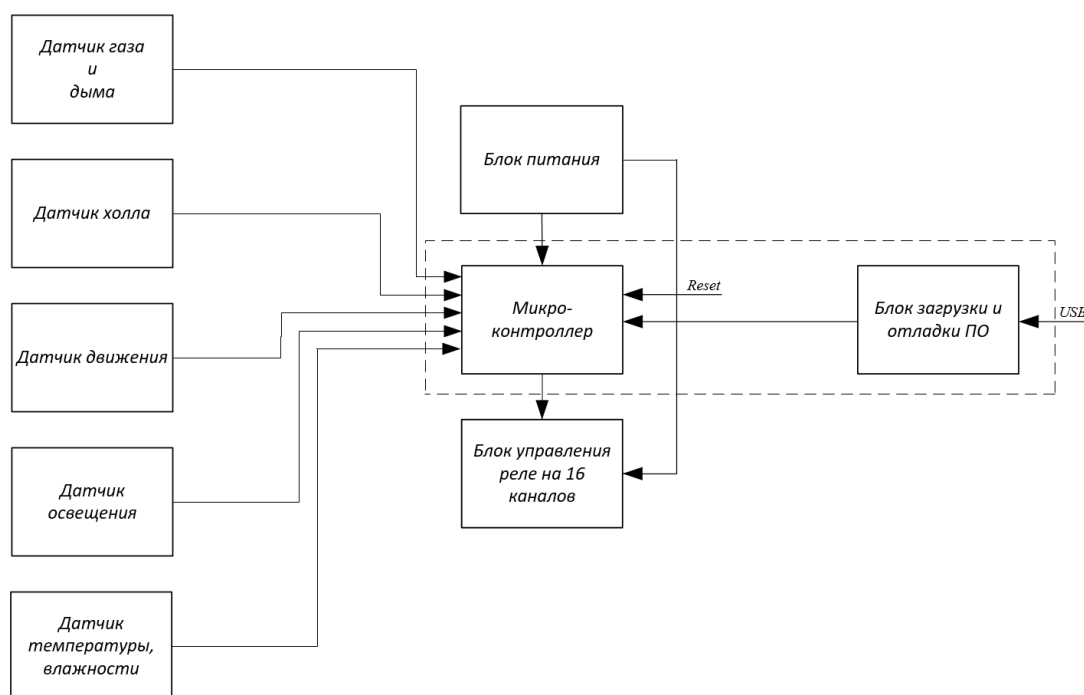


Рисунок 3.1 – Структурная схема управления датчиками и блоком реле на базе микроконтроллера RISC-V.

3.1.2 Разработка аппаратной части

На базе структурной схемы была разработана макетная схема, представленная на рисунке 3.

Для проектирования системы управления датчиками на базе микроконтроллера RISC-V была выбрана отладочная плата CH32V307VCT6– EVT - R1, представленная на рисунке 3.2.

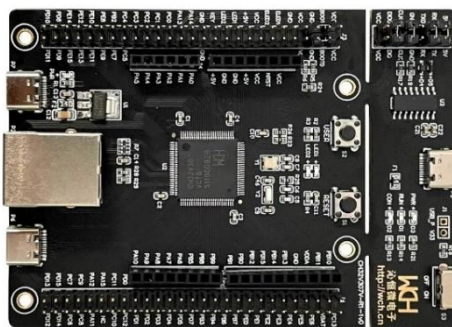


Рисунок 3.2 – Отладочная плата CH32V307VCT6 – EVT - R1

3.1.3 Подбор датчиков

Для дальнейшей постройки системы управления датчиками на базе микроконтроллера RISC-V CH32V307VCT6, были подобраны следующие датчики, представленные на рисунке 3.3.

- датчик газа и дыма MQ-2,
- датчик холла A3144,
- датчик движения HC-SR501,
- датчик освещения, фоторезист,
- датчик температуры, влажности и абсолютного давления BME280,
- блок реле на 16 каналов, 10A 250V.

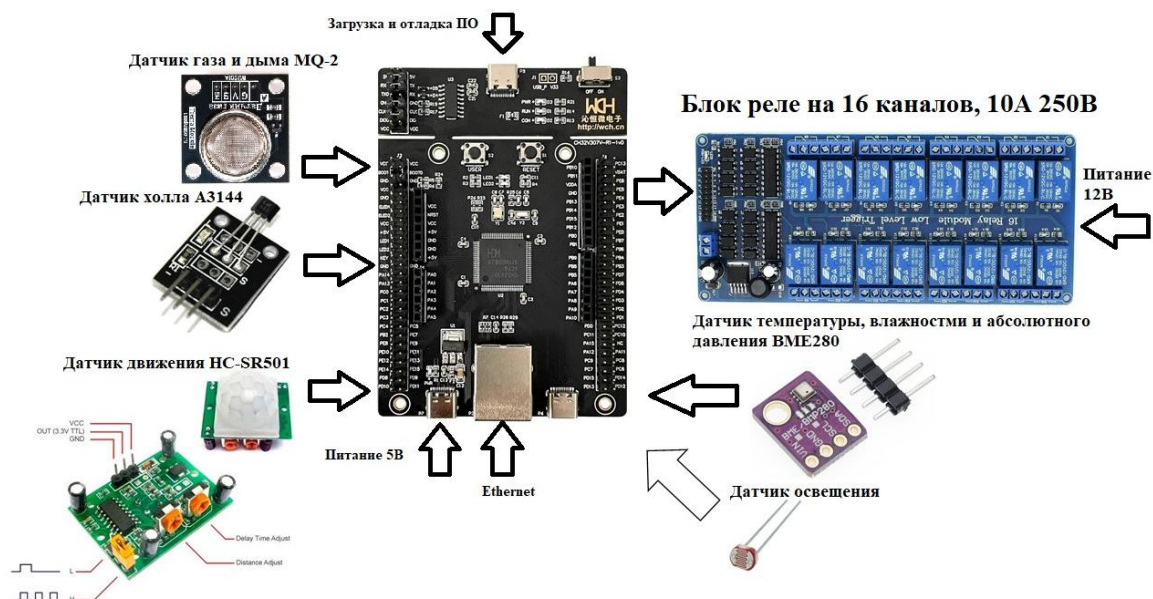


Рисунок 3.3 – Схема взаимодействия датчиков и блока реле

Датчик газа и дыма MQ-2 – отвечает за утечку газа или задымлённость, подключается к плате МК. Отправляет аналоговый сигнал в зависимости от превышения допустимых значений. Устанавливается на потолок. Представленный на рисунке 3.4.



Рисунок 3.4 – Датчик газа и дыма MQ-2

Датчики холла A3144 – отвечают за открытие или закрытие окон и дверей, подключаются к плате МК. Отправляют цифровой сигнал, 1 – закрыто или 0 – открыто. Устанавливаются на дверную раму и окна, в зависимости от количества. Представленный на рисунке 3.5.



Рисунок 3.5 – Датчики холла A3144

Датчик движения HC-SR501 – отвечает за нахождение людей или животных в комнате, подключается к плате МК. Имеет настройку чувствительности и время задержки для правильной работы. Отправляют цифровой сигнал на МК. Устанавливается на стену. Представленный на рисунке 3.6.



Рисунок 3.6 – Датчик движения HC-SR501

Датчик температуры, влажности и абсолютного давления BME280 – отвечает за температуру, влажность и абсолютное давление, подключаются к плате МК. Отправляет данные по интерфейсу I2C. Представленный на рисунке 3.7.



Рисунок 3.7 – Датчик температуры, влажности и абсолютного давления BME280

3.1.4 Подбор блока электромеханических реле

Для управления освещением и электроприборами был выбран готовый блок электромеханических реле на 16 каналов, каждое реле рассчитано на 250В и 10А. Требуется дополнительное питание на 12 вольт (для работы катушек индуктивности реле) и заземление с МК. Каждому реле требуется отдельное подключение от сети 220 вольт. Микроконтроллер отправляет TTL сигнал на 5 вольт на конкретный пин, тем самым обеспечивает управление питанием сети. Представленный на рисунке 3.8.

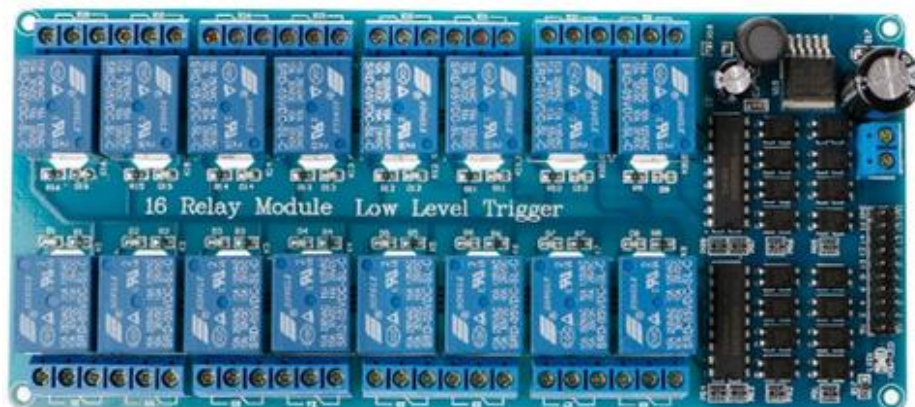


Рисунок 3.8 – Блок электромеханических реле на 16 каналов

Для системы управления умным дом была разработано программное обеспечение, которое считывает показания с датчиков: датчик холла, датчик BMP280, датчик света (фоторезистор) и датчика задымления. На показаниях данных датчиков управляем блоком реле чтобы включать и выключать объекты в зависимости от заданных условий. Для корректной работы микроконтроллера нужно задать правильную конфигурацию.

3.2 Разработка ПО

Для написания ПО микроконтроллера требуется среда разработки MounRiver Studio[23], представленная на рисунке 3.9.

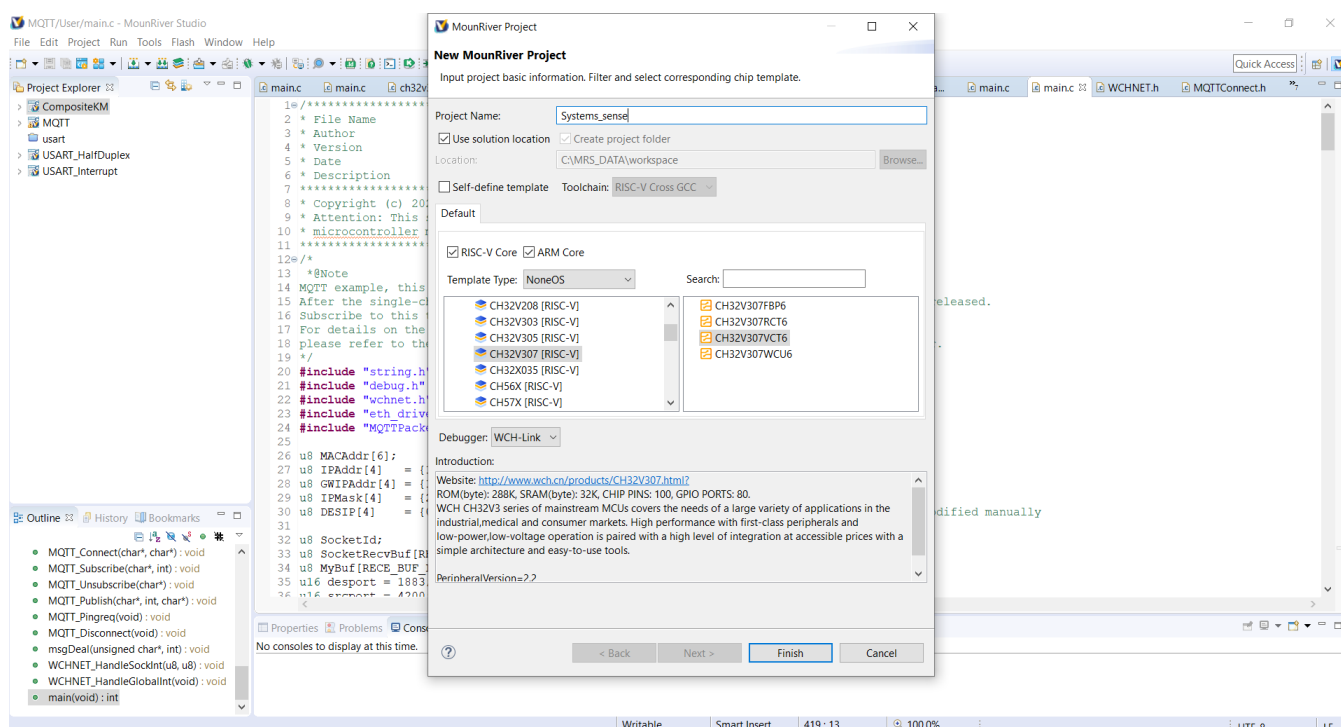


Рисунок 3.9 – Среда разработки MounRiver Studio

Производитель WCH предлагает собственную среду разработки на базе Eclipse, скачать которую можно на официальном сайте. Пакет очень похож на CubeIDE, имея в составе компилятор, утилиту для программирования устройства.

Требуется создать новую программу для этого нужно создать File - > New - > MounRiver Project. В открывшемся окне нужно выбрать Project Name и микроконтроллер, с которым будем работать. В нашем случае проект будет называться Device_sensor_rele. После выполненных действий нажимаем Finish. Как показано на рисунке 3.10.

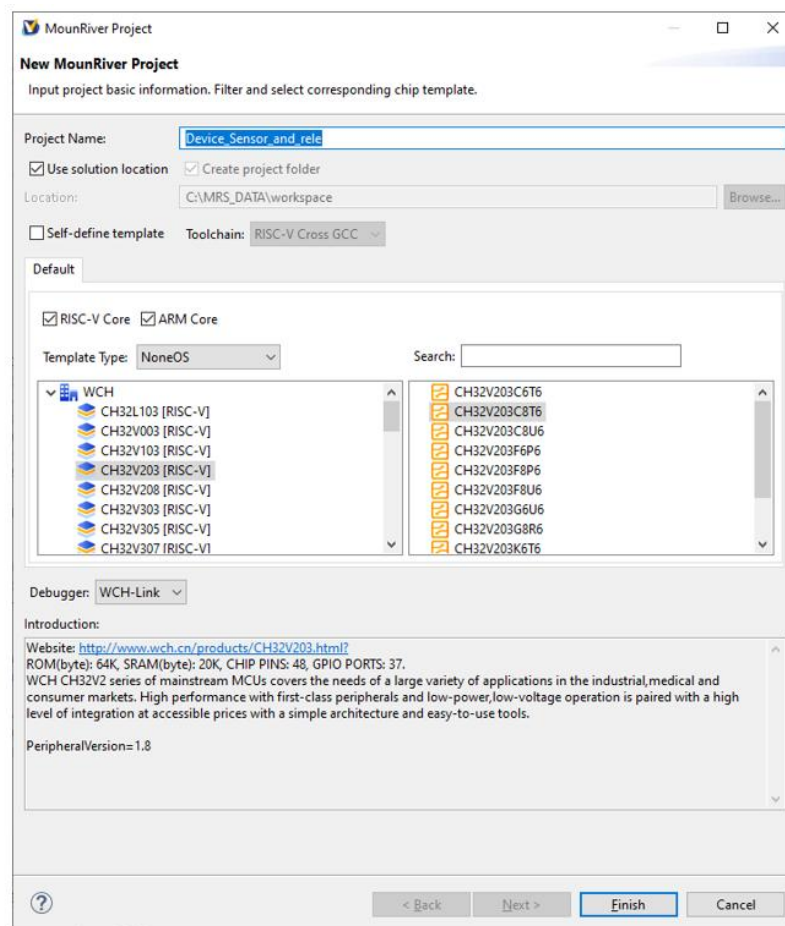


Рисунок 3.10 – Создание проекта в MounRiver Studio

Для запуска микроконтроллера нужно написать конфигурацию для правильной работы устройства. По умолчанию созданного проекта есть файлы `main.c`, `ch32v20x_it.c`, `ch32v20x_it.h`, `system_ch32v20x.c`, `system_ch32v20x.h` и `ch32v20x_conf.h`. В них изначально установлены базовые настройки микроконтроллера. В файле `main.c` написаны базовые функции `SystemCoreClockUpdate`, `NVIC_PriorityGroupConfig`, `Delay_Init` и `USART_Printf_Init(115200)`. В программировании используется язык Си, можно использовать регистры или библиотеку (SPL) Standard Peripheral Library.

Функция `SystemCoreClockUpdate` используется для запуска тактирования периферии, интерфейсов и шин микроконтроллера CH32V203C8T6. В нашем случае работает микроконтроллер работает на частоте 96 МГц. В файле `system_ch32v20x.c` нужно удалить комментарии строки «`#define`

`SYSCLK_FREQ_96MHz_HSE 96000000`». Тем самым задаём частоту. Ниже представлен листинг функции.

```
void SystemCoreClockUpdate (void)
{
    uint32_t tmp = 0, pllmull = 0, pllsource = 0, Pl1_6_5 = 0;
    tmp = RCC->CFGR0 & RCC_SWS;
    switch (tmp)
    {
        case 0x00:
            SystemCoreClock = HSI_VALUE;
            break;
        case 0x04:
            SystemCoreClock = HSE_VALUE;
            break;
        case 0x08:
            pllmull = RCC->CFGR0 & RCC_PLLMULL;
            pllsource = RCC->CFGR0 & RCC_PLLSRC;
            pllmull = ( pllmull >> 18) + 2;
            if(pllmull == 17) pllmull = 18;
            if (pllsource == 0x00)
            {
                if(EXTEN->EXTEN_CTR & EXTEN_PLL_HSI_PRE){
                    SystemCoreClock = HSI_VALUE * pllmull;
                }
                else{
                    SystemCoreClock = (HSI_VALUE >> 1) * pllmull;
                }
            }
            else
            {
                if ((RCC->CFGR0 & RCC_PLLXTPRE) != (uint32_t)RESET)
                {
                    #else
                        SystemCoreClock = (HSE_VALUE >> 1) * pllmull;
                    #endif
            }
    }
```

```

        }
        else
        {
#else
            SystemCoreClock = HSE_VALUE * pllmull;
#endif
        }
    }

    if(Pll_6_5 == 1) SystemCoreClock = (SystemCoreClock / 2);
    break;
    default:
        SystemCoreClock = HSI_VALUE;
        break;
    }

    tmp = AHBPrescTable[((RCC->CFGR0 & RCC_HPRE) >> 4)];
    SystemCoreClock >>= tmp;
}

```

Функция SetSysClockTo96_HSE используется для работы микроконтроллера на частоте 96 МГц с тактированием от внешнего кварцевого или керамического генератора. Ниже представлен листинг функции

```

#elif defined SYSCLK_FREQ_96MHz_HSE
static void SetSysClockTo96_HSE(void) {
    __IO uint32_t StartUpCounter = 0, HSEStatus = 0;
    RCC->CTLR |= ((uint32_t)RCC_HSEON);
    do{
        HSEStatus = RCC->CTLR & RCC_HSERDY;
        StartUpCounter++;
    } while((HSEStatus == 0) && (StartUpCounter !=
HSE_STARTUP_TIMEOUT));
    if ((RCC->CTLR & RCC_HSERDY) != RESET) {
        HSEStatus = (uint32_t)0x01;
    }
    else{
        HSEStatus = (uint32_t)0x00;
    }
}

```

```

if (HSEStatus == (uint32_t)0x01){
    /* HCLK = SYSCLK */
    RCC->CFGR0 |= (uint32_t)RCC_HPRE_DIV1;
    /* PCLK2 = HCLK */
    RCC->CFGR0 |= (uint32_t)RCC_PPRE2_DIV1;
    /* PCLK1 = HCLK */
    RCC->CFGR0 |= (uint32_t)RCC_PPRE1_DIV2;
    /* CH32V20x_D6-PLL configuration: PLLCLK = HSE * 12 =
96 MHz (HSE=8MHZ)
        * CH32V20x_D8-PLL configuration: PLLCLK = HSE/4 * 12 =
96 MHz (HSE=32MHZ)
        */
    RCC->CFGR0      &=      (uint32_t)((uint32_t)~(RCC_PLLSRC      |
RCC_PLLXTPRE      |RCC_PLLMULL));
    RCC->CFGR0      |=      (uint32_t)(RCC_PLLSRC_HSE      |
RCC_PLLXTPRE_HSE | RCC_PLLMULL12);
    RCC->CTLR |= RCC_PLLON;
    while((RCC->CTLR & RCC_PLLRDY) == 0)
    /* Select PLL as system clock source */
    RCC->CFGR0 &= (uint32_t)((uint32_t)~(RCC_SW));
    RCC->CFGR0 |= (uint32_t)RCC_SW_PLL;
    /* Wait till PLL is used as system clock source */
    while      ((RCC->CFGR0      &      (uint32_t)RCC_SWS)      !=
(uint32_t)0x08)

```

Описание структуры тактирования микроконтроллера

Высокоскоростной тактовый сигнал (HSI/HSE) HSI — это высокоскоростной тактовый сигнал, генерируемый RC-генератором с частотой 8 МГц в системе. Генератор HSI RC может обеспечивать системную тактовую частоту независимо от какого-либо внешнего устройства. Время запуска очень короткое, но точность тактовой частоты низкая. HSI включается/выключается установкой бита HSION в регистре RCC_CTLR. Бит HSIRDY указывает, стабилен ли RC-генератор HSI. По умолчанию HSION и HSIRDY установлены на 1 (рекомендуется не отключать). Если бит HSIRDYIE в регистре RCC_INTR установлен, генерируется соответствующее прерывание.

Заводская калибровка: разница в производственном процессе может привести к тому, что частота колебаний RC каждого чипа будет разной, поэтому калибровка HSI выполняется для каждого чипа до того, как чип будет доставлен с завода. После сброса системы значение заводской калибровки загружается в биты HSICAL[7:0] регистра RCC_CTLR.

Пользовательская настройка: в зависимости от различных напряжений или температур окружающей среды прикладная программа может регулировать частоту HSI с помощью битов HSITRIM[4:0] в регистре RCC_CTLR. В случае выхода из строя кварцевого генератора HSE часы HSI используются в качестве резервного источника синхронизации.

HSE — это высокоскоростной внешний тактовый сигнал, включающий генерацию внешнего кристаллического/керамического резонатора или внешний высокоскоростной тактовый вход.

Внешний кварцевый/керамический резонатор (кристалл HSE): внешний генератор с частотой от 3 до 25 МГц обеспечивает более точный источник синхронизации для системы. Кристалл HSE можно включить/выключить, установив бит HSEON в регистре RCC_CTLR. Бит HSERDY указывает, стабильны ли колебания кристалла HSE. Тактовый сигнал не сбрасывается до тех пор, пока бит HSERDY не будет аппаратно установлен в 1. Если бит HSERDYIE в регистре RCC_INTR установлен, может быть сгенерировано соответствующее прерывание.

Низкоскоростной тактовый сигнал (LSI/LSE)

LSI — это низкоскоростной тактовый сигнал, генерируемый внутренним RC-генератором системы. Он может продолжать работать в режимах остановки и ожидания и обеспечивать опорное значение для часов реального времени, независимого сторожевого таймера и устройства пробуждения. LSI можно включить/выключить, установив бит LSION в регистре RCC_RSTSCKR, а затем он проверяет, стабильно ли колебание RC LSI, запрашивая бит LSIRDY. Тактовый сигнал не сбрасывается до тех пор, пока бит LSIRDY не будет аппаратно установлен в 1. Если бит LSIRDYIE в реги-

стре `RCC_INTR` установлен, может быть сгенерировано соответствующее прерывание.

`LSE` — это внешний низкоскоростной тактовый сигнал, включающий генерацию внешнего кристаллического/керамического резонатора или внешний низкоскоростной тактовый вход. Он обеспечивает маломощный, но высокоточный источник синхронизации для часов реального времени или других функций синхронизации.

Внешний кристаллический/керамический резонатор (кристалл `LSE`): внешний низкоскоростной генератор с частотой 32,768 кГц. `LSE` можно включить/выключить, установив бит `LSEON` в регистре `RCC_BDCTLR`. Бит `LSERDY` указывает, стабильны ли колебания кристалла `LSE`. Тактовый сигнал не сбрасывается до тех пор, пока бит `LSERDY` не будет аппаратно установлен в 1. Если бит `LSERDYIE` в регистре `RCC_INTR` установлен, может быть сгенерировано соответствующее прерывание.

Синхронизация шины/периферийных устройств

Системная синхронизация (`SYSCLK`) источник системной синхронизации, биты `SW[1:0]` в регистре `RCC_CFGR0`. Биты `SWS[1:0]` указывают текущий статус источника системной синхронизации.

После сброса контроллера тактовая частота `HSI` выбирается в качестве источника системной тактовой частоты по умолчанию. Переключение с одного источника синхронизации на другой происходит только в том случае, если целевой источник синхронизации готов.

Периферийная тактовая частота шины `AHB/APB1/APB2` (`HCLK/PCLK1/PCLK2`)

Путем настройки битов `HPRE[3:0]`, `PPRE1[2:0]` и `PPRE2[2:0]` в регистре `RCC_CFGR0` синхронизация шин `AHB`, `APB1` и `APB2` можно настроить соответственно. Эти тактовые частоты шины определяют опорную тактовую частоту доступа периферийных интерфейсов, установленных под ними. Прикладная программа может настраивать различные значения для снижения энергопотребления некоторых периферийных устройств. Различные перифе-

рийные модули могут быть сброшены битами в регистрах RCC_AHBRSR, RCC_APB1PRSTR и RCC_APB2PRSTR, чтобы вернуть их в исходное состояние.

Установив биты в регистрах RCC_AHBPCENR, RCC_APB1PCENR и RCC_APB2PCENR, интерфейс синхронизации связи различных периферийных модулей можно включать или отключать отдельно. Чтобы использовать периферийное устройство, сначала включите соответствующий тактовый бит для доступа к его регистру.

На рисунке 3.11 продемонстрирована схема тактирования микроконтроллера серии CH32V307

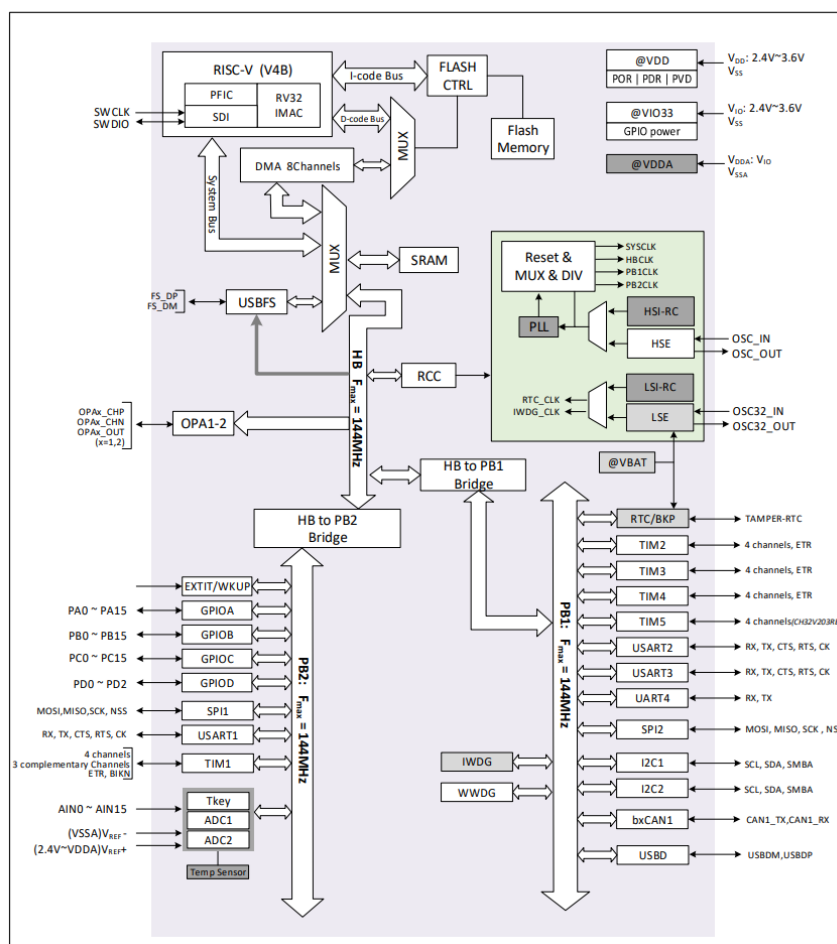


Рисунок 3.11 – Схема тактирования микроконтроллера серии CH32V307

Функция Delay_Init(); используется для запуска инициализации функции задержки. Требуется использовать функцию Delay_Ms работающий на SysTick. Ниже представлен листинг функции

```

void Delay_Ms(uint32_t n)
{
    uint32_t i;
    SysTick->SR &= ~(1 << 0);
    i = (uint32_t)n * p_ms;
    SysTick->CMP = i;
    SysTick->CTLR |= (1 << 4);
    SysTick->CTLR |= (1 << 5) | (1 << 0);
    while((SysTick->SR & (1 << 0)) != (1 << 0));
    SysTick->CTLR &= ~(1 << 0);
}

```

Функция `USART_Printf_Init(115200)`; используется для инициализации интерфейса USART (Universal Synchronous/Asynchronous Receiver/Transmitter) с заданной скоростью передачи данных (в данном случае 115200 бод). Это позволяет микроконтроллеру обмениваться данными с другими устройствами компьютером, модулем связи или другим микроконтроллером через последовательный интерфейс. В нашем случае отладку. Ниже описаны основные шаги инициализации, листинг функции и пояснения.

Основные шаги инициализации USART

- Включение тактирования для периферии: Подача тактового сигнала на периферийное устройство USART и необходимые порты ввода-вывода.
- Настройка GPIO для USART: Конфигурация выводов микроконтроллера для работы в режиме альтернативной функции USART (TX и RX).
- Настройка параметров USART: Установка параметров, таких как скорость передачи данных (baud rate), количество бит данных, стоп-биты, паритет и режим работы (передача, прием).
- Включение USART: Включение самого модуля USART для начала передачи и приема данных.

```

USART_Printf_Init(115200); {
    // Включение тактирования для USART1 и порта GPIOA

```

```

        RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1
RCC_APB2Periph_GPIOA, ENABLE);
    // Конфигурация выводов GPIOA для USART1
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9; // PA9 (TX)
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; // Альтерна-
тивная функция Push-Pull
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10; // PA10 (RX)
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; // Вход
с подтяжкой
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    // Конфигурация параметров USART1
    USART_InitTypeDef USART_InitStructure;
    USART_InitStructure.USART_BaudRate = baudrate; // Скорость пере-
дачи данных
    USART_InitStructure.USART_WordLength = USART_WordLength_8b; //
8 бит данных
    USART_InitStructure.USART_StopBits = USART_StopBits_1; // 1
стоп-бит
    USART_InitStructure.USART_Parity = USART_Parity_No; // Без па-
ритета
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None; // Без аппаратного контроля
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
// Режимы передачи и приема
    USART_Init(USART1, &USART_InitStructure);
    // Включение USART1
    USART_Cmd(USART1, ENABLE);
}

```

Пояснения к листингу функции USART_Printf_Init(115200)

– включение тактирования. RCC_APB2PeriphClockCmd
(RCC_APB2Periph_USART1 | RCC_APB2Periph_GPIOA, ENABLE);
Включает тактирование для периферии USART1 и GPIOA,

- настройка GPIO: настраивает выводы PA9 и PA10 для работы в режиме альтернативной функции для передачи и приема данных соответственно,
- конфигурация параметров USART. устанавливает параметры USART, такие как скорость передачи данных, длина слова, стоп-биты, паритет и режимы работы,
- включение USART: включает модуль USART для начала работы.

Функция `USART_Printf_Init(115200)`; выполняет инициализацию интерфейса USART с заданной скоростью передачи данных, обеспечивая настройку периферийных устройств и выводов микроконтроллера. Это позволяет использовать USART для передачи и приема данных, что необходимо для коммуникации с другими устройствами через последовательный интерфейс.

Для структуризации проекта создаём файл `adc.c`, в папке `user` нужно создать файл. В контекстном меню нужно выбрать `New->Source File`. Создаём файл `adc.c` в нём будут функции для инициализации `void ADC_Function_Init`, для калибровки значений `Get_ConversionVal`, для управления светом `void light`.

Функция `ADC_Function_Init` используется для инициализации аналого-во-цифрового преобразователя АЦП микроконтроллера. АЦП преобразует аналоговые сигналы в цифровые значения, которые можно обработать программно. Инициализация АЦП включает настройку различных параметров, таких как каналы, скорость преобразования, разрешение и другие настройки, необходимые для правильной работы АЦП. Это необходимо для работы с датчиками, потенциометрами и другими аналоговыми устройствами, где требуется точное измерение аналоговых величин. В нашем случае датчик света (фоторезист). Ниже описаны основные шаги инициализации, листинг функции и пояснения.

Основные шаги инициализации АЦП

- включение тактирования для АЦП: Подача тактового сигнала на модуль АЦП и необходимые порты ввода-вывода,
- настройка GPIO для входов АЦП: Конфигурация выводов микроконтроллера для работы в режиме аналогового ввода,
- настройка параметров АЦП: Установка параметров, таких как каналы, разрешение, режимы преобразования, скорость выборки и так далее,
- калибровка и включение АЦП: Выполнение калибровки и включение АЦП для начала преобразований.

```

void ADC_Function_Init(void) {
    // Включение тактирования для ADC1 и порта GPIOA
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1|
RCC_APB2Periph_GPIOA, ENABLE);

    // Настройка GPIOA для аналогового ввода PA0
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;    // PA0
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN; // Аналоговый
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    // Настройка параметров ADC1
    ADC_InitTypeDef ADC_InitStructure;
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;    // Незави-
симый режим
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;    // Отключение
сканирующего режима
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;    // Непре-
рывный режим
    ADC_InitStructure.ADC_ExternalTrigConv =
ADC_ExternalTrigConv_None;    // Без внешнего триггера
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;    // Вы-
равнивание данных вправо
    ADC_InitStructure.ADC_NbrOfChannel = 1; // Количество каналов
    ADC_Init(ADC1, &ADC_InitStructure);

    // Настройка канала для PA0
    ADC-RegularChannelConfig(ADC1,          ADC_Channel_0,          1,
ADC_SampleTime_55Cycles5);

```

```

// Включение ADC1
ADC_Cmd(ADC1, ENABLE);
// Калибровка ADC1
ADC_ResetCalibration(ADC1);
while(ADC_GetResetCalibrationStatus(ADC1)); // Ожидание за-
вершения сброса калибровки
ADC_StartCalibration(ADC1);
while(ADC_GetCalibrationStatus(ADC1)); // Ожидание заверше-
ния калибровки
// Запуск преобразования
ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}

```

Пояснения к листингу функции ADC_Function_Init:

- включение тактирования:
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1
RCC_APB2Periph_GPIOA, ENABLE); Включает тактирование для периферии
ADC1 и порта GPIOA,
- настройка GPIO: GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0;
GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AIN; Настраивает вывод PA0 для
работы в режиме аналогового ввода,
- конфигурация параметров АЦП: ADC_InitStruct.ADC_Mode =
ADC_Mode_Independent; ADC_InitStruct.ADC_ScanConvMode = DISABLE;
ADC_InitStruct.ADC_ContinuousConvMode = ENABLE;
ADC_InitStruct.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
Устанавливает параметры АЦП, такие как режим работы, количество
каналов, выравнивание данных и прочее,
- настройка канала: ADC_RegularChannelConfig(ADC1,
ADC_Channel_0, 1, ADC_SampleTime_55Cycles5); Конфигурирует канал 0
для ввода с выводом PA0 и задает время выборки,
- включение и калибровка АЦП: ADC_Cmd(ADC1, ENABLE);
ADC_ResetCalibration(ADC1); while(ADC_GetResetCalibrationStatus(ADC1));

```
ADC_StartCalibration(ADC1);          while(ADC_GetCalibrationStatus(ADC1));
```

Включает АЦП и выполняет его калибровку,

– запуск преобразования: `ADC_SoftwareStartConvCmd(ADC1, ENABLE);` Запускает непрерывное преобразование аналоговых сигналов в цифровые.

Функция `GPIO_Toggle_INIT();` предназначена для инициализации и настройки восьми GPIO (General Purpose Input/Output) пинов для их последующего использования в режиме переключения (toggle). Эта функция подготавливает пины к работе в режиме выхода, что позволяет управлять их состоянием (включать и выключать) программно. В нашем случае это нужно для управления блоком реле. Ниже представлен листинг функции.

```
void GPIO_Toggle_INIT(void)
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); //
разрешаем тактирование порта A
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 |
GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 |
GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //
устанавливаем режим Push-pull output
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //
скорость работы порта
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

Функция `GPIO_Toggle_out();` предназначена для запуска инициализации пинов PB4 и PB5. Которые будут работать на ввод данных. В нашем случае для считываний показаний с датчиков холла. Ниже представлен листинг функции.

```
void GPIO_Toggle_out(void) {
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); //
разрешаем тактирование порта B
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5; //
какие пины должны работать в данном порте
```

```

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD; // устанавли-
ваем режим Pull-down input
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; // ско-
рость работы порта
    GPIO_Init(GPIOB, &GPIO_InitStructure);
}

```

Функция `I2C_Init(100000, 0x38)`; используется для инициализации интерфейса I²C (Inter-Integrated Circuit) с заданной скоростью передачи данных (в данном случае 100 кГц) и с заданным адресом устройства (в данном случае 0x38). I²C — это широко используемый протокол связи, который позволяет микроконтроллерам общаться с периферийными устройствами (например, датчиками, памятью и другими микросхемами) через двухпроводную шину (*SDA* и *SCL*). Это необходимо для обеспечения корректной работы шины I²C и обмена данными с другими устройствами. Правильная настройка параметров I²C позволяет гарантировать надежную и эффективную коммуникацию между микроконтроллером и периферийными устройствами. В нашем случае она нужна для чтобы считывать показания с датчика температуры, влажности и атмосферного давления. Ниже описаны основные шаги инициализации, листинг функции и пояснения.

Основные шаги инициализации I²C

- Включение тактирования для I²C и GPIO: Включение тактирования для модуля I²C и портов GPIO, необходимых для работы шины I²C.
- Настройка GPIO для I²C: Конфигурация выводов микроконтроллера для работы в режиме I²C (*SDA* и *SCL*).
- Настройка параметров I²C: Установка параметров, таких как скорость передачи данных (baud rate), адрес устройства, режим работы (master/slave), и других необходимых настроек.

- Включение I²C: Включение модуля I²C для начала работы.

```

void IIC_Init(uint32_t clockSpeed, uint8_t ownAddress) {
    // Включение тактирования для I2C1 и порта GPIOB

```

```

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
// Настройка GPIOB для I2C1 (PB6 - SCL, PB7 - SDA)
GPIO_InitTypeDef GPIO_InitStructure;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7; // SCL и
SDA

GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD; //
Альтернативная функция Open-Drain
GPIO_Init(GPIOB, &GPIO_InitStructure);
// Настройка параметров I2C1
I2C_InitTypeDef I2C_InitStructure;
I2C_InitStructure.I2C_ClockSpeed = clockSpeed; // Скорость
передачи данных
I2C_InitStructure.I2C_Mode = I2C_Mode_I2C; // Режим I2C
I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2; //
Коэффициент заполнения
I2C_InitStructure.I2C_OwnAddress1 = ownAddress; // Собственный
адрес устройства
I2C_InitStructure.I2C_Ack = I2C_Ack_Enable; // Включение
подтверждения (ACK)
I2C_InitStructure.I2C_AcknowledgedAddress =
I2C_AcknowledgedAddress_7bit; // 7-битный адрес
I2C_Init(I2C1, &I2C_InitStructure);
// Включение I2C1
I2C_Cmd(I2C1, ENABLE);

```

Пояснения к листингу функции

– включение тактирования:
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE); Включает
тактирование для периферии I²C1 и порта GPIOB,

– настройка GPIO. GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 |
GPIO_Pin_7; GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD; GPIO_Init(GPIOB,

&GPIO_InitStruct); Настраивает выходы PB6 (SCL) и PB7 (SDA) для работы в режиме альтернативной функции Open-Drain для шины I²C,

- конфигурация параметров I²C. I2C_InitStruct.I2C_ClockSpeed = clockSpeed; I2C_InitStruct.I2C_Mode = I2C_Mode_I2C; I2C_InitStruct.I2C_DutyCycle = I2C_DutyCycle_2; I2C_InitStruct.I2C_OwnAddress1 = ownAddress; I2C_InitStruct.I2C_Ack = I2C_Ack_Enable; I2C_InitStruct.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit; I2C_Init(I2C1, &I2C_InitStruct); Устанавливает параметры I²C, такие как скорость передачи данных, режим работы, собственный адрес устройства и режим подтверждения,

- включение I²C. I2C_Cmd(I2C1, ENABLE); Включает модуль I²C для начала работы.

Функция `rele_Init()`; предназначена для конфигурации пинов управления реле к логической 1, так как пины нашего блок реле управляется логическим 0. Для удобства программирования были созданы макросы, которые содержат базовую установку. Ниже представлен листинг функции.

```
/* Макросы для установки пинов в лог. 1 */
#define d0_set GPIOA->BSHR |= GPIO_Pin_0;
#define d1_set GPIOA->BSHR |= GPIO_Pin_1;
#define d2_set GPIOA->BSHR |= GPIO_Pin_2;
#define d3_set GPIOA->BSHR |= GPIO_Pin_3;
#define d4_set GPIOA->BSHR |= GPIO_Pin_4;
#define d5_set GPIOA->BSHR |= GPIO_Pin_5;
#define d6_set GPIOA->BSHR |= GPIO_Pin_6;
#define d7_set GPIOA->BSHR |= GPIO_Pin_7;
/* Макросы для установки пинов в лог. 0 */
#define d0_reset GPIOA->BCR |= GPIO_Pin_0;
#define d1_reset GPIOA->BCR |= GPIO_Pin_1;
#define d2_reset GPIOA->BCR |= GPIO_Pin_2;
#define d3_reset GPIOA->BCR |= GPIO_Pin_3;
#define d4_reset GPIOA->BCR |= GPIO_Pin_4;
```

```

#define d5_reset GPIOA->BCR |= GPIO_Pin_5;
#define d6_reset GPIOA->BCR |= GPIO_Pin_6;
#define d7_reset GPIOA->BCR |= GPIO_Pin_7;

void rele_Init(void) // функция инициализации блока реле
{
    d0_set;  d1_set;  d2_set;  d3_set;  d4_set;  d5_set;  d6_set;
d7_set;
}

```

Требуется создать 2 файла bmp280.h и bmp280.c, в папке user нужно создать 2 файла. В контекстном меню нужно выбрать New->Source File. Создаём файл bmp280.c тоже самое для bmp280.h. В файле bmp280.h будут храниться макросы для регистров калибровки, регистров конфигурации датчика bmp280, для адреса устройства и регистров фильтрации значений. В файле bmp280.c, хранятся функции для инициализации void BMP280_Init, для конвертации значений функция void bmp280Convert, для отправки 1 байта датчику по интерфейсу i2c функция void bmp280WriteByte.

Функция BMP280_Init(); выполняет калибровку датчика отправляя короткие команды функцией bmp280ReadShort. Для установки конфигурации датчика с помощью функции bmp280WriteByte отправляем 1 байт, то что находится в макросах BMP280_REG_CONFIG, BMP280_CONFIG, BMP280_REG_CONTROL и BMP280_MEAS. Ниже представлен листинг функции.

```

void BMP280_Init(void)
{
    dig_T1 = bmp280ReadShort(0x88); //dig_T1
    dig_T2 = bmp280ReadShort(0x8A); //dig_T2
    dig_T3 = bmp280ReadShort(0x8C); //dig_T3
    dig_P1 = bmp280ReadShort(0x8E); //dig_P1
    dig_P2 = bmp280ReadShort(0x90); //dig_P2
    dig_P3 = bmp280ReadShort(0x92); //dig_P3
    dig_P4 = bmp280ReadShort(0x94); //dig_P4
    dig_P5 = bmp280ReadShort(0x96); //dig_P5
}

```

```

    dig_P6 = bmp280ReadShort(0x98); //dig_P6
    dig_P7 = bmp280ReadShort(0x9A); //dig_P7
    dig_P8 = bmp280ReadShort(0x9C); //dig_P8
    dig_P9 = bmp280ReadShort(0x9E); //dig_P9
    bmp280WriteByte(BMP280_REG_CONFIG, BMP280_CONFIG);
    bmp280WriteByte(BMP280_REG_CONTROL, BMP280_MEAS);
}

```

Функция *bmp280WriteByte* предназначена для отправки один байт по интерфейсу *i2c* в датчик *bmp280*. Использует стандартные функции *i2c* интерфейса с подключённой библиотекой *SPL*. Для запуска *i2c*, для отправки 7 битного адреса *i2c* устройства, для отправки данных в датчик. Принцип работы ожидаем флаг готовности. Отправляем адрес устройства и ожидаем флаг готовности. Отправляем данные, ожидаем флаг готовности и останавливаем *i2c* шину. Ниже представлен листинг функции.

```

void bmp280WriteByte(unsigned char address, unsigned char
data)
{
    // запускаем I2C1 и ожидаем флага готовности
    I2C_GenerateSTART(I2C1, ENABLE);
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT))
        // отправляем адрес устройства и ожидаем флага готовности
        I2C_Send7bitAddress(I2C1, BMP280_addr, I2C_Direction_Transmitter);
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED));

    I2C_SendData(I2C1, address);
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED));

    I2C_SendData(I2C1, data);
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED));

    I2C_GenerateSTOP(I2C1, ENABLE);
}

```

```
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_TRANSMITTED
));
```

Функции `bmp280ReadShort` предназначена для чтения коротких значений такие как калибровка, конфигурация и `bmp280ReadLong` предназначена для чтения длинных значений. Они идентичны, чтобы считать значения ожидаем флаг готовности шины i2c, нужно отправить адрес датчика ожидать флаг готовности шины. Далее отправить адрес команды и ожидать готовности шины. Приём данных осуществляется функцией `I2C_ReceiveData` и ожидаем флаг готовности шины. Для `bmp280ReadShort` заполняются переменные `lsb` и `msb`. Для `bmp280ReadLong` заполняются переменные `lsb`, `msb` и `xsb`. После каждой заполненной переменной ожидаем флаг готовности и останавливаем i2c шину. Ниже представлен листинг функции.

```
short bmp280ReadShort(unsigned char address)
{
short msb=0;
short lsb=0;
I2C_AcknowledgeConfig(I2C1,ENABLE);
I2C_GenerateSTART(I2C1,ENABLE);
while( I2C_GetFlagStatus( I2C2, I2C_FLAG_BUSY ) != RESET
I2C_GenerateSTART( I2C2, ENABLE );
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_MODE_SELECT)
I2C_Send7bitAddress(I2C1,BMP280_addr,I2C_Direction_Transmitter);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED));
I2C_SendData(I2C1,address);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_TRANSMITTED));
I2C_GenerateSTART(I2C1,ENABLE);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_MODE_SELECT))
I2C_Send7bitAddress(I2C1,BMP280_addr,I2C_Direction_Receiver);
;
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED));
```

```

while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_RECEIVED)
lsb = I2C_ReceiveData(I2C1);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_RECEIVED)
msb = I2C_ReceiveData(I2C1);
I2C_GenerateSTOP(I2C1,ENABLE);
I2C_AcknowledgeConfig(I2C1,DISABLE);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_RECEIVED)
I2C_ReceiveData(I2C1);
return (msb << 8) | lsb;
}

unsigned long bmp280ReadLong(unsigned char address)
{
unsigned long result=0;
unsigned long msb=0;
unsigned long lsb=0;
unsigned long xsb=0;
I2C_AcknowledgeConfig(I2C1,ENABLE);
I2C_GenerateSTART(I2C1,ENABLE);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_MODE_SELECT))
I2C_Send7bitAddress(I2C1,BMP280_addr,I2C_Direction_Transmitt
er);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_TRANSMITTER_MODE
_SELECTED));
I2C_SendData(I2C1,address);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_TRANSMITTED
));
I2C_GenerateSTART(I2C1,ENABLE);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_MODE_SELECT))
I2C_Send7bitAddress(I2C1,BMP280_addr,I2C_Direction_Receiver)
;
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_RECEIVER_MODE_SE
LECTED));
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_RECEIVED)
msb = I2C_ReceiveData(I2C1);

```

```

while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_RECEIVED)
lsb = I2C_ReceiveData(I2C1);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_RECEIVED)
xsb = I2C_ReceiveData(I2C1);
I2C_GenerateSTOP(I2C1,ENABLE);
I2C_AcknowledgeConfig(I2C1,DISABLE);
while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_BYTE_RECEIVED)
I2C_ReceiveData(I2C1);
result = (msb << 16) | (lsb << 8) | xsb;
return (result >> 4);
}

```

Функция void bmp280Convert предназначена для конвертации данных в температуру и атмосферное давление. В переменные adc_T и adc_P заполняются значениями, полученными с датчика bmp280. С помощью функции bmp280ReadLong. С помощью переменных var1, var2, p и t_fine производит конвертацию значений. В переменные temperature и pressure сохраняем конечные значения. Ниже представлен листинг функции.

```

void bmp280Convert(long* temperature, long* pressure)
{
unsigned long adc_T;
unsigned long adc_P;
adc_T = bmp280ReadLong(BMP280_REG_RESULT_TEMPRERATURE);
adc_P = bmp280ReadLong(BMP280_REG_RESULT_PRESSURE);
double var1, var2, p, t_fine;
var1 = (((double)adc_T)/16384.0 - ((double)dig_T1)/1024.0) *
((double)dig_T2);
var2 = (((double)adc_T)/131072.0 - ((double)dig_T1)/8192.0)
* (((double)adc_T)/131072.0 - ((double) dig_T1)/8192.0)) *
((double)dig_T3);
t_fine = (var1 + var2);
*temperature = (int32_t) (t_fine*10 / 5120.0);
var1 = ((double)t_fine/2.0) - 64000.0;
var2 = var1 * var1 * ((double)dig_P6) / 32768.0;
var2 = var2 + var1 * ((double)dig_P5) * 2.0;

```

```

var2 = (var2/4.0)+(((double)dig_P4) * 65536.0);
var1 = (((double)dig_P3) * var1 * var1 / 524288.0 + ((double)dig_P2) * var1)
var1 = (1.0 + var1 / 32768.0)*((double)dig_P1);
if (var1 == 0.0)
{
return; // avoid exception caused by division by zero
}
p = 1048576.0 - (double)adc_P;
p = (p - (var2 / 4096.0)) * 6250.0 / var1;
var1 = ((double)dig_P9) * p * p / 2147483648.0;
var2 = p * ((double)dig_P8) / 32768.0;
p = (p + (var1 + var2 + ((double)dig_P7)) / 16.0);
*pressure = (long)p;
}

```

3.2.1 Разработка функций сценариев.

В цикле программы запускаем следующие функции `bmp280Convert(&temperature, &pressure)`, `rele_rule`, `rele_rule1`, `light`, `door` и `Delay_Ms(1000)`. Они будут повторяться.

Для управления климатом были созданы функции `void rele_heating(void)` и `void rele_fan(void)`, которые при похолодании включают реле обогрева и при потеплении включают реле вентилятора. Они находятся в цикле программы. В зависимости от значений переменной `temperature` производится сравнение с переменной `temp` (это для `rele_heating`) и `temp1` (это для `rele_fan`). Если сравнение с условием соответствует, тогда включаем реле и с помощью функции `printf` отправляем в терминал значение (это нужно для отладки). Иначе выключаем. В переменные `temp`, `temp 1` устанавливаем температуру для включения обогрева или вентиляции. Ниже представлен листинг функции.

```

long temp = 255; //установка тепературы включения обогрева
long temp1 = 280; //установка тепературы включения ветиляции
void rele_heating (void) // функция включения обогрева при
похолодании

```

```

{
    if (temperature < temp) // если становится холодно
    {
        d0_reset; // включаем реле обогрева
        printf("heating on\r\n");
    }
    else { // иначе выключаем
        d0_set;
    }
}

void rele_fan(void) // функция включения вентиллятора при по-
теплении
{
    if (temperature > temp1) // если становится тепло
    {
        d2_reset; // включаем реле вентиллятора
        printf("fan on\r\n");
    }
    else { // иначе выключаем
        d2_set;
    }
}

```

Для считывания показаний с фоторезиста (датчика освещения) была создана функция `void light(void)` и переменная `Get_ConversionVal` в файле `adc.c`. В переменной `ADC_val` записываются значения от `ADC1`. В переменной `Get_ConversionVal` сохраняются откалиброванные значения. В функции `void light` Функция `void light(void)` ожидает флаг готовности `ADC1`, после готовности очищает его. В условии сравнивает `Get_ConversionVal` с значением 1500, если выполняется включаем реле света с помощью функции `printf` отправляем в терминал значение (это нужно для отладки). Иначе выключаем. Ниже представлен листинг функции.

```

u16 Get_ConversionVal(s16 val)
{

```

```

        if((val + ADC_val) < 0||val==0)
            return 0;
        if((ADC_val + val) > 4095||val==4095)
            return 4095;
        return (val + ADC_val);
void light(void)
{
    while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
    ADC_ClearFlag(ADC1, ADC_FLAG_EOC);
    if (Get_ConversionVal(ADC1->>IDATAR1)>1500)
    {
        GPIOA->BCR |= GPIO_Pin_1;
        printf("light on\r\n");
    }
    else {
        GPIOA->BSHR |= GPIO_Pin_1;
    }
}

```

Функция void door(void) выполняет опрос датчиков холла на закрытие дверей и окон void door(void). Так как написана функция void GPIO_Toggle_out на ввод данных. Условие проверяет порты PB4, PB5 на логическую 1, если условие верно отправляем в терминал значение с помощью функции printf (это нужно для отладки). Ниже представлен листинг функции.

```

void door(void) // функция опроса датчиков холла
{
    if (GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_5)!=0) // если
дверь закрыта
    {
        printf("door close\r\n");
    }
    if (GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_4)!=0) // если
окно закрыто
    {
        printf("window close\r\n");
    }
}

```

3.3 Вывод

В третьей главе было разработано устройство для считывания датчиков и управления блоком реле на базе микроконтроллера CH32V307VCT6. В ходе разработки устройства был произведён подбор датчиков и блока реле. Разработан алгоритм взаимодействия периферийных устройств с системой управления объектами умного дома основанной на открытой платформе Home Assistant. С использованием микроконтроллера CH32V307VCT6 и одноплатного компьютера Orange Pi 5B. Произведена интеграция искусственного интеллекта в Home Assistant.

Для правильной работы требуется произвести отладку системы управления объектами умного дома.

4 Тестирование и отладка системы

Для автономности и стабильности работы каждого устройства необходимо произвести отладку ПО, сценариев и работу системы. Локальный сервер, на котором установлен Home Assistant должен управлять устройством через интерфейс Ethernet по протоколу MQTT. В Home Assistant установлены и настроены необходимые расширения для сопряжения с устройством считывания и управления объектами умного дома на базе МК CH32V307VCT6.

Допускаются случаи если локальный сервер выйдет из строя, на этот случай предусмотрена автономность работы каждого устройства.

4.1 Отладка ПО на микроконтроллере CH32V307VCT6

Для автономности работы системы в случае отказа или потери соединения с локальным сервером. Нужно отладить работу с устройством управления объектами умного дома, а также работу сценариев.

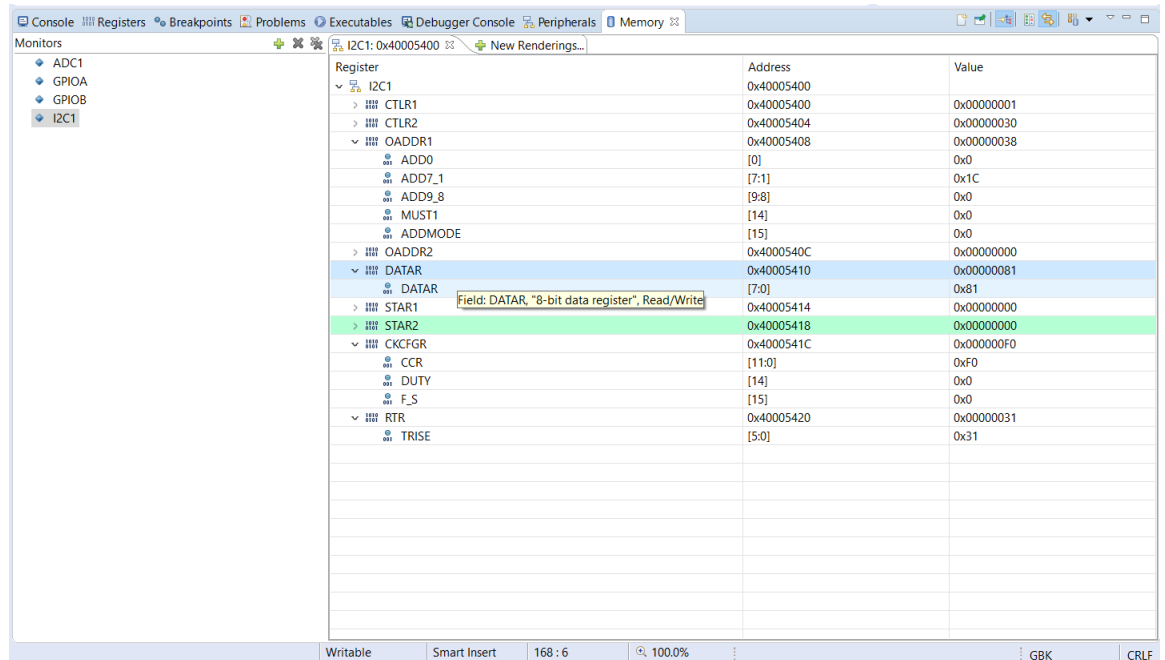
Для проверки правильности работы устройства была проведена отладка программного обеспечения. В среде разработки MounRiver. В контекстном меню нужно перейти Debug As -> OpenOCD MRS Debugging. В микроконтроллерах архитектуры RISC-V отличии от ARM, нельзя смотреть состояние регистров в реальном времени. Отладка будет производится по шагам и показаниям в терминале.

Вкладке peripherals нужно выбрать ADC1, GPIOA, GPIOB и I2C1 для просмотра состояния регистров. Продемонстрированно на рисунке 4.1.

Для отладки датчика света (фоторезист) в регистарах периферии ADC1. В нашем случае условие выполняется, так как значения больше 1500. В переменной val записано значение 2749. Терминал информирует о том, что свет включен. Продемонстрированно на рисунке 4.2.

Рисунок 4.3 – Просмотр значений АЦП

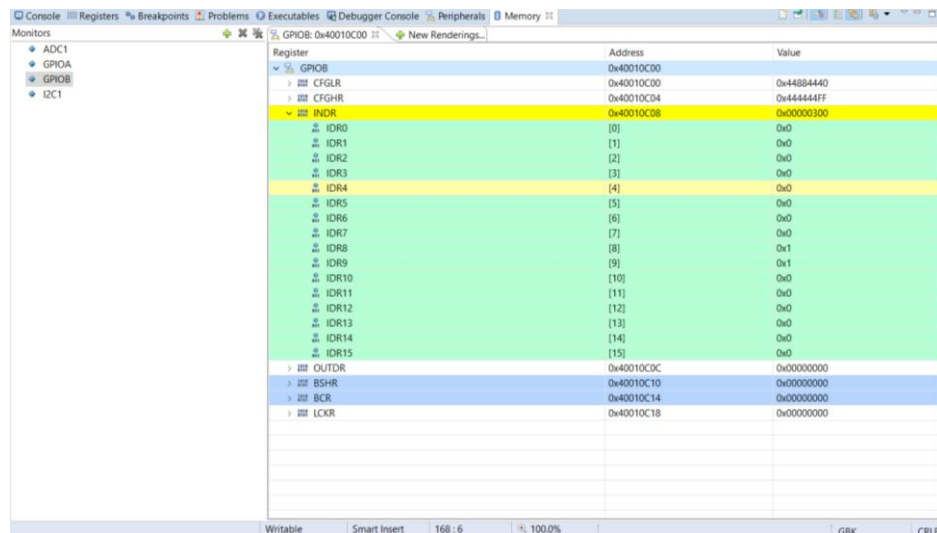
Для отладки датчика BMP280 требуется проверить регистры периферии интерфейса I2C1. В регистре DATAR записываются данные буфера I2C1, в них пишется сами данные. Продемонстрировано на рисунке 4.4



Register	Address	Value
I2C1		
CTLR1	0x40005400	0x00000001
CTLR2	0x40005404	0x00000030
OADDR1	0x40005408	0x00000038
ADD0	[0]	0x0
ADD7_1	[7:1]	0x1C
ADD9_8	[9:8]	0x0
MUST1	[14]	0x0
ADDMODE	[15]	0x0
OADDR2	0x4000540C	0x00000000
DATAR	0x40005410	0x00000081
STAR1	0x40005414	0x00000000
STAR2	0x40005418	0x00000000
CKCFGR	0x4000541C	0x000000F0
CCR	[11:0]	0xF0
DUTY	[14]	0x0
F_S	[15]	0x0
RTR	0x40005420	0x00000031
TRISE	[5:0]	0x31

Рисунок 4.4 – Просмотр значений I2C1

Для отладки датчика холла требуется проверить состояние регистра INDR. В регистр IDR4 хранятся значения порта PB4. Продемонстрировано на рисунке 4.5.



Register	Address	Value
GPIOB		
CFGLR	0x40010C00	0x440B4440
CFGLR	0x40010C00	0x444444FF
INDR	0x40010C08	0x00000000
IDR0	[0]	0x0
IDR1	[1]	0x0
IDR2	[2]	0x0
IDR3	[3]	0x0
IDR4	[4]	0x0
IDR5	[5]	0x0
IDR6	[6]	0x0
IDR7	[7]	0x0
IDR8	[8]	0x1
IDR9	[9]	0x1
IDR10	[10]	0x0
IDR11	[11]	0x0
IDR12	[12]	0x0
IDR13	[13]	0x0
IDR14	[14]	0x0
IDR15	[15]	0x0
OUTDR	0x40010C0C	0x00000000
BSHR	0x40010C10	0x00000000
BCR	0x40010C14	0x00000000
LCKR	0x40010C18	0x00000000

Рисунок 4.5 – Просмотр значений порта GPIOB

Для отладки сценария климат контроля. Требуется получить данные с датчика bmp280.c. В терминале продемонстрирована отладка. Представлено на рисунке 4.6. На рисунке 4.7 показано состояние регистров GPIOA.

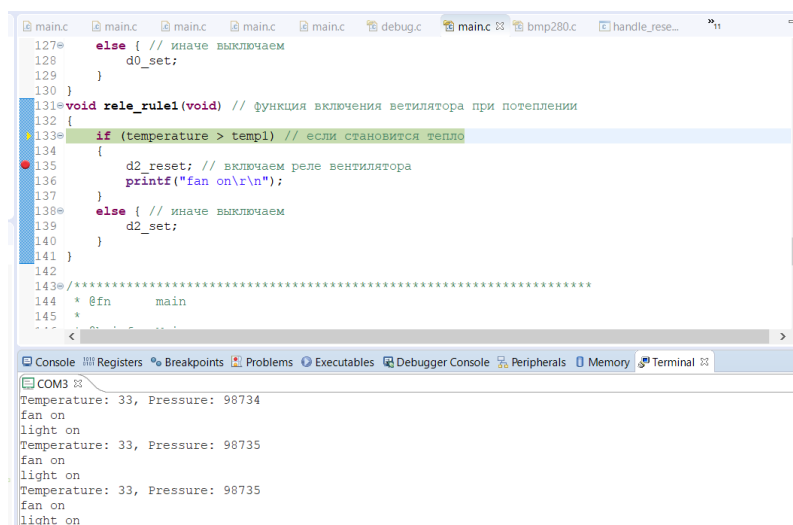


схема взаимодействия устройств умного дома. Для этого требуется установить расширение Mosquitto broker на Home Assistant.

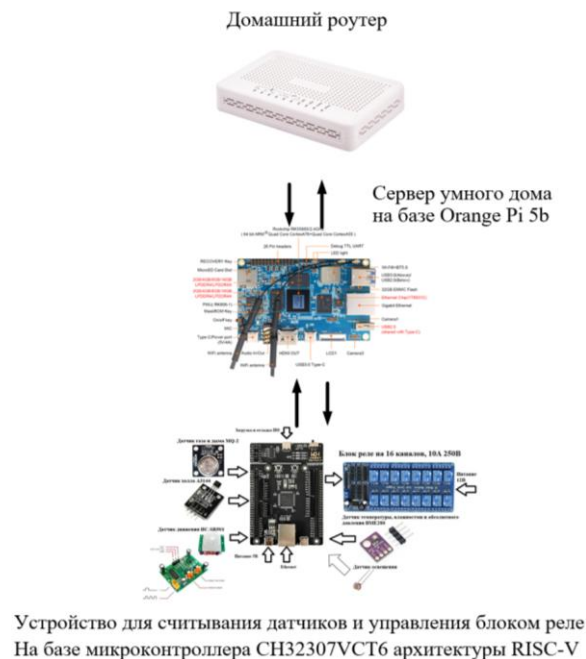


Рисунок 4.8 – Взаимодействие устройств

4.2.1 Установка MQTT-брокера на Home Assistant.

Установка Mosquitto broker через Supervisor:

- 1) Нужно открыть Home Assistant и перейти в раздел Supervisor;
- 2) Перейти в раздел add-on Store;
- 3) Найти и установить Mosquitto broker.

На рисунке 4.9 представлено где находится приложение Mosquitto broker в add-on Store.

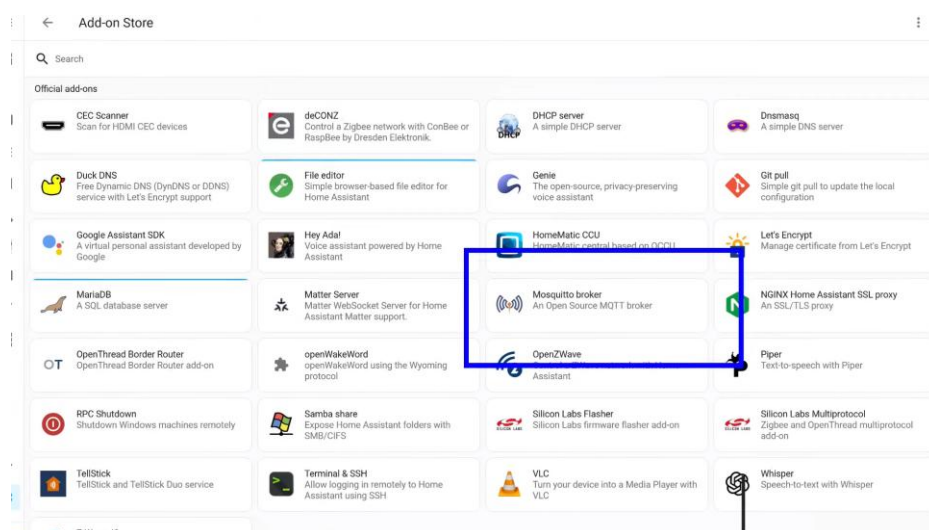


Рисунок 4.9 – Взаимодействие устройств

После установки необходимо перейти в конфигурацию и задать логин и пароль брокера. Представлено на рисунке 4.10.

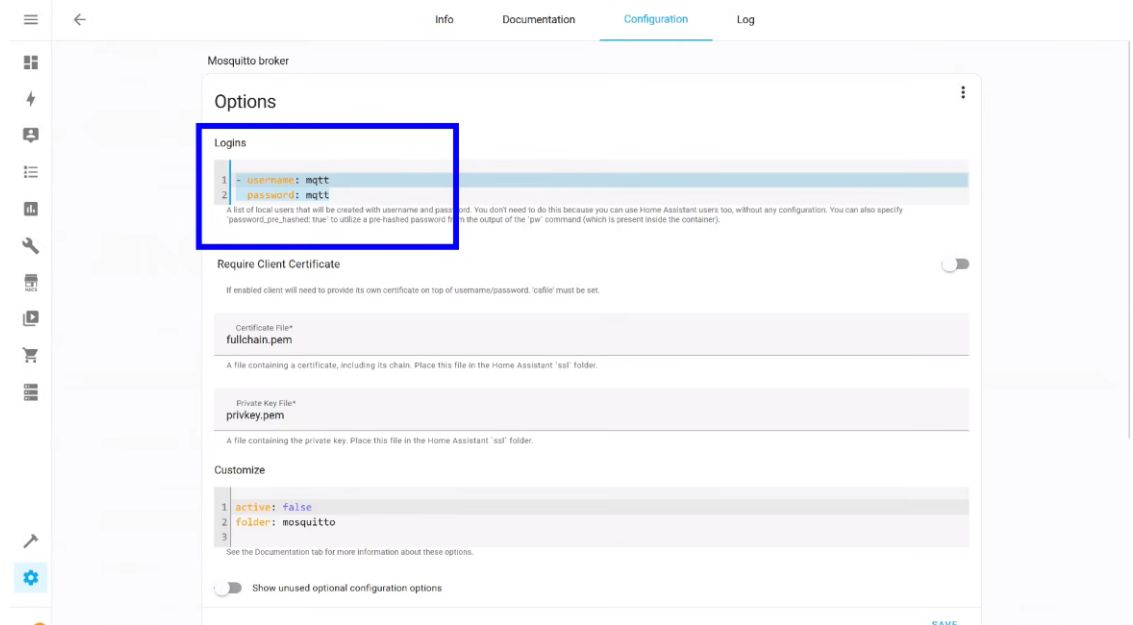


Рисунок 4.10 – Конфигурация MQTT брокера

После конфигурации требуется включить Watchdog и запустить расширение. Представлено на рисунке

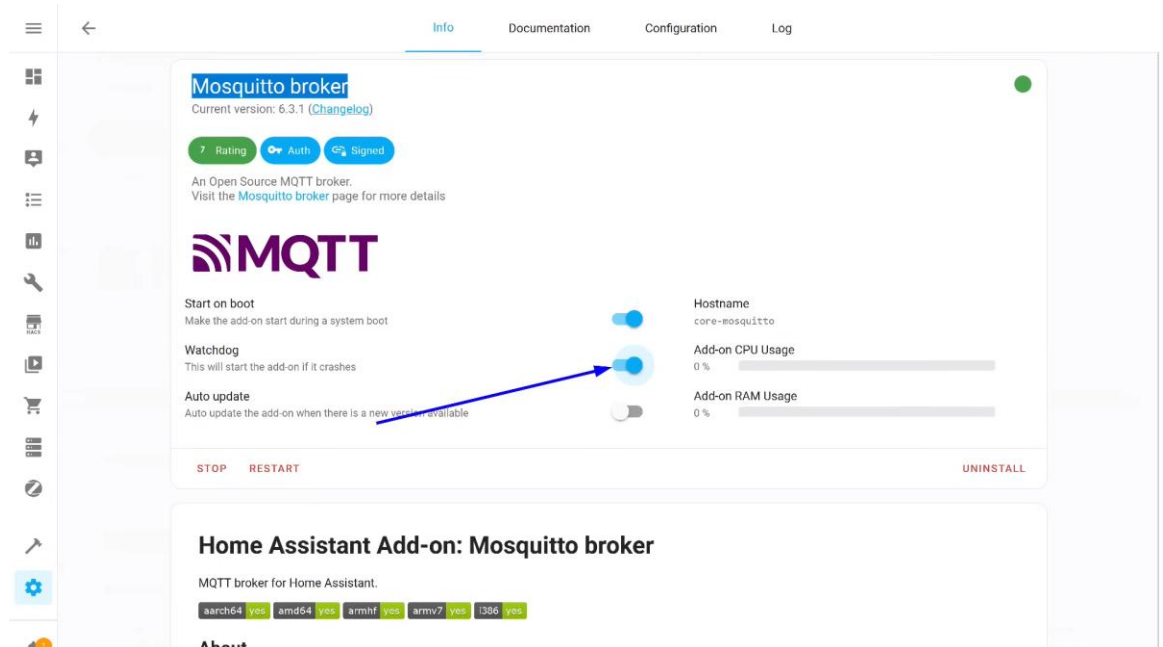


Рисунок 4.11 – Конфигурация MQTT брокера

Просмотр состояния работы MQTT присматривается во вкладке Log. Представлено на рисунке

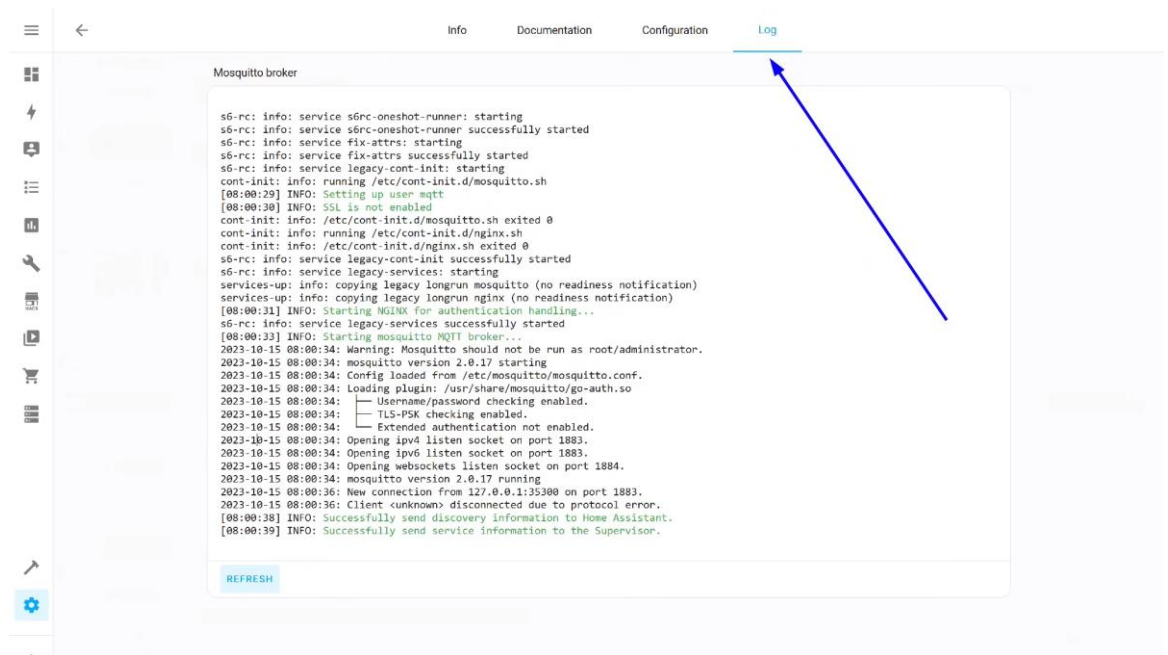


Рисунок 4.12 – Конфигурация MQTT брокера

В добавлении к интеграции требуется ввести порт (1883), логин и пароль. Представлено на рисунке 4.13

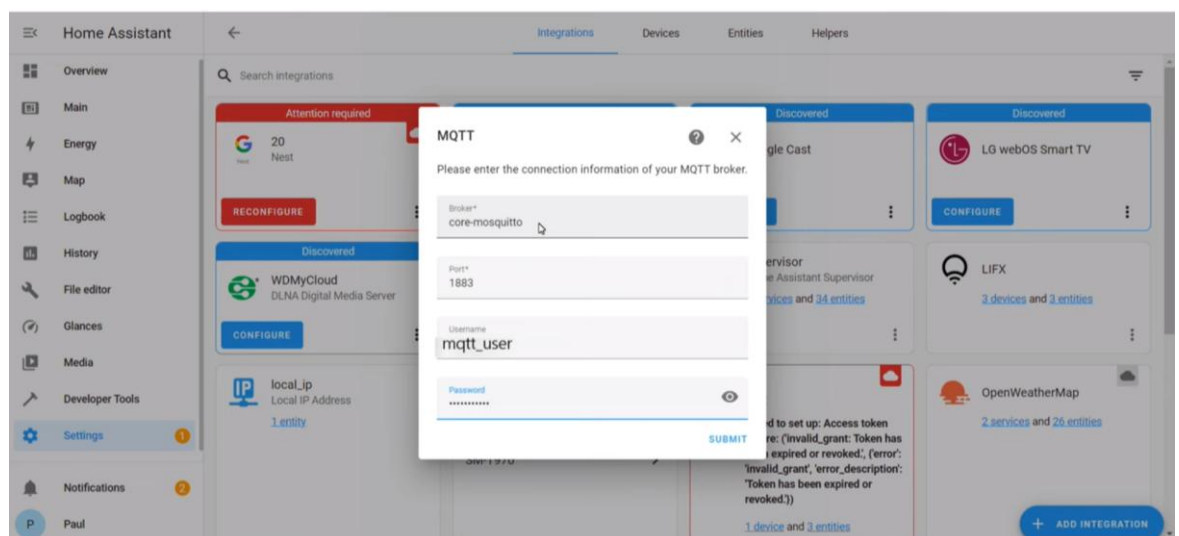


Рисунок 4.13 – Конфигурация MQTT брокера

Далее перейти в File Editor в файле configuration.yaml. Вести
 mqtt: !include mqtt.yaml

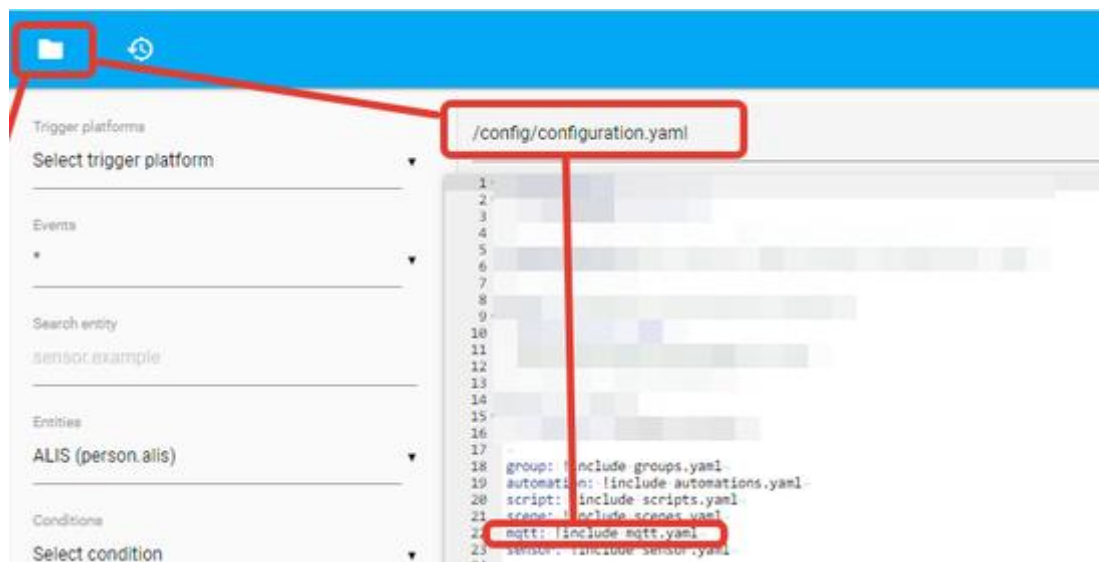


Рисунок 4.14 – Конфигурация MQTT брокера

Требуется создать в корне папки config, файл mqtt.yaml. Далее ввести параметры подключения к своему локальному брокеру. Ниже приведён листинг yaml файла.

```
broker: 192.168.1.150
port: 1883
client_id: home-assistant
keepalive: 60
username: mqtt
password: mqtt
protocol: 3.1
birth_message:
  topic: "HomeAssistant"
  payload: "Online"
  qos: 1
  retain: true
will_message:
  topic: "HomeAssistant"
  payload: "Offline"
  qos: 1
  retain: true
```

Сохранить все yaml файлы. Для управления МК требуется интегрировать LLM модель в Home Assistant

4.3 Интеграция искусственного интеллекта в Home Assistant

Для автоматизированного управления умным домом требуется интегрирование технологий искусственного интеллекта. Для полной автономии системы. На электронном ресурсе Home Assistant [24] публикуется, что модели искусственного интеллекта активно интегрируются в НА.

С развитием технологий искусственного интеллекта, большие языковые модели (LLM), такие как GPT от OpenAI, показали значительные возможности в обработке и генерации естественного языка. Интеграция этих моделей в системы управления умным домом, такие как Home Assistant, открывает новые горизонты для автоматизации и повышения функциональности. Данная работа описывает процесс интеграции LLM в Home Assistant, предоставляя методологию, реализации и потенциальные применения.

Большие языковые модели, обученные на обширных корпусах текстовых данных, способны понимать контекст и генерировать человеческие ответы на разнообразные запросы. Их использование в умных домах может значительно улучшить взаимодействие пользователя с системой, предоставляя возможность естественного языкового управления и сложной автоматизации. Для интеграции LLM модели в Home Assistant, нужно воспользоваться электронным ресурсом [25]. Требуется установить Text Generation WebUI перейти по show add-on-on. Представлено на рисунке 4.15

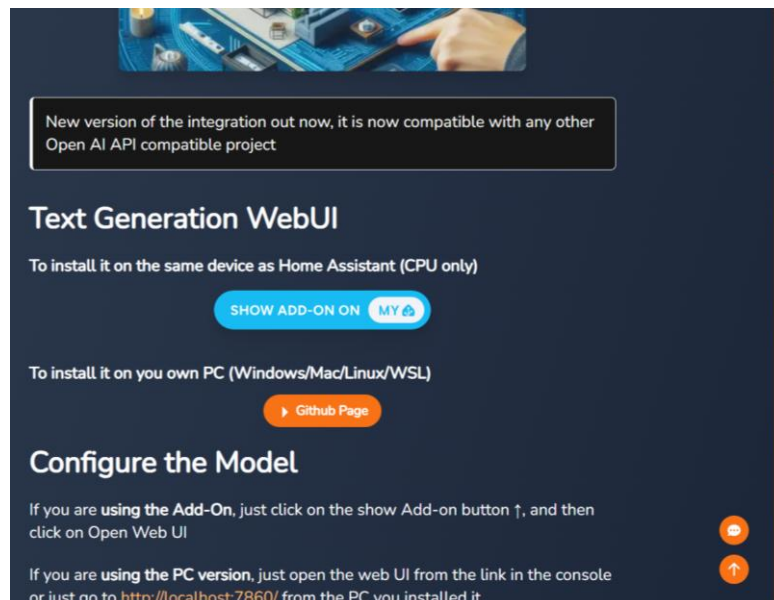


Рисунок 4.15 – Сайт fixtse

После приёма всех соглашений, нужно запустить Text Generation. Представлено на рисунке 4.16.

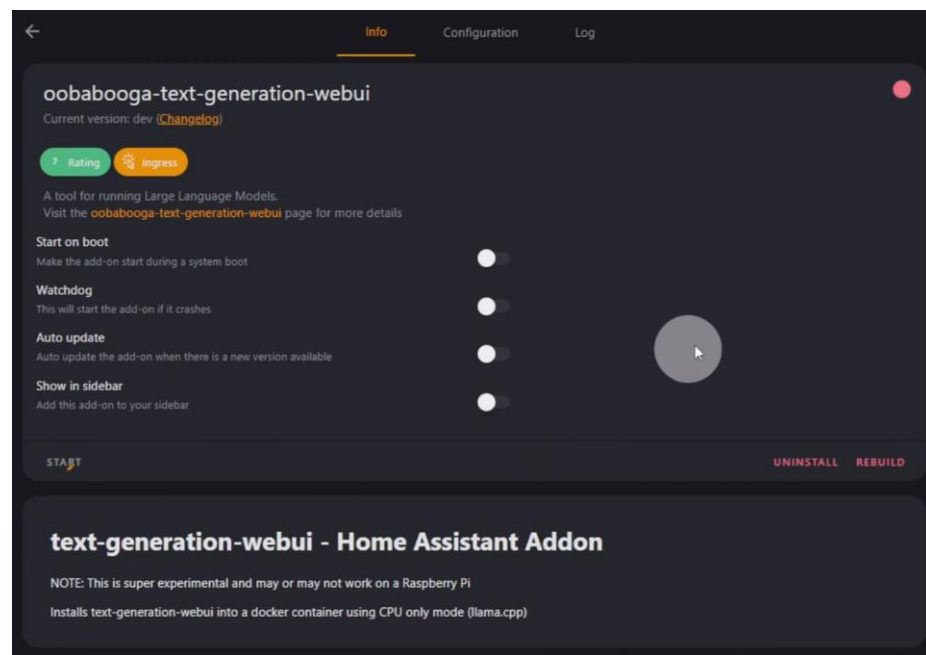


Рисунок 4.16 – Запуск Text Generation

Для работы LLM модели требуется установить LLaMa Conversation. Представлено на рисунке 4.17

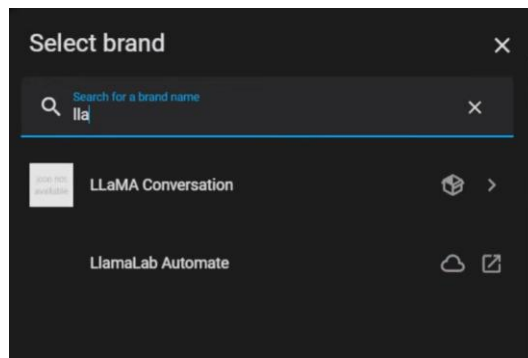


Рисунок 4.17 – Установка LLaMa Conversation

Для работы LLM модели требуется задать конфигурацию. Представлено на рисунке 4.18.

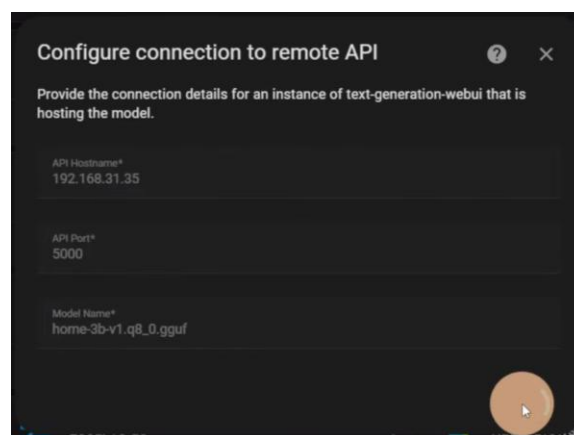


Рисунок 4.18 – Конфигурация соединения для подключения языковой модели

В конфигурации домашнего помощника выбрать языковую модель. Представлено на рисунке 4.19

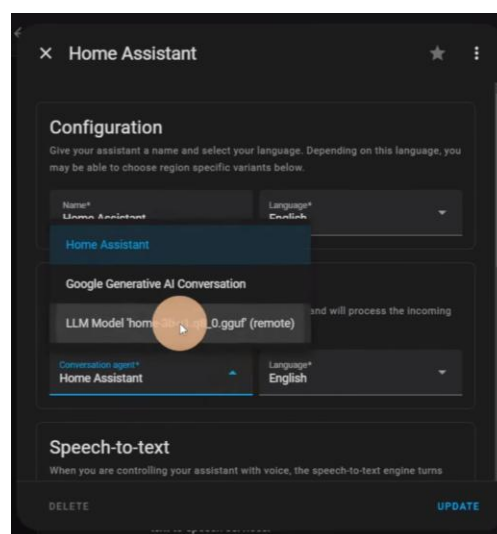


Рисунок 4.19 – Выбор языковой модели

В вкладке Expose можно просмотреть список подключенных устройств. Представлено на рисунке 4.20.

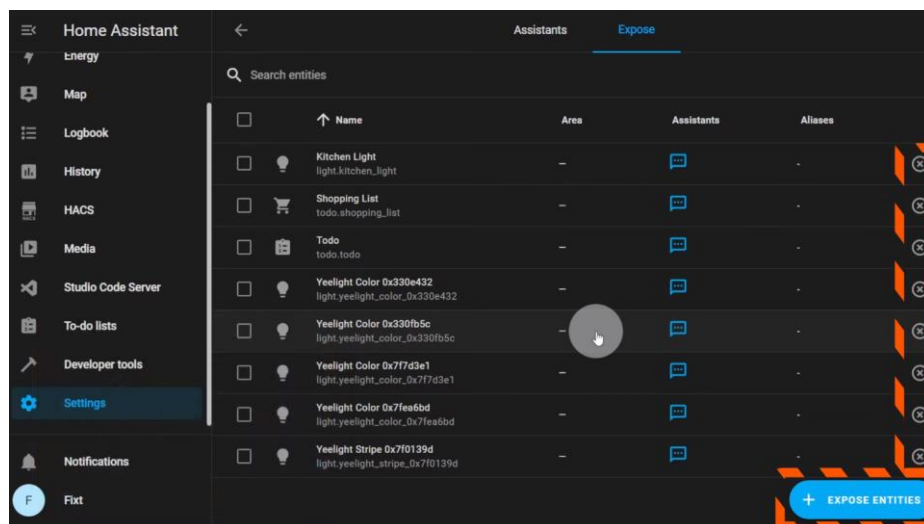


Рисунок 4.20 – Список найденных устройств

Для тестирования голосового помощника было. Был выполнен запрос проверки включен ли свет на кухне. Представлено на рисунке 4.21.

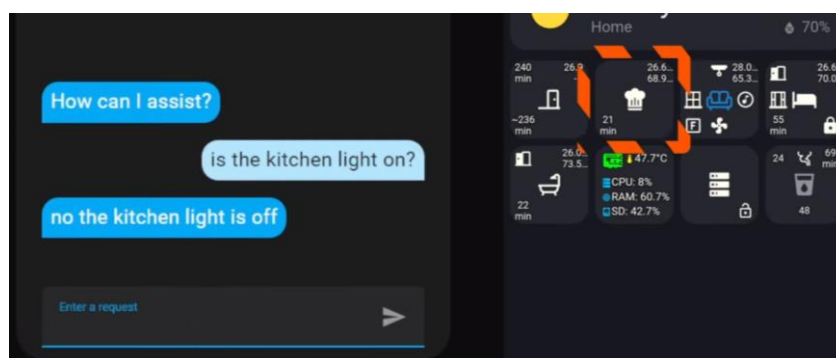


Рисунок 4.21 – Проверка запроса

Следующий запрос был на включение света на кухне. На рисунке 4.22 продемонстрирована работа домашнего помощника.



Рисунок 4.22 – Проверка работы на включение света

Была интегрирована и протестирована LLM модель в Home Assistant, но работа данной модели не стабильна. Она требует доработки.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной диссертационной работы были выполнены следующие результаты:

- разработан алгоритм взаимодействия периферийных устройств на базе микроконтроллера CH32V307VCT6 на архитектуре RISC-V
- разработано и отлажено ПО для микроконтроллера CH32V307VCT6
- предложен подход к объединению различных технологий и протоколов с применением микроконтроллера CH32V307VCT6 и одноплатного компьютера Orange Pi 5B
- применён протокол MQTT для микроконтроллера CH32V307VCT6 с целью оптимизации взаимодействия устройств и обеспечения надежной и эффективной связи элементов СУОУД
- внедрена модель LLM для дальнейшего обучения и адаптивного управления, что позволяет значительно повысить уровень автоматизации и комфорта в СУОУД
- разработка программного обеспечения для бесперебойного взаимодействия всех элементов СУОУД.

Дальнейшее развитие СУОУД может включать в себя:

- сопряжение беспроводных протоколов, такие как Matter
- дальнейшую доработку ИИ в СУОУД
- финансирование проекта Федерального масштаба

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Крюкова А.А. Особенности развития концепции «умный дом»: российский и зарубежный опыт / А.А.Крюкова, К.О. Шматок // Азимут научных исследований: экономика и управление. — 2019. — Т. 8. — No. 3 (28). — с.397-399.
2. Провалихин С. СОВРЕМЕННЫЕ АДАПТИВНЫЕ СИСТЕМЫ В УМНОМ ДОМЕ / С. Провалихин // Международный научно-исследовательский журнал. — 2024. — №1 (139). — URL: <https://research-journal.org/archive/1-139-2024-january/10.23670/IRJ.2024.139.162> — DOI: 10.23670/IRJ.2024.139.162 (дата обращения: 10.06.2024).
3. Е. В. Королёв Разработка унифицированного стека сетевых протоколов для полевых шин корабельных систем управления техническими средствами // ВАК РФ 05.13.06, 2011, // С. 9.
4. В.Г. Олифер, Н.А Олифер. Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. - СПб: Изд-во "Питер", 2010, - С. 943.
5. D. Kant, A. Johannsen, and R. Creutzburg, “Analysis of iot security risks based on the exposure of the mqtt protocol,” Electronic Imaging, vol. 2021, no. 3, pp. 96–1, 2021.
6. A. Shafqat and F.A. Tahir, “Miniaturized Tapered Meandered Dual Band Dipole Antenna for WiFi 2.4/5.8 GHz Application,” Prog. Electromagn. Res. Symp., 2017, pp. 1640–1642.
7. Данилин, С. В. Оптимальное управление беспроводной сенсорной сетью Zigbee в условиях преднамеренного электромагнитного воздействия пофору / С. В. Данилин // Телекоммуникации. - No 5 - С. 35 (из перечня ВАК). – 2013.
8. Vatheuer, Christopher & Liu, Charlie & Abedi, Ali & Abari, Omid.. Is Z-Wave Reliable for IoT Sensors. IEEE Sensors Journal. PP. 1-1. 10.1109/JSEN.2023.3328014. – 2023.

9. Маслов, В. А. Построение информационного пространства вуза с использованием беспроводной технологии bluetooth / В. А. Маслов, А. Г. Финогеев, Д. А. Кувшинников, А. А. Финогеев / / Прикладная информатика.. - № 6(18). — С. 20-31. – 2008.

10. Mota, Afonso & Serôdio, Carlos & Valente, António.. Matter Protocol Integration Using Espressif's Solutions to Achieve Smart Home Interoperability. Electronics. 13. 2217. 10.3390/electronics13112217. – 2024.

11. H.-S. Kim, S. Kumar, and D. E. Culler, "Thread/OpenThread: A compromise in low-power wireless multihop network architecture for the internet of things," IEEE Commun. Mag., vol. 57, no. 7, pp. 55–61, - 2019.

12. Абзалов, Д.Б, Сафин М.А.. Централизованная и децентрализованная система умного дома. ТЕНДЕНЦИИ РАЗВИТИЯ НАУКИ И ОБРАЗОВАНИЯ. 104. 11-14. 10.18411/trnio-12- 2023.

13. ИИ в быту: как технологии меняют наше домашнее пространство. – [URL: https://ai.gov.ru/mediacenter/ii-v-bytu-kak-tekhnologii-menyayut-nashe-domashnee-prostranstvo/](https://ai.gov.ru/mediacenter/ii-v-bytu-kak-tekhnologii-menyayut-nashe-domashnee-prostranstvo/) (дата обращения 2024-02-02).

14. Home Assistant [Электронный ресурс] URL: <https://www.home-assistant.io/> (Дата обращения: 07.05.2024).

15. Google Home [Электронный ресурс] URL: <https://home.google.com/u/0/> (Дата обращения: 07.05.2024).

16. Яндекс Алиса [Электронный ресурс] URL:<https://alice.yandex.ru/> (Дата обращения: 07.05.2024).

17. Сбер Салют [Электронный ресурс] URL:<https://salute.sber.ru/> (Дата обращения: 07.05.2024).

18. GigaDevice [Электронный ресурс] URL: <https://www.gigadevice.com/product/mcu/arm-cortex-m3/gd32f107vct6> (Дата обращения: 17.05.2023).

19. [Электронный ресурс] URL: <https://wch-ic.com/products/CH32V307.html> (Дата обращения: 17.05.2023).

20. Raspberry pi Pico [Электронный ресурс] URL: <https://www.raspberrypi.com/products/raspberry-pi-pico/> (Дата обращения: 07.11.2023).
21. Orange Pi 5b [Электронный ресурс] URL: <http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-5B.html> (Дата обращения: 02.02.2024).
22. Raspberry pi 5 [Электронный ресурс] URL: <https://www.raspberrypi.com/products/raspberry-pi-5/> (Дата обращения: 02.02.2024).
23. Mounriver [Электронный ресурс] URL: <http://mounriver.com/download> (Дата обращения: 02.05.2023).
24. 2024.6: Dipping our toes in the world of AI using LLMs [Электронный ресурс] URL: <https://current--home-assistant-docs.netlify.app/blog/2024/06/05/release-20246/> (Дата обращения: 05.06.2024).
25. FixSE [Электронный ресурс] URL: <https://fixtse.com/blog/local-ai-home-assistant> (Дата обращения: 02.05.2024).
26. Бозаров, С. (). Правовое регулирование общественных отношений, связанных с производством и применением искусственного интеллекта. Общество и инновации. 4. с. 253-266. 10.47689/2181-1415-vol4-iss9/S-pp253-266.- 2023.
27. Щербинина, М. Стефанова Н. Концепция интернет вещей. Creative Economy. 10. 1323. 10.18334/ce.10.11.37074. – 2016.
28. Basyal, Lochan, Fu, Chenglong, Du, Xiaojiang. IoT Security with Matter Protocol: Exploring a Secure and Reliable Home Automation. 10.13140/RG.2.2.31849.06249. – 2023.
29. Sittampalam, Gopinath, Ragel, Roshan, Karunarathna, Suneth.. Two-Factor Commissioning for Thread Protocol. 10.1109/ICARC57651.2023.10145748. – 2023.
30. Tulshan A.S. Survey on Virtual Assistant: Google Assistant, Siri, Cortana, Alexa / A.S. Tulshan, S.N. Dhage // Advances in Signal Processing and

Intelligent Recognition Systems: 4th International Symposium SIRS. — 4. — p. 190-201. — 2019.

31. Madakam S. The Future Digital Work Force: Robotic Process Automation (RPA) / S. Madakam, R. M. Holmukhe, D. K. Jaiswal // JISTEM-Journal of Information Systems and Technology Management. — Т. 16. — 2019.

32. Шестаков Н.А. Умный дом от Яндекс / Н.А. Шестаков // Молодежь и научно-технический прогресс.— 7. — с. 285-287. — 2021.

33. Маршев Д.В. Внедрение искусственного интеллекта в систему «Умный Дом» / Д.В. Маршев // Современные тенденции развития науки и мирового сообщества в эпоху цифровизации.. — 1. — с. 139-144. — 2023.

34. - Выбор контроллера управления и интеллектуальные датчики: учебное пособие / сост.: Г.Б. Минигалиев, А.В. Долганов - Нижнекамск: Нижнекамский химико-технологический институт (филиал) ФГБОУ ВПО «КНИТУ»,. - 136 с. – 2015.

35. - Исследование технологии RISC-V / Фролов В.А., Галактионов В.А., Санжаров В.В. Труды ИСП РАН, том 32, вып. 2, 2020 г., стр. 81-98. DOI: 10.15514/ISPRAS-2020-32(2)-7 – 2020.

36. - Брякин, Л.А. Схемотехника модулей встраиваемых систем : учеб. пособие / Л. А. Брякин, А. Л. Брякин. – Пенза : Изд-во ПГУ, — 124 с. 2016.

37. Шерстобитов И.Д. Система умного дома: перспективы и возможности развития, Вестник Науки №11 (68) т.3, ISSN 2712-8849, г. Тольяти, Россия, 19 ноября 2023.

ПРИЛОЖЕНИЕ А

(Обязательное)

Система управления объектами умного дома

Листинг программы

Файл main.c

```
long temperature = 0;
long pressure = 0;
#include "debug.h"
#include "bmp280.h"
#include "adc.h"
#define d0_set GPIOA->BSHR |= GPIO_Pin_0;
#define d1_set GPIOA->BSHR |= GPIO_Pin_1;
#define d2_set GPIOA->BSHR |= GPIO_Pin_2;
#define d3_set GPIOA->BSHR |= GPIO_Pin_3;
#define d4_set GPIOA->BSHR |= GPIO_Pin_4;
#define d5_set GPIOA->BSHR |= GPIO_Pin_5;
#define d6_set GPIOA->BSHR |= GPIO_Pin_6;
#define d7_set GPIOA->BSHR |= GPIO_Pin_7;
#define d0_reset GPIOA->BCR |= GPIO_Pin_0;
#define d1_reset GPIOA->BCR |= GPIO_Pin_1;
#define d2_reset GPIOA->BCR |= GPIO_Pin_2;
#define d3_reset GPIOA->BCR |= GPIO_Pin_3;
#define d4_reset GPIOA->BCR |= GPIO_Pin_4;
#define d5_reset GPIOA->BCR |= GPIO_Pin_5;
#define d6_reset GPIOA->BCR |= GPIO_Pin_6;
#define d7_reset GPIOA->BCR |= GPIO_Pin_7;
/* Global typedef */
void IIC_Init(u32 bound, u16 address)
{
    I2C_InitTypeDef I2C_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB |
RCC_APB2Periph_AFIO, ENABLE );
    GPIO_PinRemapConfig(GPIO_Remap_I2C1, ENABLE);
    RCC_APB1PeriphClockCmd( RCC_APB1Periph_I2C1, ENABLE );

    /* Configure I2C_EE pins: SCL and SDA */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    /* I2C configuration */
    I2C_InitStructure.I2C_Mode = I2C_Mode_I2C;
    I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
    I2C_InitStructure.I2C_OwnAddress1 = address; //0x38;
    I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;
    I2C_InitStructure.I2C_AcknowledgedAddress =
I2C_AcknowledgedAddress_7bit;
    I2C_InitStructure.I2C_ClockSpeed = bound;
```

```

I2C_Init(I2C1, &I2C_InitStructure);
    /* I2C Peripheral Enable */
    I2C_Cmd(I2C1, ENABLE);
    /* Apply I2C configuration after enabling it */
}
void GPIO_Toggle_INIT(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0| GPIO_Pin_1|
GPIO_Pin_2| GPIO_Pin_3| GPIO_Pin_4| GPIO_Pin_5| GPIO_Pin_6|
GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
void GPIO_Toggle_out(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
}
void door(void)
{
    if (GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_5)!=0)
    {
        printf("door close\r\n");
    }
    else if (GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_4)!=1)
    {
        printf("window close\r\n");
    }
}
void rele_Init(void)
{
    d0_set; d1_set; d2_set; d3_set; d4_set; d5_set; d6_set;
d7_set;
}
void rele_rule(void)
{
    if (temperature > 260)
    {

```

```

        d0_reset;
    }
    else {
        d0_set;
    }
}
int main(void)
{
    SystemCoreClockUpdate();
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    Delay_Init();
    USART_Printf_Init(115200);
    GPIO_Toggle_INIT();
    GPIO_Toggle_out();
    ADC_Function_Init();
    printf("SystemClk:%d\r\n", SystemCoreClock);

    printf("This is printf example\r\n");

    IIC_Init( 100000, 0x38);
    BMP280_Init();
    rele_Init();
    TIM1_PWM_In(1000, 48000 - 1, 500);
    while(1)
    {
        bmp280Convert(&temperature, &pressure)
        printf( "Temperature: %d, Pressure: %d\r\n",
        (int)temperature/10, (int)pressure);
        rele_rule();
        light();
        door();
        Delay_Ms(1000);
    }
}

```

Файл system_ch32v20x.c

```

#define SYSCLK_FREQ_96MHz_HSE
#elif defined SYSCLK_FREQ_96MHz_HSE static void
SetSysClockTo96_HSE( void );

void SystemCoreClockUpdate (void)
{
    uint32_t tmp = 0, pllmull = 0, pllsource = 0, Pll_6_5 = 0;

    tmp = RCC->CFGR0 & RCC_SWS;

    switch (tmp)
    {
        case 0x00:
            SystemCoreClock = HSI_VALUE;
            break;
        case 0x04:

```

```

        SystemCoreClock = HSE_VALUE;
        break;
    case 0x08:
        pllmull = RCC->CFGR0 & RCC_PLLMULL;
        pllsource = RCC->CFGR0 & RCC_PLLSRC;
        pllmull = ( pllmull >> 18) + 2;
        if(pllmull == 17) pllmull = 18;
        if (pllsource == 0x00)
        {
            if(EXTEN->EXTEN_CTR & EXTEN_PLL_HSI_PRE){
                SystemCoreClock = HSI_VALUE * pllmull;
            }
            else{
                SystemCoreClock = (HSI_VALUE >> 1) * pllmull;
            }
        }
        else
        {
            if ((RCC->CFGR0 & RCC_PLLXTPRE) != (uint32_t)RESET)
            {
#else
                SystemCoreClock = (HSE_VALUE >> 1) * pllmull;
#endif
            }
            else
            {
#else
                SystemCoreClock = HSE_VALUE * pllmull;
#endif
            }
        }

        if(Pll_6_5 == 1) SystemCoreClock = (SystemCoreClock / 2);

        break;
    default:
        SystemCoreClock = HSI_VALUE;
        break;
}

tmp = AHBPrescTable[((RCC->CFGR0 & RCC_HPRE) >> 4)];
SystemCoreClock >>= tmp;
}

```

Файл adc.c

```

void ADC_Function_Init(void)
{
    ADC_InitTypeDef  ADC_InitStructure = {0};
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
    RCC_ADCCLKConfig(RCC_PCLK2_Div8);
}

```

```

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    ADC_DeInit(ADC1);
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
    ADC_InitStructure.ADC_ExternalTrigConv =
ADC_ExternalTrigInjecConv_T1_CC4;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
    ADC_Init(ADC1, &ADC_InitStructure);
    ADC_InjectedSequencerLengthConfig(ADC1, 1);
    ADC_InjectedChannelConfig(ADC1, ADC_Channel_8,
1, ADC_SampleTime_239Cycles5);
    ADC_DiscModeChannelCountConfig(ADC1, 1);
    ADC_InjectedDiscModeCmd(ADC1, ENABLE);
    ADC_ExternalTrigInjectedConvCmd(ADC1, ENABLE);
    ADC_Cmd(ADC1, ENABLE);
    ADC_BufferCmd(ADC1, DISABLE); //disable buffer
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1));
    ADC_val = Get_CalibrationValue(ADC1);
}
u16 Get_ConversionVal(s16 val)
{
    if((val + ADC_val) < 0 || val==0)
        return 0;
    if((ADC_val + val) > 4095 || val==4095)
        return 4095;
    return (val + ADC_val);
}
void light(void)
{
    while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
    ADC_ClearFlag(ADC1, ADC_FLAG_EOC);
    if (Get_ConversionVal(ADC1->IDATAR1)>1500)
    {
        GPIOA->BCR |= GPIO_Pin_1;
    }
    else {
        GPIOA->BSHR |= GPIO_Pin_1;
    }
}

```

Файл bmp280.c

```

#include "debug.h"
#include "bmp280.h"
#include <math.h>

unsigned short dig_T1;

```

```
short dig_T2;
short dig_T3;
unsigned short dig_P1;
```

```
short dig_P2;
short dig_P3;
short dig_P4;
short dig_P5;
short dig_P6;
short dig_P7;
short dig_P8;
short dig_P9;
```

```
short bmp280ReadShort(unsigned char address)
{
    short msb=0;
    short lsb=0;
    I2C_AcknowledgeConfig(I2C1,ENABLE);
    I2C_GenerateSTART(I2C1,ENABLE);
    while( I2C_GetFlagStatus( I2C2, I2C_FLAG_BUSY ) != RESET );
    I2C_GenerateSTART( I2C2, ENABLE );
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
    I2C_Send7bitAddress(I2C1, BMP280_addr,
I2C_Direction_Transmitter);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED));
    I2C_SendData(I2C1,address);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_BYTE_TRANSMITTED));
    I2C_GenerateSTART(I2C1,ENABLE);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
    I2C_Send7bitAddress(I2C1, BMP280_addr,
I2C_Direction_Receiver);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED));
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    lsb = I2C_ReceiveData(I2C1);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    msb = I2C_ReceiveData(I2C1);
    I2C_GenerateSTOP(I2C1,ENABLE);
    I2C_AcknowledgeConfig(I2C1,DISABLE);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    I2C_ReceiveData(I2C1);
    return (msb << 8) | lsb;
}
```

```
unsigned long bmp280ReadLong(unsigned char address)

{
    unsigned long result=0;
    unsigned long msb=0;
    unsigned long lsb=0;
    unsigned long xsb=0;
```

```

    I2C_AcknowledgeConfig(I2C1,ENABLE);
    I2C_GenerateSTART(I2C1,ENABLE);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
    I2C_Send7bitAddress(I2C1, BMP280_addr,
I2C_Direction_Transmitter);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED));
    I2C_SendData(I2C1,address);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_BYTE_TRANSMITTED));
    I2C_GenerateSTART(I2C1,ENABLE);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
    I2C_Send7bitAddress(I2C1, BMP280_addr,
I2C_Direction_Receiver);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED));
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    msb = I2C_ReceiveData(I2C1);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    lsb = I2C_ReceiveData(I2C1);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    xsb = I2C_ReceiveData(I2C1);
    I2C_GenerateSTOP(I2C1,ENABLE);
    I2C_AcknowledgeConfig(I2C1,DISABLE);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    I2C_ReceiveData(I2C1);
    result = (msb << 16) | (lsb << 8) | xsb;
    return (result >> 4);
}

void bmp280WriteByte(unsigned char address, unsigned char data)
{
    I2C_GenerateSTART(I2C1,ENABLE);
    while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
    I2C_Send7bitAddress(I2C1, BMP280_addr,
I2C_Direction_Transmitter);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED));
    I2C_SendData(I2C1,address);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_BYTE_TRANSMITTED));
    I2C_SendData(I2C1,data);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_BYTE_TRANSMITTED));
    I2C_GenerateSTOP(I2C1,ENABLE);
    while (!I2C_CheckEvent(I2C1,
I2C_EVENT_MASTER_BYTE_TRANSMITTED));
}
//-----

void bmp280Convert(long* temperature, long* pressure)
{
    unsigned long adc_T;

```

```

unsigned long adc_P;
adc_T = bmp280ReadLong(BMP280_REG_RESULT_TEMPRERATURE);
adc_P = bmp280ReadLong(BMP280_REG_RESULT_PRESSURE);

double var1, var2, p, t_fine;
var1 = (((double)adc_T)/16384.0 - ((double)dig_T1)/1024.0) *
((double)dig_T2);
var2 = (((double)adc_T)/131072.0 - ((double)dig_T1)/8192.0) *
(((double)adc_T)/131072.0 - ((double) dig_T1)/8192.0)) * ((double)dig_T3);
t_fine = (var1 + var2);
*temperature = (int32_t) (t_fine*10 / 5120.0);

var1 = ((double)t_fine/2.0) - 64000.0;
var2 = var1 * var1 * ((double)dig_P6) / 32768.0;
var2 = var2 + var1 * ((double)dig_P5) * 2.0;
var2 = (var2/4.0)+(((double)dig_P4) * 65536.0);
var1 = (((double)dig_P3) * var1 * var1 / 524288.0 + ((double)dig_P2) * var1) / 524288.0;
var1 = (1.0 + var1 / 32768.0)*((double)dig_P1);
if (var1 == 0.0)
{
    return; // avoid exception caused by division by zero
}
p = 1048576.0 - (double)adc_P;
p = (p - (var2 / 4096.0)) * 6250.0 / var1;
var1 = ((double)dig_P9) * p * p / 2147483648.0;
var2 = p * ((double)dig_P8) / 32768.0;
p = (p + (var1 + var2 + ((double)dig_P7)) / 16.0);

*pressure = (long)p;
}
//-----
void BMP280_Init(void)
{
    dig_T1 = bmp280ReadShort(0x88); //dig_T1
    dig_T2 = bmp280ReadShort(0x8A); //dig_T2
    dig_T3 = bmp280ReadShort(0x8C); //dig_T3
    dig_P1 = bmp280ReadShort(0x8E); //dig_P1
    dig_P2 = bmp280ReadShort(0x90); //dig_P2
    dig_P3 = bmp280ReadShort(0x92); //dig_P3
    dig_P4 = bmp280ReadShort(0x94); //dig_P4
    dig_P5 = bmp280ReadShort(0x96); //dig_P5
    dig_P6 = bmp280ReadShort(0x98); //dig_P6
    dig_P7 = bmp280ReadShort(0x9A); //dig_P7
    dig_P8 = bmp280ReadShort(0x9C); //dig_P8
    dig_P9 = bmp280ReadShort(0x9E); //dig_P9

    bmp280WriteByte(BMP280_REG_CONFIG, BMP280_CONFIG);
    bmp280WriteByte(BMP280_REG_CONTROL, BMP280_MEAS);
}

```

Файл bmp280.h

```

#ifndef USER_BMP280_H_
#define USER_BMP280_H_

#define BMP280_addr 0xED

#define BMP280_REG_CONTROL 0xF4
#define BMP280_REG_CONFIG 0xF5

#define BMP280_REG_RESULT_PRESSURE 0xF7          // 0xF7(msb) ,
0xF8(lsb) , 0xF9(xlsb) : stores the pressure data.
#define BMP280_REG_RESULT_TEMPRERATURE 0xFA      // 0xFA(msb) ,
0xFB(lsb) , 0xFC(xlsb) : stores the temperature data.

#define BMP280_OVERSAMPLING_T1          0x20
#define BMP280_OVERSAMPLING_T2          0x40
#define BMP280_OVERSAMPLING_T4          0x60
#define BMP280_OVERSAMPLING_T8          0x80
#define BMP280_OVERSAMPLING_T16         0xA0

#define BMP280_OVERSAMPLING_P1          0x04
#define BMP280_OVERSAMPLING_P2          0x08
#define BMP280_OVERSAMPLING_P4          0x0C
#define BMP280_OVERSAMPLING_P8          0x10
#define BMP280_OVERSAMPLING_P16         0x14

#define BMP280_MODE_SLEEP                0x00
#define BMP280_MODE_FORCED               0x01
#define BMP280_MODE_NORMAL               0x03

#define BMP280_TSB_0_5                   0x00
#define BMP280_TSB_62_5                  0x20
#define BMP280_TSB_125                   0x40
#define BMP280_TSB_250                   0x60
#define BMP280_TSB_500                   0x80
#define BMP280_TSB_1000                  0xA0
#define BMP280_TSB_2000                  0xC0
#define BMP280_TSB_4000                  0xE0

#define BMP280_FILTER_OFF                0x00
#define BMP280_FILTER_COEFFICIENT2       0x04
#define BMP280_FILTER_COEFFICIENT4       0x08
#define BMP280_FILTER_COEFFICIENT8       0x0C
#define BMP280_FILTER_COEFFICIENT16      0x10

#define BMP280_SPI_OFF 0x00
#define BMP280_SPI_ON 0x01

#define BMP280_MEAS (BMP280_OVERSAMPLING_T16 |
BMP280_OVERSAMPLING_P16 | BMP280_MODE_NORMAL)
#define BMP280_CONFIG (BMP280_TSB_0_5 |
BMP280_FILTER_COEFFICIENT16 | BMP280_SPI_OFF)

void BMP280_Init(void);

```

```
void bmp280Convert(long* temperature, long* pressure);  
long bmp280CalcAltitude(long pressure);  
  
#endif /* USER_BMP280_H_ */
```

ПРИЛОЖЕНИЕ Б

(Обязательное)

Система управления объектами умного дома

Презентация

СИСТЕМА УПРАВЛЕНИЯ ОБЪЕКТАМИ УМНОГО ДОМА

Автор работы: магистрант группы 22ВВИм1 Шерстобитов И.Д.

Научный руководитель: к.т.н. доцент кафедры Никишин К.И.

Актуальность

Современные пользователи стремятся создать удобную, функциональную и надежную систему, способную интегрировать различные аспекты домашнего быта.

Применение открытых платформ, таких как Home Assistant, имеет ряд недостатков:

1. проблема безопасности,
2. нестабильная работа технологий ИИ,
3. проблема выхода из строя из-за применения одноплатного компьютера.

Предполагается что разработка автономного устройства считывания датчиков и управления блоком реле с применением LLM (large language model), сможет решить данные проблемы, обеспечить гибкость и стабильность работы системы управления объектами умного дома (СУОУД).

Цели и задачи

Целью данной работы - создание функциональной и надежной системы умного дома, которая будет учитывать современные тенденции в области IoT и обеспечит интеграцию с технологиями ИИ.

Задачи

1. анализ существующих решений СУОУД,
2. выбор оптимальных компонентов и методов их интеграции,
3. разработка программного обеспечения для бесперебойного взаимодействия всех элементов СУОУД,
4. интеграция ИИ для улучшения функциональности и адаптивности СУОУД.
5. реализация результатов тестирования СУОУД с применением технологий LLM

Популярные типы соединений СУОУД



Сравнительный анализ экосистем умного дома

Функция	Home Assistant	Google Home	Яндекс. Алиса	Сбер Салют
Открытый исходный код	Да	Нет	Нет	Нет
Кол-во интеграция с устройствами	+1800	+1000	+1000	+500
Голосовое управление	Да	Да	Да	Да

Графические интерфейсы экосистем умного дома



Home Assistant



Google Home



Яндекс.Алиса



Сбер Салют

Выбор среды для СУОУД



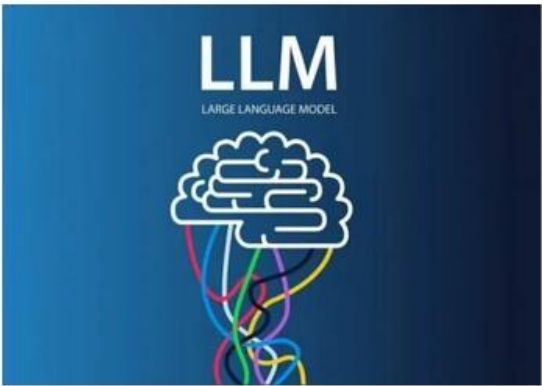
Преимущества Home Assistant

- Открытость и кастомизация
- Локальная работа и автономность
- Интеграция с множеством устройств и сервисов
- Мощные возможности автоматизации
- Экономичность

Применяемая модель ИИ

LLM (Large Language Model) — это большая языковая модель, которые используют алгоритмы машинного обучения для обобщения, прогнозирования и генерации человеческих языков на основе больших наборов текстовых данных.

В СУОУД LLM модель используется для голосового и рукописного ввода, что позволяет обучить среду Home Assistant и облегчить управление умным домом.

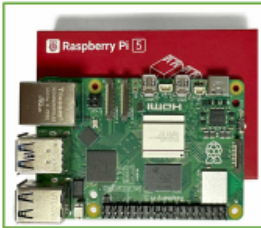


Языковые модели для Home Assistant

asor96/Home-3B-v3-GGUF					
Имя	Квантовый метод	Биты	Размер	Требуется минимальное количество оперативной памяти	Пример использования
Home-3B-v3.q2_k.gguf	Q2_K	2	1.08 ГБ	3.50 ГБ	наименьшая, значительная потеря качества - не рекомендуется для большинства целей
Home-3B-v3.q3_k_m.gguf	Q3_K_M	3	1.39 ГБ	3.89 ГБ	очень небольшая потеря качества
Home-3B-v3.q4_k_m.gguf	Q4_K_M	4	1.71 ГБ	4.21 ГБ	среднее, сбалансированное качество - рекомендуется
Home-3B-v3.q5_k_m.gguf	Q5_K_M	5	1.99 ГБ	4.49 ГБ	большая потеря качества при очень низком уровне - рекомендуется
Home-3B-v3.q8_0.gguf	Q8_0	8	2.97 ГБ	5.47 ГБ	очень большая и крайне низкая потеря качества - не рекомендуется

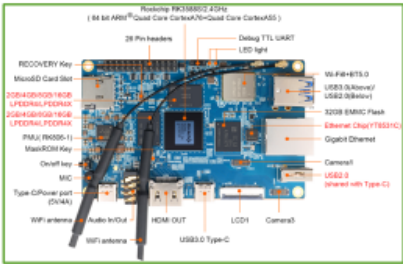
Источник: fixtse.com

Сравнительный анализ одноплатных компьютеров для локального сервера СУОУД



Основные характеристик Raspberry Pi 5

- процессор Broadcom BCM2712, 4-ядерный Cortex-A76,
- графика VideoCore VII,
- оперативная память 4 Гб или 8 Гб LPDDR4,
- накопитель microSD, PCIe 2.0,
- сетевое соединение Gigabit Ethernet, Wi-Fi 6, Bluetooth 5.0,
- интерфейсы 2 x USB 3.0, 2 x USB 2.0, 1 x USB Type-C



Основные характеристики Orange Pi 5B:

- процессор: Rockchip RK3588, 8-ядерный (4 x Cortex-A76 + 4 x Cortex-A55),
- графика: ARM Mali-G610 MP4,
- оперативная память: До 8 Гб LPDDR4/LPDDR4X,
- накопитель: eMMC модуль (до 256 Гб), слот для microSD,
- сетевое соединение: Gigabit Ethernet, Wi-Fi 6, Bluetooth 5.3,
- интерфейсы: 2 x USB 3.0, 2 x USB 2.0, 1 x USB Type-C (OTG), HDMI 2.1, 3.5mm аудиоразъем, MIPI DSI, MIPI CSI,

Сравнительный анализ микроконтроллеров для взаимодействия периферийных устройств



GD32F107VCT6



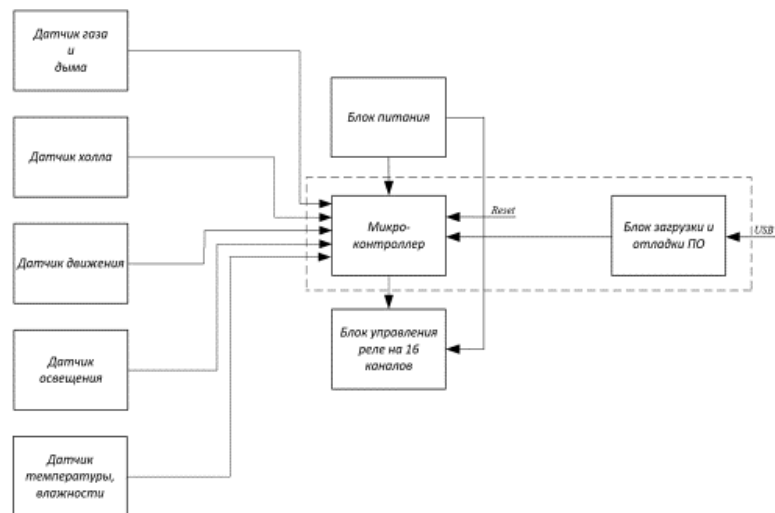
CH32V307VCT6



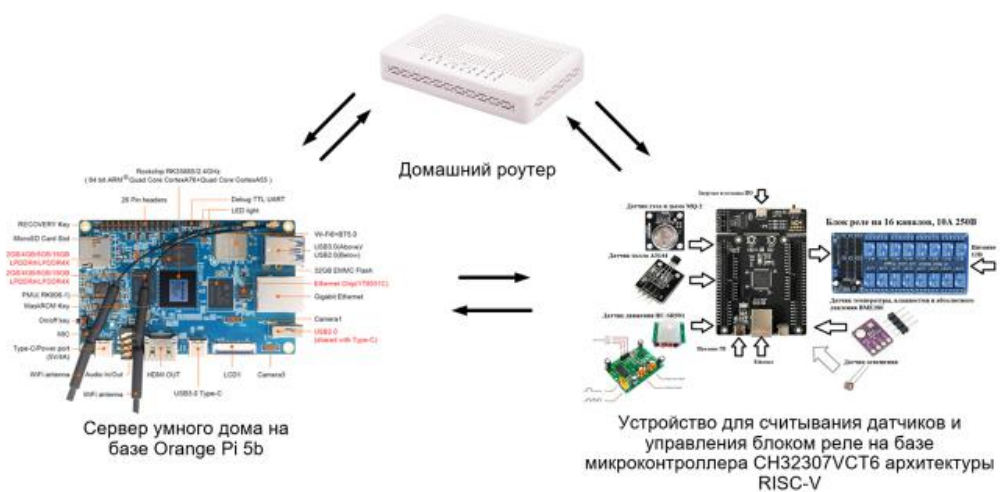
Raspberry Pi Pico

Характеристика	GD32F107VCT6	CH32V307VCT6	Raspberry Pi Pico
Процессор	ARM Cortex-M3	RISC-V	ARM Cortex-M0+
Частота процессора	до 108 МГц	до 144 МГц	до 133 МГц
Память	256 Кб Flash, 64 Кб SRAM	3072 Кб Flash, 256 Кб SRAM	2 Мб Flash, 264 Кб SRAM
Интерфейсы	USB OTG, Ethernet, CAN, I2C, SPI, UART	Ethernet, USB 2.0, CAN, I2C, SPI, UART	I2C, SPI, UART, USB 1.1
Количество пинов	100	100	26
Рабочее напряжение	2.6В - 3.6В	2.7В - 3.6В	1.8В - 3.3В
Цифровые/Аналоговые порты	80 GPIO, 3 АЦП (12-бит)	80 GPIO, 2 АЦ (12-бит)	26 GPIO, 3 АЦП (12-бит)
Применение	Сетевое оборудование, промышленная автоматизация	Высокопроизводительные встроенные системы, IoT	Образование, DIY проекты, прототипирование

Структурная схема взаимодействия периферийных устройств



Графическая схема взаимодействия всех устройств СУОУД



The figure consists of five flowcharts illustrating the algorithm for determining the degree of damage to the structure of the ship's hull.

- Flowchart 1 (Main Algorithm):** Starts with "Начало" (Start), followed by "Оценить состояние" (Evaluate condition), then "Сопоставить фактически полученное значение с нормативом (ГОСТ)" (Compare the actually obtained value with the standard (GOST)). A decision diamond asks "Есть ли повреждение?" (Is there damage?). If "Да" (Yes), it proceeds to "Оценить степень повреждения по ГОСТ" (Evaluate the degree of damage according to GOST). If "Нет" (No), it proceeds to "Завершение алгоритма" (End of algorithm). The evaluation process involves a series of decision diamonds for different types of damage (e.g., "Есть ли повреждение обшивки?" - Is there damage to the plating?), each leading to a specific "Степень повреждения" (Degree of damage) value (e.g., "Степень повреждения обшивки - 1", "Степень повреждения обшивки - 2", etc.).
- Flowchart 2 (Sub-algorithm for determining the degree of damage to the hull structure):** Starts with "Начало" (Start), followed by "Оценить состояние" (Evaluate condition), then "Сопоставить фактически полученное значение с нормативом (ГОСТ)" (Compare the actually obtained value with the standard (GOST)). A decision diamond asks "Есть ли повреждение?" (Is there damage?). If "Да" (Yes), it proceeds to "Оценить степень повреждения по ГОСТ" (Evaluate the degree of damage according to GOST). If "Нет" (No), it proceeds to "Завершение алгоритма" (End of algorithm). The evaluation process involves a series of decision diamonds for different types of damage (e.g., "Есть ли повреждение обшивки?" - Is there damage to the plating?), each leading to a specific "Степень повреждения" (Degree of damage) value (e.g., "Степень повреждения обшивки - 1", "Степень повреждения обшивки - 2", etc.).
- Flowchart 3 (Sub-algorithm for determining the degree of damage to the hull structure):** Starts with "Начало" (Start), followed by "Оценить состояние" (Evaluate condition), then "Сопоставить фактически полученное значение с нормативом (ГОСТ)" (Compare the actually obtained value with the standard (GOST)). A decision diamond asks "Есть ли повреждение?" (Is there damage?). If "Да" (Yes), it proceeds to "Оценить степень повреждения по ГОСТ" (Evaluate the degree of damage according to GOST). If "Нет" (No), it proceeds to "Завершение алгоритма" (End of algorithm). The evaluation process involves a series of decision diamonds for different types of damage (e.g., "Есть ли повреждение обшивки?" - Is there damage to the plating?), each leading to a specific "Степень повреждения" (Degree of damage) value (e.g., "Степень повреждения обшивки - 1", "Степень повреждения обшивки - 2", etc.).
- Flowchart 4 (Sub-algorithm for determining the degree of damage to the hull structure):** Starts with "Начало" (Start), followed by "Оценить состояние" (Evaluate condition), then "Сопоставить фактически полученное значение с нормативом (ГОСТ)" (Compare the actually obtained value with the standard (GOST)). A decision diamond asks "Есть ли повреждение?" (Is there damage?). If "Да" (Yes), it proceeds to "Оценить степень повреждения по ГОСТ" (Evaluate the degree of damage according to GOST). If "Нет" (No), it proceeds to "Завершение алгоритма" (End of algorithm). The evaluation process involves a series of decision diamonds for different types of damage (e.g., "Есть ли повреждение обшивки?" - Is there damage to the plating?), each leading to a specific "Степень повреждения" (Degree of damage) value (e.g., "Степень повреждения обшивки - 1", "Степень повреждения обшивки - 2", etc.).
- Flowchart 5 (Sub-algorithm for determining the degree of damage to the hull structure):** Starts with "Начало" (Start), followed by "Оценить состояние" (Evaluate condition), then "Сопоставить фактически полученное значение с нормативом (ГОСТ)" (Compare the actually obtained value with the standard (GOST)). A decision diamond asks "Есть ли повреждение?" (Is there damage?). If "Да" (Yes), it proceeds to "Оценить степень повреждения по ГОСТ" (Evaluate the degree of damage according to GOST). If "Нет" (No), it proceeds to "Завершение алгоритма" (End of algorithm). The evaluation process involves a series of decision diamonds for different types of damage (e.g., "Есть ли повреждение обшивки?" - Is there damage to the plating?), each leading to a specific "Степень повреждения" (Degree of damage) value (e.g., "Степень повреждения обшивки - 1", "Степень повреждения обшивки - 2", etc.).

[illegible]

Основные результаты работы

1. На основе анализа существующих решений СУОУД определена Home Assistant в качестве оптимальной среды разработки СУОУД
2. Интегрирована LLM модель ИИ в среду Home Assistant и проанализирована возможность её применения для более удобного и эффективного управления на уровне пользователя.
3. Выбран МК CH32V307VCT6 на архитектуре RISC-V в качестве базы для эксперимента при исследовании возможности применения периферийных устройств СУОУД
4. Разработано ПО на основе алгоритма взаимодействия периферийных устройств СУОУД, предполагающего возможность локализации функционирования, для более стабильной работы системы.

Научная новизна

1. разработан алгоритм взаимодействия периферийных устройств на базе микроконтроллера CH32V307VCT6 на архитектуре RISC-V, что позволяет создать гибкую и стабильную СУОУД
2. предложен подход к объединению различных технологий и протоколов с применением микроконтроллера CH32V307VCT6 и одноплатного компьютера Orange Pi 5B
3. применён протокол MQTT для микроконтроллера CH32V307VCT6 с целью оптимизации взаимодействия устройств и обеспечения надежной и эффективной связи элементов СУОУД
4. внедрена модель LLM для дальнейшего обучения и адаптивного управления, что позволяет значительно повысить уровень автоматизации и комфорта в СУОУД

СПАСИБО ЗА ВНИМАНИЕ

