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

Институт машиностроения
(*Наименование института полностью*)

Кафедра «Промышленная электроника»
(*Наименование кафедры*)

HTTPS://DOCS.GOOGLE.COM/SPREADSHEETS/D/1TNKMTGBVLP2LUNOW8W1DEUFDWS
LR5CROG4UCTHQGKTO/EDIT#GID=1365482727

11.04.04 «Электроника и наноэлектроника»
(код и *наименование подготовки*)

Электронные приборы и устройства
(*направленность (профиль)*)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Сетевая группа коммутирующих устройств на основе модулей Е-18»

Студент _____А.С. Минеев_____ _____
(И.О. Фамилия) *личная подпись*

Научный руководитель доцент, канд. техн. наук, А.А. Шевцов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2022

Содержание

Введение

В связи с тем, что наше общество постоянно развивает все новые и новые технологии, создает устройства для улучшения условий трудовой деятельности или же для автоматизации процессов с последующим их улучшением, то, безусловно, возникает желание, а также необходимость в их использовании. А для этого, в большинстве случаев, требуется некая вспомогательная часть, так как для того, чтобы передать и обработать информацию, которая будет использоваться в повседневности, на предприятиях и других сферах жизнедеятельности, нам потребуются не только кабели с питанием, но и информационные кабели для управления, как на улице, так и в помещениях, или отдельных строениях.

На данный момент существует большое количество стандартов [22] беспроводной передачи данных, как известных всем - Bluetooth, WiFi, так и более специализированных – ZigBee[24],[27]. Рассмотрим их более подробно, а также сравним между собой и выясним достоинства и недостатки.

Для начала рассмотрим Bluetooth. Был создан в качестве надежного, универсального и очень дешевого беспроводного радиоустройства [1]. Технология позволяет сопрягаться с оборудованием в режимах для передачи речи, мультимедиа и данных, а также гарантирует совместимость с другими устройствами.

Скорость передачи данных до 1 Мб/с. В новой пятой версии, которая еще слабо распространена – до 2 Мб/с. В современной версии с низким энергопотреблением характеризуется в активном и спящем режимах: 30 мА / 1 мкА [2].

Wifi. Наиболее распространенный стандарт [3]. Стремились к безопасности и наибольшей скорости передачи данных. Из-за чего потребляемый ток является существенным – 300 мА [4].

Zigbee. Данный стандарт был разработан в 2001 году. Предназначался для сетей из распределенных по значительной площади маломощных датчиков [5]. В соответствии с потребностями своей рыночной ниши, Zigbee использует ячеистую сетевую структуру и изощренные методы экономии [40] энергии. Данное название (пчела, зигзагообразно летающая) связано с маршрутизацией сети. Скорость передачи данных до 250 кбит/с. Предусмотрено быстрое включение и выключения прибора. Потребление тока в активном и спящем режимах: 30 мА / 1 мкА [6].

Таким образом, следует исходить из конкретных условий для выбора[28] и использования какого-либо из перечисленных стандартов. Но уже можно составить следующие требования для законченного беспроводного устройства, которое должно содержать приемопередатчик:

- устойчивая двухсторонняя связь;
- дальность работы – 5 метров и более;
- малые габаритные размеры;
- низкое энергопотребление;
- невысокая цена для конечного пользователя;

Вышеперечисленным требованиям соответствует стандарт ZigBee. Данная технология оправдана[29] и экономически эффективна[30] для таких систем, требующих автономности[31] и относительно невысокой скорости передачи данных, а именно в сети датчиков для электроэнергии, воды, газа, температуры, освещения и других.

1 Преимущества и недостатки каждого стандарта:

1.1 Wi-Fi

а. Преимущества

1. Самая высокая пропускная способность

2. Повышенная защита

б. Недостатки

1. Плохо подходит для датчиков с питанием от батарей.

1.2 Bluetooth/BLE

а. Преимущества

1. Очень малое потребление энергии

2. Отличная работоспособность в условиях интенсивных сторонних и взаимных помех.

3. Простота применения

б. Недостатки

1. Максимальная пропускная способность 2 Мб/с.

1.3 Zigbee

а. Преимущества

1. Очень малое потребление энергии

2. Фиксированные каналы между каналами WiFi в диапазоне 2,4 ГГц.

3. Поддержка диапазонов ниже 1 ГГц.

б. Недостатки

1. Сложная ячеистая сеть

2. Максимальная пропускная способность 250 кб/с.

Как можно увидеть из представленных данных, все стандарты конкурируют и пересекаются между собой. Так WiFi стремится сделать энергопотребление ниже, в свою очередь Bluetooth и ZigBee, являясь фаворитами в данном вопросе, не могут похвастать столь впечатляющей скоростью передачи данных.

Как итог, можно утверждать, что стандарты могут перекрываться друг другом и конкретный выбор пользователя будет зависеть как от личных предпочтений, так и от конкретного места использования: в крупных городах множество точек WiFi, наиболее массовые это Bluetooth и WiFi, если же необходимо покрытие большой территории с низким потреблением – наиболее перспективен ZigBee.

Позиционирование на рынке: в бытовой технике чаще используются WiFi и Bluetooth, также WiFi используется в аппаратуре с видеосигналом, Bluetooth же в наушниках и колонках, в свою очередь ZigBee чаще применяется в сети датчиков для электроэнергии, воды, газа, температуры, освещения.

2 Описание ZigBee

Рассмотрим более подробно ZigBee, а в частности ZigBee-модуль. И чтобы его использование имело смысл, он должен быть:

- компактным — чтобы не увеличивать размеры устройств;
- экономичным — чтобы долго работать даже на батарейках;
- дешёвым — чтобы его использование имело какой-то экономический смысл;

Как известно, во многих приложениях требуются беспроводные сети связи, не обладающие высокой скоростью передачи, но надежные, живучие [55] (способные к самовосстановлению), простые в развертывании [37] и эксплуатации.

Важно также, чтобы оборудование таких сетей допускало длительную работу от автономных [36] источников питания, имело низкую стоимость, и было компактным.

Пример такого приложения – «умный дом». Такому сочетанию требований еще 10 лет назад не отвечал ни один из сетевых стандартов, что и привело к созданию стандартов IEEE 802.15.4 и ZigBee, описывающих устойчивые масштабируемые многошаговые беспроводные сети, простые в развертывании и поддерживающие самые разные приложения.

2.1 Почему именно ZigBee?

Сети ZigBee, в отличие от других беспроводных сетей передачи данных, полностью [19] удовлетворяют перечисленные выше требования, а именно:

- благодаря ячеистой (mesh) топологии сети и использованию специальных алгоритмов маршрутизации сеть ZigBee обеспечивает

самовосстановление и гарантированную доставку пакетов в случаях обрыва связи между отдельными узлами (появления препятствия), перегрузки или отказа какого-то элемента;

- спецификация ZigBee предусматривает криптографическую защиту данных, передаваемых по беспроводным каналам, и гибкую политику безопасности;
- устройства ZigBee отличаются низким электропотреблением, в особенности конечные устройства, для которых предусмотрен режим «сна», что позволяет этим устройствам работать до трех лет от одной обычной батарейки AA и даже AAA;
- сеть ZigBee – самоорганизующаяся, ее структура задается параметрами профиля стека конфигуратора и формируется автоматически путем присоединения (повторного присоединения) к сети образующих ее устройств, что обеспечивает простоту развертывания и легкость масштабирования путем простого присоединения дополнительных устройств;
- устройства ZigBee компактны и имеют относительно невысокую стоимость;

Связь в сети ZigBee осуществляется путем последовательной ретрансляции пакетов от узла источника до узла адресата [59] . В сети ZigBee предусмотрено несколько альтернативных алгоритмов маршрутизации, выбор которых происходит автоматически.

Стандарт [56] предусматривает возможность использования каналов в нескольких частотных диапазонах. Наибольшая скорость передачи и наилучшая помехоустойчивость достигается в диапазоне от 2,4 до 2,48 ГГц.

В этом диапазоне предусмотрено 16 каналов по 5 МГц.

Цена, которую пришлось заплатить в сетях ZigBee за минимизацию энергопотребления, компактность и дешевизну – относительно низкая скорость передачи данных.

«Брутто» скорость (включая служебную информацию) составляет 250 кбит/с. Средняя скорость передачи полезных данных, в зависимости от загрузки сети и числа ретрансляций, составляет от 5 до 40 кбит/с.

Расстояние между рабочими станциями сети составляет десятки метров внутри помещений и сотни метров на открытом воздухе. За счет ретрансляций покрываемая сетью зона может быть весьма значительной: до нескольких тысяч квадратных метров в помещении и до нескольких гектар на открытом пространстве.

Более того, сеть ZigBee в любой момент может быть расширена добавлением новых элементов или наоборот разбита на несколько зон простым назначением соответствующего числа новых конфигураторов сети. Это бывает полезно для снижения нагрузки и соответственно повышения скорости передачи данных.

2.2 Устройства ZigBee

Сети ZigBee строятся из базовых станций трех основных типов: координаторов, маршрутизаторов и конечных устройств.

Координатор запускает сеть и управляет ею. Он формирует сеть, выполняет функции центра управления сетью и доверительного центра (trust-центра) – устанавливает политику безопасности, задает настройки в процессе присоединения устройств к сети, ведает ключами безопасности.

Маршрутизатор транслирует пакеты, осуществляет динамическую маршрутизацию, восстанавливает маршруты при перегрузках в сети или отказе какого-либо устройства. При формировании сети [39] маршрутизаторы присоединяются к координатору или другим

маршрутизаторам, и могут присоединять дочерние устройства – маршрутизаторы и конечные устройства.

Маршрутизаторы работают в непрерывном режиме, имеют стационарное питание и могут обслуживать «спящие» устройства. Маршрутизатор может обслуживать до 32 спящих устройств.

Конечное устройство может принимать и отправлять пакеты, но не занимается их трансляцией и маршрутизацией. Конечные устройства могут подключаться к координатору или маршрутизатору, но не могут иметь дочерних устройств.

Конечные устройства могут переводиться в спящий режим [47] для экономии заряда аккумуляторов. Именно конечные устройства имеют дело с датчиками, локальными контроллерами и исполнительными механизмами.

На рисунке 1 ниже показан пример переподключения [53]. Устройство с адресом «0E3B» переподключается как «097D», а затем как «0260». Каждый раз оно присоединяется к другому маршрутизатору и получает адрес из имеющегося в распоряжении присоединяющего маршрутизатора диапазона адресов.

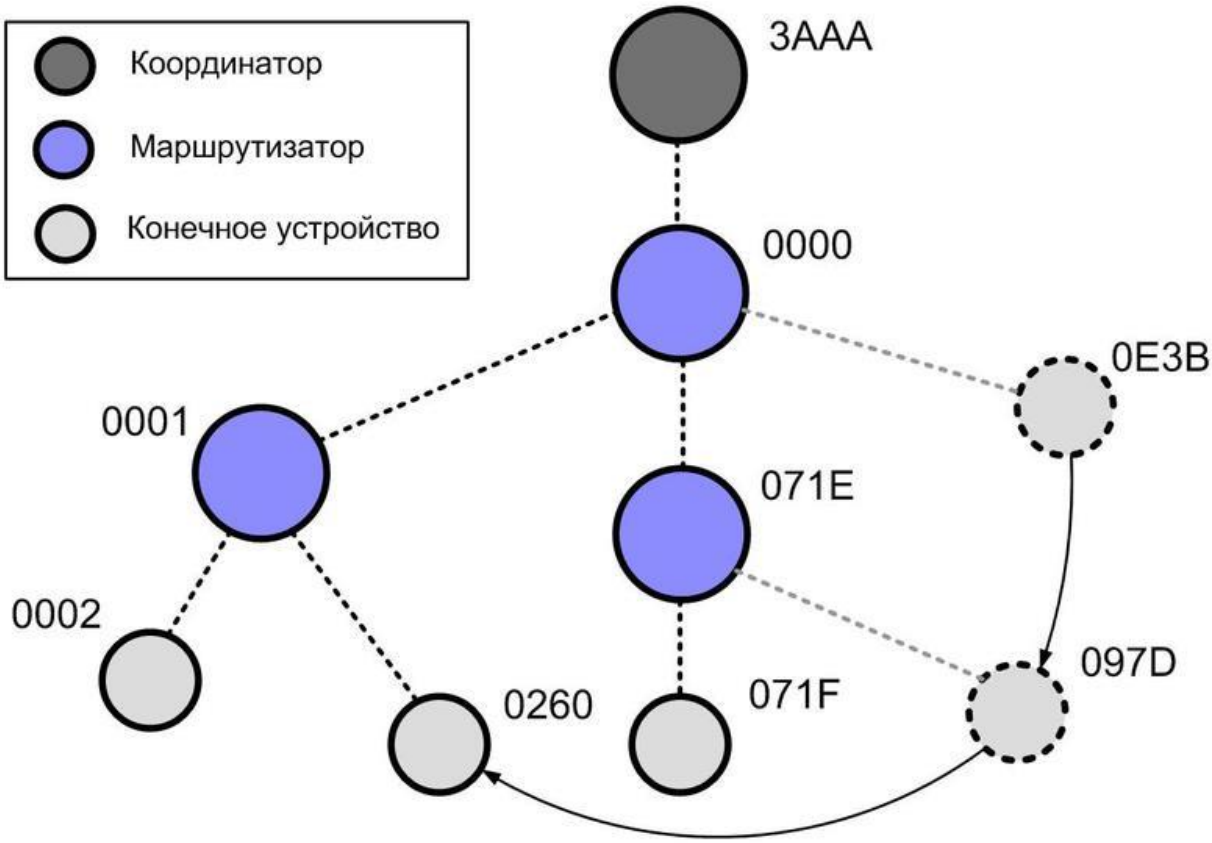

Рисунок 1 – Переподключение конечного устройства в древовидной сети

По вышеперечисленным причинам был взят ZigBee-модуль приемника\передатчика E18-MS1-PCB с чипом TI cc2530 внешний вид которого показан на рисунке 2.

Рисунок 2 – Модуль ZigBee

E18-MS1-PCB — это беспроводной модуль ZigBee небольшого размера с частотой 2,4 ГГц [38]. В наличии идет встроенная антенна с мощностью

передачи 100мВт и расстоянием между контактами 1,27мм [7]. Данное устройство [41] можно использовать в различных приложениях, например умный дом. Принципиальная схема приведена в приложении А.

Этот модуль [42] использует чип cc2530 с микроконтроллером 8051 и беспроводной приемо-передатчик, встроенный в него, для значительного **увеличения дальности связи и улучшения стабильности.**

2.3 Особенности

- расстояние для связи в идеальных условиях составляет до 800 метров;
- встроенный стек протоколов ZigBee;
- поддержка периферийных устройств, таких как АЦП, ШИМ, GPIO;
- встроенный кварцевый генератор с частотой 32,768 кГц;
- диапазон передачи 2.4 ГГц;
- flash-память 256 кБ, RAM-память 8 кБ;
- маломощный и высокопроизводительный микроконтроллер 8051;
- источник питания 2.0 – 3.6 В, при 3.3 В – наилучшая производительность;
- способность длительной работы при температуре от -40 до 85^0C;
- встроенная антенна – нет нужды во внешней;

2.4 Применение

- умный дом и датчики в промышленности;
- система безопасности и система позиционирования;
- дроны, беспроводные пульты для дистанционного управления;
- пульты для дистанционного управления игрой;
- товары для здоровья;
- беспроводные гарнитуры;
- автомобильная промышленность;

2.5 Параметры

Далее в таблицах 1 и 2 будут приведены предельные и рабочие параметры [42],[44].

Таблица 1 – Предельные параметры

Основной параметр	Производительность		Примечание
	Минимум	Максимум	
Источник питания (В)	0	3,6	Напряжение свыше 3.6 приведет к необратимому повреждению модуля
Блокирующая мощность (дБм)	-	10	Шанс невелик, когда модуль используется на коротких расстояниях
Рабочая температура (^{0}C)	-40	+85	Промышленный уровень

Таблица 2 – Рабочие параметры

Основной параметр	Производительность			Примечание
	Мин.	Тип.	Макс.	
Рабочее напряжение (В)	2,0	3,3	3,6	≥3,3 В обеспечивает выходную мощность
Уровень связи (В)	-	3,3	-	Для 5 В ТТЛ может возникнуть риск сгорания
Рабочая температура (^{0}C)	-40	-	+85	Промышленный уровень

Продолжение таблицы 2

Рабочая частота (ГГц)		2400	-	2480	ISM-диапазом
Потребляемый ток	Ток передачи (мА)	-	98	-	Мгновенное энергопотребление
	Ток приема (мА)	-	36	-	-
Ток сна (мкА)		-	2	-	Выключается программно
Максимальная мощность передачи (дБм)		19,6	20	20,5	-
Чувствительность приема (дБм)		-99	-98	-97	Скорость передачи данных в эфире 250кБ\с

Затем далее в таблице 3 приведено описание модуля

Таблица 3 - Краткое описание модуля

Основной параметр	Описание	Примечание
Базовое расстояние	800м	Условия тестирования: чистая и открытая местность, скорость передачи 250кБ\с
Протокол	ZigBee	-
Коммуникационный интерфейс	Ввод \ вывод	Все порты ввода \ вывода выведены наружу
Flash	256 кБ	-
RAM	8 кБ	-

Продолжение таблицы 3

Микроконтроллер	8051	-
Размеры	16*27мм	-
Антенна	Встроена	Сопротивление 50 Ом

2.6 Использование

Перед началом [49] использования необходимо проверить, что модуль надежно заземлен, коэффициент пульсации источника сведен к минимуму.

Обратить внимание на правильное подключение положительного и отрицательного полюсов питания, так как обратное подключение может привести к необратимому повреждению модуля.

Убедитесь, что напряжение от источника питания соответствует рекомендуемому и не превышает максимальное, иначе модуль может быть безвозвратно поврежден.

Напряжение питания не должно часто и сильно колебаться.

Рекомендуется резервировать более 30% мощности про запас, поскольку это необходимо для долговременной стабильной работы.

Для корректной работы модуля используйте его как можно дальше от источника питания, трансформаторов, высокочастотной проводки и других механизмов с большими электромагнитными помехами.

Если же данная установка невозможна, то рекомендуется экранировать его или соответствующе заизолировать.

При монтаже обратите внимание, чтобы антенна располагалась открыто и предпочтительно вертикально вверх, так как это оказывает существенное влияние на работу модуля.

Если монтаж производится внутри корпуса, то используйте удлинитель для вытягивания антенны наружу.

Антенну нельзя устанавливать внутри металлического корпуса, так как это приведет к значительному ослаблению дальности передачи.

Также на дальность связи будут оказывать влияние препятствия – что также приведет к ее снижению.

На скорость передачи данных оказывают влияние температура, влажность и помехи.

При использовании вблизи земли производительность будет плохой, поскольку земля будет поглощать и отражать беспроводные радиоволны.

Схожим эффектом обладает морская вода – поэтому вблизи моря производительность будет понижена

Если напряжение источника питания низкое при комнатной температуре, то скорость передачи данных будет снижена.

2.7 Легкая повреждаемость модуля

Пожалуйста, проверьте источник питания, убедитесь, что он находится между рекомендуемым напряжением питания, напряжение выше, чем максимум повредит модуль.

Пожалуйста, проверьте стабильность источника питания, напряжение не должно сильно колебаться.

Пожалуйста, убедитесь, что при установке и использовании приняты антистатические меры, высокочастотные устройства имеют восприимчивость к электростатическому заряду.

Пожалуйста, убедитесь, что влажность находится в ограниченном диапазоне, некоторые части чувствительны к влажности;

Пожалуйста, избегайте использования модулей при слишком высокой или слишком низкой температуре.

Исходя из вышеперечисленного, можно судить об обоснованности выбора протокола беспроводной передачи данных, а также самого модуля и его условий для использования.

3 Практическая часть исследования

3.1 Подготовка к программной части

Для начала разработки нужно скачать и установить Z-Stack 3.0.2 — это SDK[33],[34] для разработки прошивок [16] с примерами и документацией.

Также нужно скачать и установить IAR Embeded Workbench for 8051 — это среда разработки [15] с возможностью компиляции под чипы TI cc2530.

Для разработки и отладки понадобится CCDebugger [14] — он позволяет не только прошивать чипы cc2531/cc2530, но и выполнять отладку приложения в среде[35] IAR – что показано на рисунке 3.

Рисунок 3 – CCDebugger

Создаем новый проект, который будет располагаться по адресу: 3.0.2\Projects\zstack. И далее как назовем папку. Затем смотрим, что также создалось, когда мы осуществили данное действие в программе, описанной выше.

Внутри папки CC2530DB есть файлы:

- «ваше название».ewd — настройки проекта для C-SPY;
- «ваше название».ewp — файл проекта;
- «ваше название».eww — рабочая область Workspace;

В папке CC2530DB создастся папка [17] необходимой функциональности, а внутри, в папке EXE, появится файл «ваше название».d51 — этот файл удобен для прошивки и отладки из IAR.

Но если нам надо прошить прошивку через SmartRF Flash Programmer, то сделаем небольшие изменения – изображено на рисунке 4. Для этого изменим расширение файла с «ваше название».d51 на «ваше название».hex. И уже его будем прошивать.

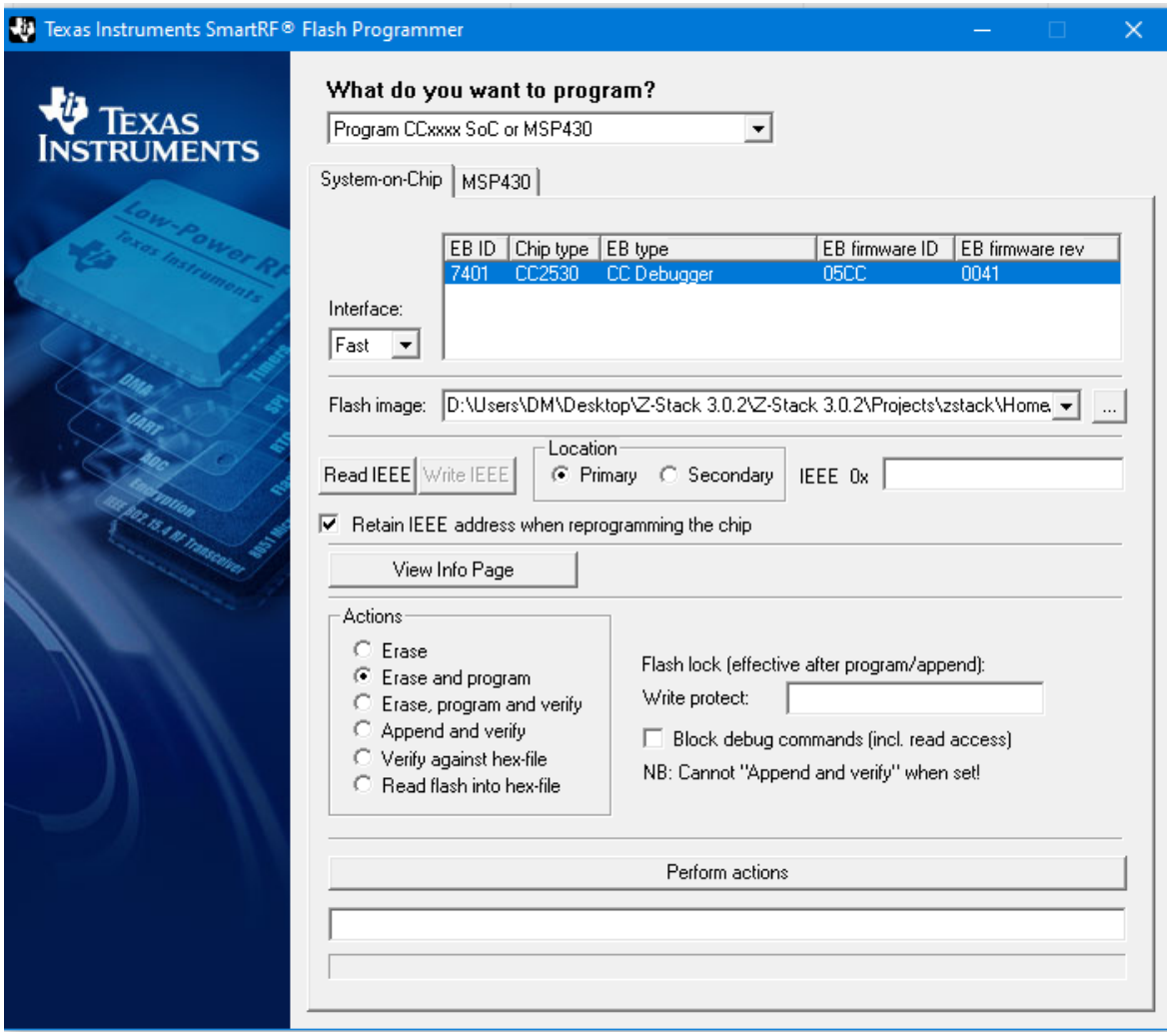

Рисунок 4 - SmartRF Flash Programmer

Выделенная синим цветом строка показывает на наш чип cc2530[18], а ниже в строке «Flash image:» после нажатия на 3 точки можно выбрать какой файл будет загружен.

После этого нажимаем «Perform action» и происходит процесс заливки скетча на устройство.

3.2 Программная часть

Для удобства пользователя и отображения информации в режиме реального времени [21] с нашего устройства, работающего с сетью ZigBee [50], было найдено программное решение, позволяющее это сделать.

Это так называемый домашний помощник (Home Assistant / HA).

Далее немного подробнее о его установке и сопутствующих программах.

Есть несколько возможных вариантов: более простой и более дешевый.

Простой включает в себя дополнительную плату Raspberry PI, которая будет выступать в качестве сервера и на которую будет установлен HA.

Дешевый – установка непосредственно на тот ПК, с экрана которого пользователь будет наблюдать за параметрами устройств в режиме реального времени.

Мы воспользуемся более дешевым методом – установим HA в Windows. Однако для того, чтобы появился режим Supervisor, который не доступен в данной ОС, мы установим еще пару программ.

Это позволит нам доустановить в самом HA приложения, с которыми тот может интегрироваться, например MQTT.

Для начала установим любую виртуальную машину, например VirtualBox – была использована версия 6.1.22.

Ее можно скачать с официального сайта [8]. Выбираем файл для нашей операционной системы (хосты Windows), запускаем и устанавливаем.

После чего скачаем файл с образом HA. Для этого перейдем по ссылке [9] на страницу и скачаем файл с именем VMDK. Это архив весом около 300 мегабайт. Распакуем его и установим.

Создаем виртуальную машину. Для этого запускаем VirtualBox – показано на рисунке 5.

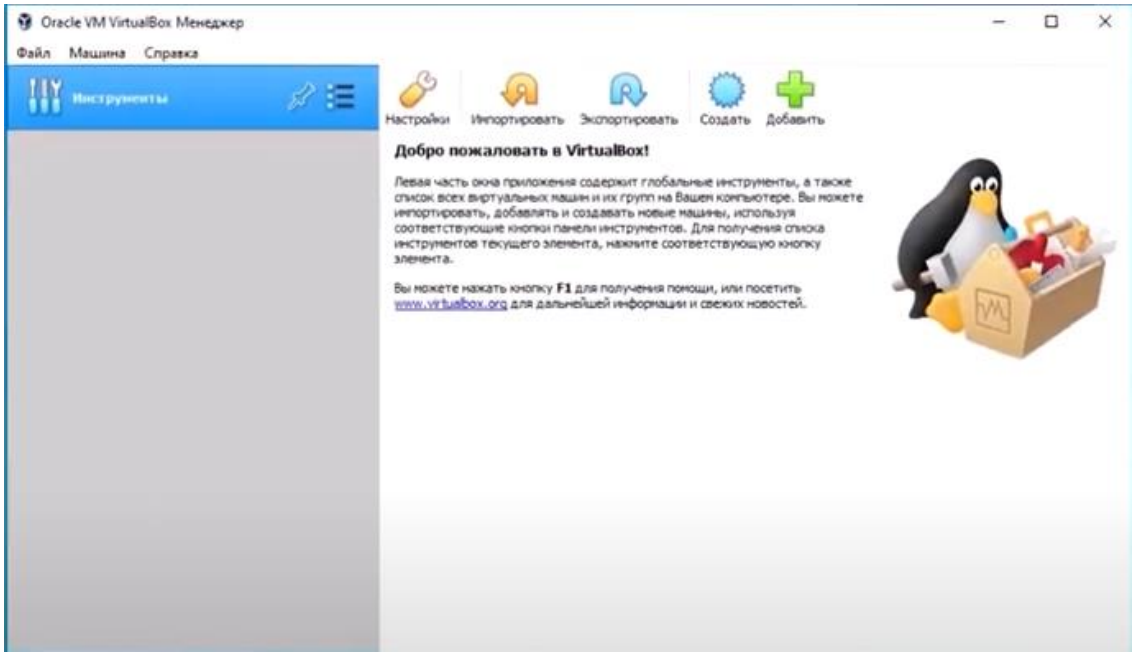

Рисунок 5 – Стартовое окно при запуске VirtualBox

Нажимаем создать. Вводим свое имя, тип выбираем Linux, версия Other Linux (64-bit). Объем памяти рекомендуется 2048 МБ.

После чего нажимаем экспертный режим и «Использовать существующий виртуальный диск». Находим скачанный файл, о котором писалось выше (VMDK) и нажимаем «создать» - отображено на рисунке 6.

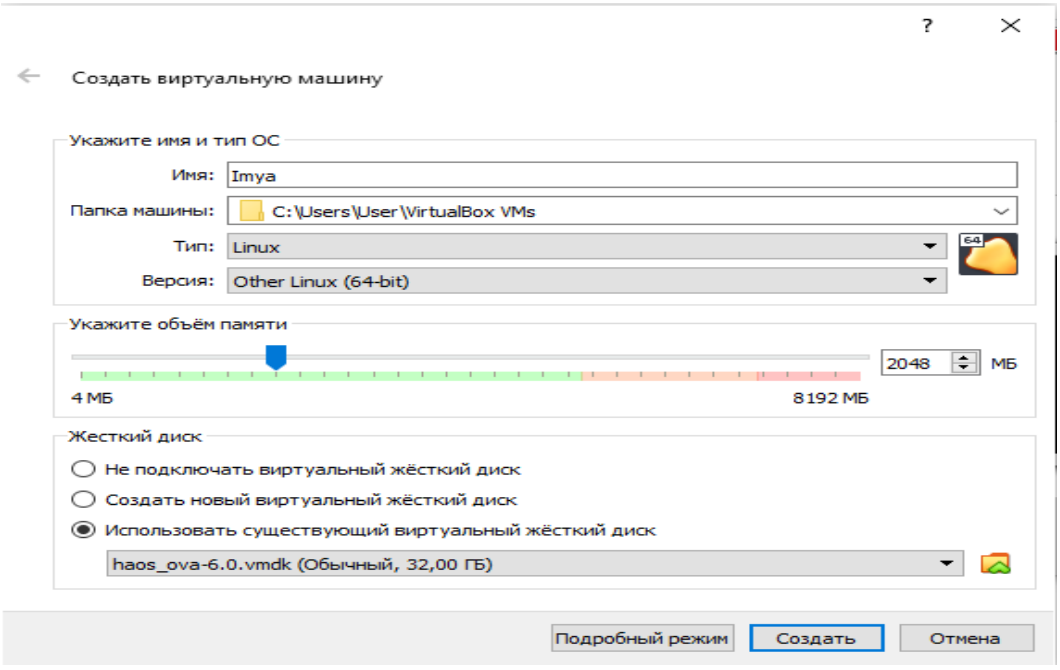

Рисунок 6 – Создание виртуальной машины

После этого появится виртуальная машина с именем Imya и статусом «Выключена».

Прежде чем ее включить зайдем в настройки – «система» и установим галочку в Enable EFI – если этого не сделать будет появляться ошибка при попытке включить виртуальную машину.

Затем перейдем в настройки – «сеть» и выберем вместо NAT – «Сетевой мост». Благодаря этому виртуальная машина станет частью локальной сети.

Нажимаем «запустить», увидим загрузку файлов, возможно нужно будет нажать пару раз Enter. После чего нам нужно выяснить ip адрес HA.

Для того, чтобы это выяснить введем в консоли [10] виртуальной машины «homeassistant login: root» и нажмем Enter [11]. После этого мы попали в консоль HA.

Введем «login», Enter, после чего вслед появившейся # введем «nmcli» и в строке: inet4 увидим требуемый адрес (192.168.***.***/24).

В строке браузера введем полученный адрес, заменив «/24» на «:8123» (192.168.***.***:8123). После чего мы окажемся в окне регистрации – показано на рисунке 7.

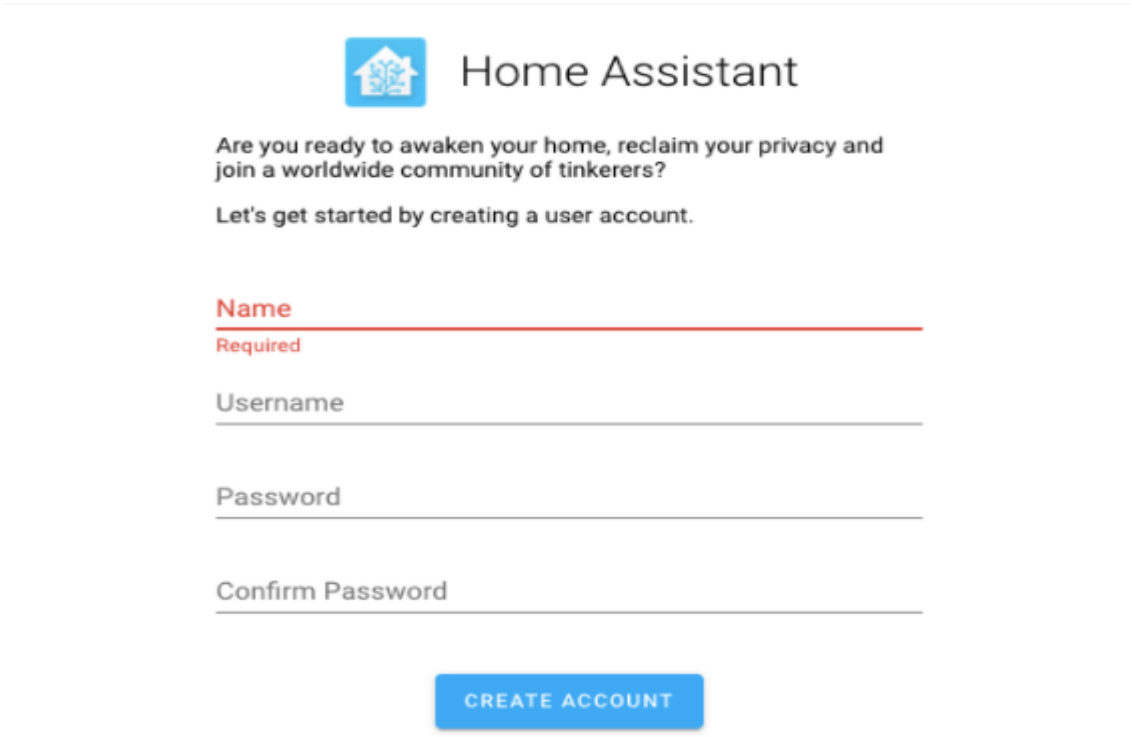

Рисунок 7 – Окно регистрации в Home Assistant

После чего придумаем имя, логин и пароль и нажимаем создать.

Далее на ваш выбор можно определить местоположение и часовой пояс нажав «определить» или же просто пропустив и выбрав эти пункты на карте вручную позднее. После чего нажимаем «завершить» и видим пользовательский интерфейс HA – представлено на рисунке 8.

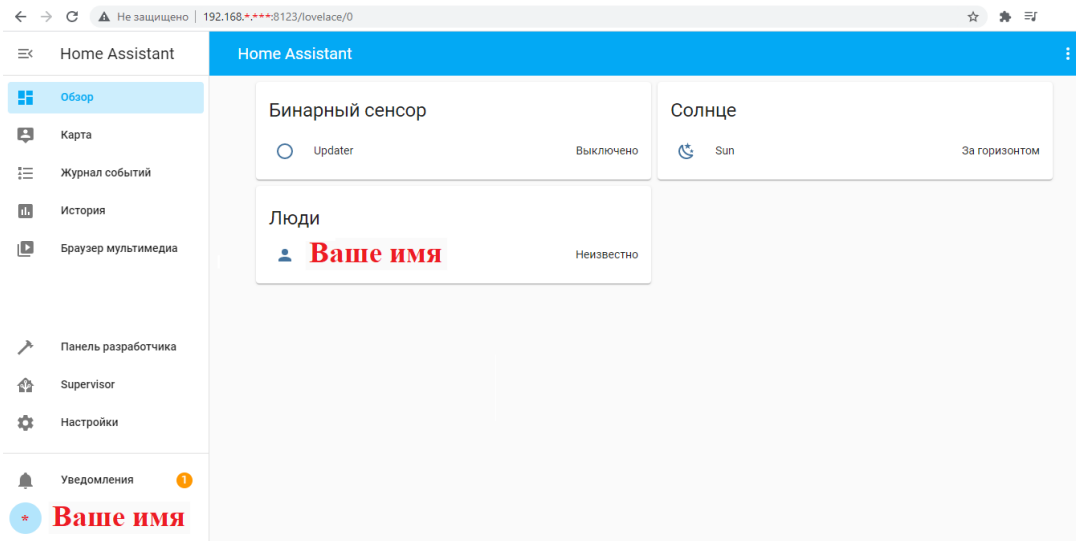

Рисунок 8 – Пользовательский интерфейс Home Assistant

Он условно разделён на 2 части: панель управления слева и основное окно справа.

Первая вкладка «Обзор» - пользовательский интерфейс, который называется «Лавлэйс». Сейчас видны все элементы, они называются сущности, которые есть по умолчанию.

Следующая вкладка «Карта» - по умолчанию показывает Амстердам, чуть позже исправим на правильное отображение.

Далее «Журнал» - показывает историю событий по каждой сущности в системе. Можно выбрать период или какой-то объект в строке поиска сверху.

Далее «История» - также данные по изменениям, но в графическом отображении на временной шкале. Также можно выбрать период.

На рисунке 9 можно увидеть «Панель разработчика» - наиболее часто посещаемое место. Первая вкладка “Состояния” – показывает статус всех сущностей в системе.

На данный момент их немного, но в дальнейшем их может быть несколько сотен и здесь поддерживается поиск по вводимым символам в любом столбце.

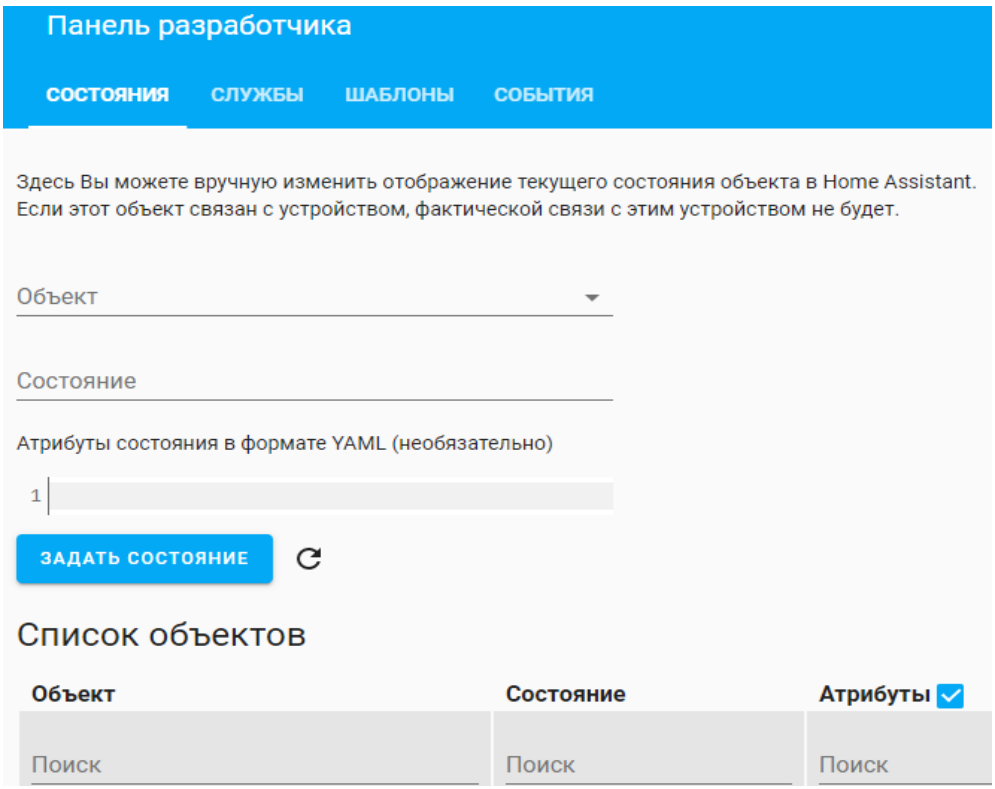

Рисунок 9 - Панель разработчика

“Службы” – дает доступ к перечню существующих в системе сервисов. Их количество тоже будет увеличиваться по мере настроек системы и дает возможность ручного запуска каждой из них.

“Шаблоны” – страница редактора шаблонов. Тут можно проверить корректность составления шаблона перед тем как ввести его в конфигурацию системы.

“События” - эта страница предназначена для просмотра и создания событий в системе, начиная с запуска или остановки самого homeassistant и заканчивая изменением состояний, времени, запуском сервисов и прочего.

Вкладка «Supervisor» - оболочка, в которой работает HA.На вкладке “Панель” – видим наличие обновлений homeassistant, которых пока нет, и установленных дополнений – показаны на рисунке 10.

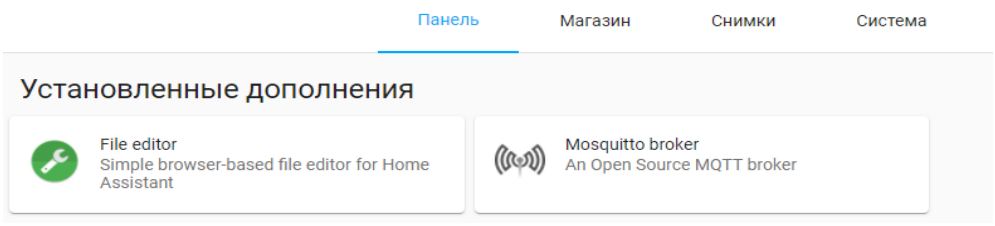

Рисунок 10 – Установленные дополнения через Supervisor

“Снимки” – снимки системы, т.е. резервные копии конфигурации. Их можно создать вручную, увидеть перечень снимков системы и восстановить конфигурацию.

“Магазин” – магазин дополнений, этакий плэймаркет. Отсюда ставятся полезные удобные дополнения.

Сразу можно установить file editor с функцией файлового менеджера. Он нам неоднократно пригодится. Ползунком включаем отображение на панели инструментов.

“Система” – видна текущая версия и операционная система. Ниже - лог событий на уровне системы.

На рисунке 11 можно увидеть «Настройки» - обладает наибольшим количеством вкладок. “Home Assistant Cloud” – предназначена для подключения к платному облаку.

“Интеграции” – служит для подключения различных служб и сервисов. А если нажать кнопку в левом нижнем углу ‘добавить интеграцию’, то увидим множество сторонних платформ. Это одно из преимуществ homeassistant – совместимость с ними.

Следующие 4 вкладки “Устройства”, “Автоматизации”, “Сценарии”, “Скрипты” – предназначены для управления сущностями в режиме конструктора.

“Общие” – в этой вкладке можно задать основные параметры системы.

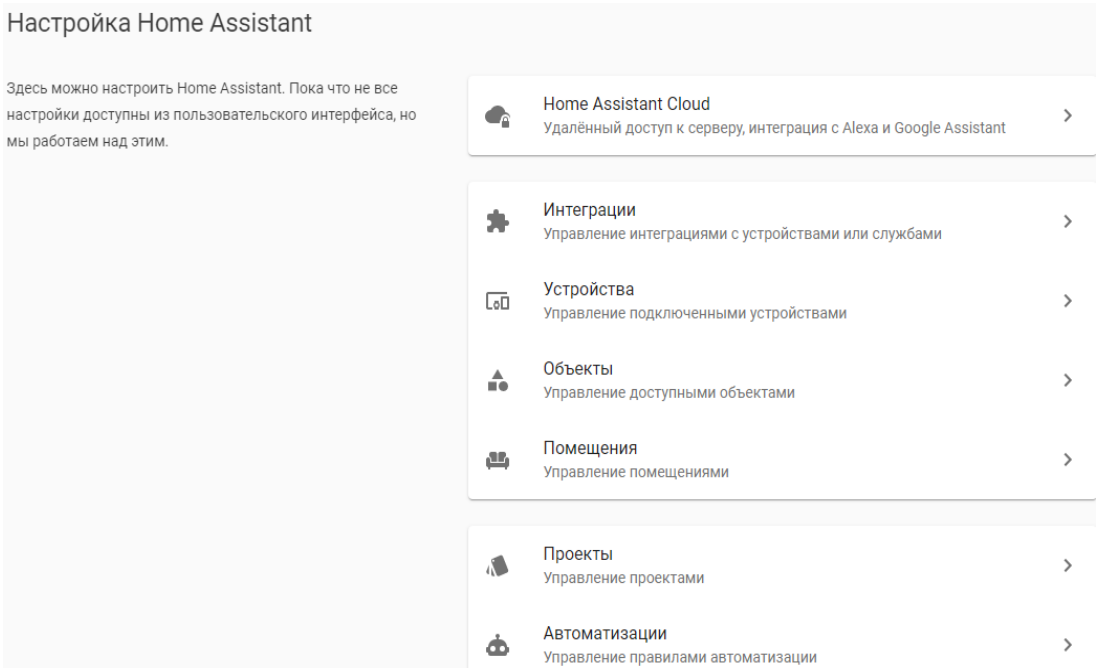

Рисунок 11 – Настройки Home Assistant

“Сервер” – управление сервером. Пока что тут одно окно перезапуска или остановки системы, но позже мы расширим этот раздел.

“Объекты” – в этом меню получаем доступ к перечню всех сущностей. Здесь можно изменить название идентификатора или удалить его совсем.

“Помещения” – позволяет создать список локаций.

“Пользователи” – меню показывает учетные записи тех, кто имеет права на управление или просмотр homeassistant.

«Уведомления» - в данном меню показываются системные сообщения. Их количество отображается на фоне оранжевого кружочка.

«Пользователь» - тут можно изменить язык интерфейса, изменить пароль, тема темная или светлая, цвет текста.

Еще тут находится переключатель ‘Расширенный режим’ - включим его. И в меню «Настройки» - “Сервер” увидим еще 2 окна: ‘Проверка и Перезагрузка конфигурации’ – в этом режиме и будем дальше работать.

Затем продолжим развивать HA в ручном [23] режиме – нажмем file editor - показано на рисунке 12. Нажав на кнопку в виде папки сверху, мы попадем в корневой каталог homeassistant “config/”.

Тут уже находится несколько файлов, откроем configuration.yaml – главный конфигурационный файл системы.

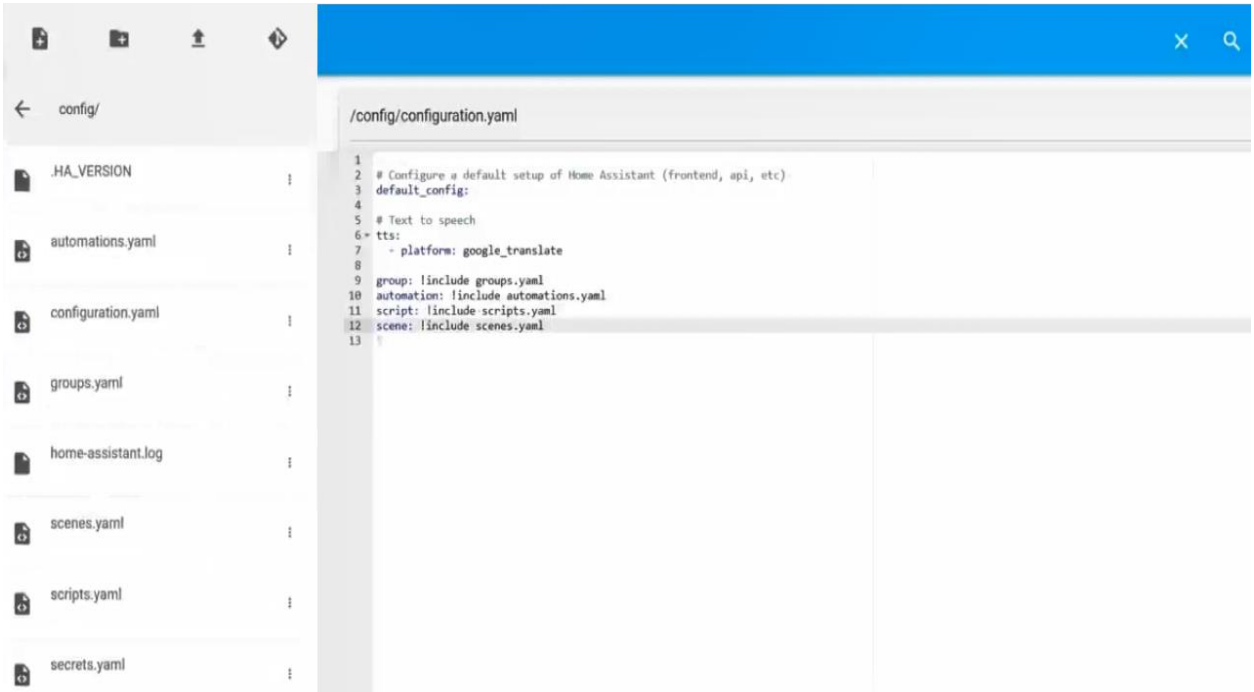

Рисунок 12 – File Editor

Все начинается именно с него: все настройки, описания сущностей, объектов автоматизации и скриптов.

Информация в этом файле расположена следующим образом: разделы – их названия указаны в отдельных строках без пробелов с левой стороны. Названия заканчиваются двоеточием.

То, что имеет отношение к разделу, пишется или правее или ниже и уже отделяется пробелами от левого края. Это правило работает для всех конфигурационных файлов. Пробелами определяется степень вложенности.

Как можно увидеть, то держать все настройки в одном файле довольно неудобно, так как занимает много места на экране. Поэтому тут поддерживается вынос в отдельные файлы или даже папки.

Например, разделы «группы, автоматизации, скрипты и сцены» уже по умолчанию вынесены во внешние файлы, о чем говорит команда include после двоеточия.

Если перед именем файла ничего не написано, то значит он находится в той же папке, что configuration.yaml. Их можно увидеть на рисунке 12, расположенном выше.

Также имеется файл secrets.yaml – он предназначен для хранения конфиденциальной информации, такой как пароли или ссылки.

Координаты и часовой пояс: для корректной работы нам необходимо ввести данные о своём местоположении, часовом поясе.

Местоположение, например, нужно для погодных сущностей, определении времени восхода и заката, а часовой пояс для правильной работы автоматизации, связанной с текущим временем.

Открываем основной раздел homeassistant, далее уже с отступами в этом разделе прописываем имя сервере – свое, любое.

Далее координаты, которые можно получить при помощи гугл-карт. После этого указываем высоту над уровнем моря – для Тольятти это 113 метров.

Единицы измерения – метрические. И в конце – часовой пояс. Его можно взять из списка на википедии (Europe/Samara).

Как только в файл вносятся изменения, то иконка сохранения в верхнем правом углу становится красного цвета. Это позволяет не забыть сохранить сделанные изменения.

Проверяем и перезагружаем конфигурацию. Для этого заходим в настройки – сервер и начать проверку и перезапустить.

Создадим несколько сенсоров: даты, времени и системного мониторинга. Сенсор – это тип информационных сущностей или объектов, которые могут иметь текстовые или числовые значения.

В configuration.yaml создадим раздел sensor, а далее, не забыв про отступы, прописываем сенсоры даты и времени.

Сначала указывается штатная платформа time_date, затем укажем сразу 3 сенсора: отдельно дата, время и время-дата в одном сенсоре.

Также добавим еще одну штатную [25],[26] платформу – системного мониторинга (systemmonitor).

Добавим следующие сенсоры: использование процессора в процентах (processor_use), процент используемого объема хранилища (disk_use_percent), аргумент слэш (arg/) – означает, что от корневой папки, использование оперативной памяти (memory_use_percent), время с последней загрузки (last_boot) и свободное место в гигабайтах тоже от корневой папки (disk_free, arg/) – показано на рисунке 13.

```
sensor:
  - platform: time_date
    display_options:
      - 'time'
      - 'date'
      - 'time_date'

  - platform: systemmonitor
    resources:
    - type: processor_use
    - type: disk_use_percent
      arg: /
    - type: memory_use_percent
    - type: last_boot
    - type: disk_free
      arg: /
```

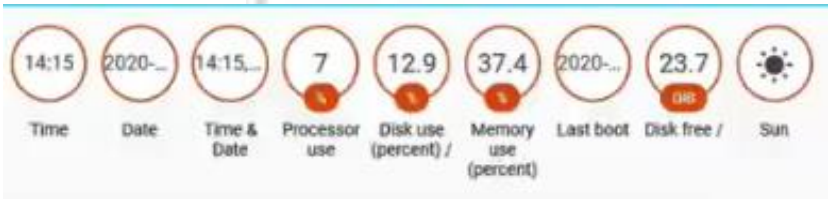

Рисунок 13 – Добавленные сенсоры и их вид на панели

Не забудем проверить правый верхний угол: если там зеленый кружочек с галочкой – значит все в порядке, если нет – то при нажатии на восклицательный знак на красном фоне, кликнув на него, получим строку с ошибкой.

Затем сохраняем, проверяем конфигурацию и перезагружаем. После перезагрузки в интерфейсе появились созданные сенсоры, отображенные на рисунке 13 выше.

Запустим file editor, откроем configuration,yaml, в котором укажем раздел lovelace и прописываем режим yaml (mode: yaml), сохраняя правила вложенности.

Сразу же после этого создаем файл для ручной настройки ui-lovelace.yaml. В самом верху укажем название, которое будет отображаться на всех страницах интерфейса и откроем раздел views.

Все дальнейшие записи будут включены в него.

Создадим первую страницу и затем укажем ее название «Главная», после чего укажем иконку и ее отображение (вторая строка) – их можно выбрать из каталога material icons – в него можно перейти сразу из окна file editor, нажав на шестеренку в верхнем правом углу.

В этом каталоге имеется огромное количество иконок и поддерживается поиск по названию.

Например, иконка home assistant. Нажимаем на нее, выделяем ее имя, копируем, возвращаемся в config и в строке icons:mdi пишем имя иконки без пробела. После чего сохраняем.

Откроем «Панель разработчика» - вкладка «Состояния». Отсюда удобно копировать имена сущностей. Создаем первую карту сущности.

В первой в первой строке ее название, во второй отключение общего переключателя, далее список отображаемых сущностей – соблюдая правила вложенности.

Ставим сенсор времени – параметр name позволяет написать любое отображаемое имя, аналогично заголовку страницы можно вставить собственную иконку – показано на рисунке 14.

Рисунок 14 – Создание карты сущности

Проверяем корректность конфигурации и перезагружаем. После перезагрузки открываем вкладку “Обзор” и наблюдаем созданную карточку и сенсор.

Все дальнейшие изменения будут появляться сразу – без перезагрузки. Из-за того что страница пока одна, то ее иконка отображаться не будет. Поэтому в новой вкладке запускаем file editor после чего, скопировав заголовок, создадим вторую вкладку с именем «Погода», на том же уровне вложенности. Выберем и добавим подходящую иконку.

Для применения обновления достаточно просто сохранить и обновить страницу. Теперь у нас есть две вкладки со своими иконками.

Продолжим заполнять первую страницу. Добавим сенсор даты.

Для сенсоров уровня загрузки применим дугой тип – измеритель (type: gauge). Создаем на том же уровне, что и карта сущностей.

Прописываем отображаемое имя (Загрузка ЦПУ), единицы измерения (%), указываем сущность – сенсор загрузки процессора (entity: sensor.ptocessor_use) и цвета для разграничения (severity: green 0 – от нуля, yellow 60 – от 60%, red 85 – от 85%).

Сохраняем – обновляем – проверяем. По нажатию на сенсор открывается его история.

Аналогично создаем ОЗУ – изменив название (Загрузка ОЗУ) и имя сенсора (sensor.memory_use_percent).

Проверяем результат – сенсоры слишком широкие. Лучше будет их совместить по горизонтали.

Для этого перед этими двумя картами объявим блок объединения по горизонтали для большей компактности, а измерительные карты вложим в него, сдвинув вправо.

Проверяем результат – информация та же самая, но теперь компактнее.

По созданным выше шаблонам создаем карту загрузки диска. Цвета оставим те же.

А у сенсора свободного места - обратная шкала – тут чем меньше, тем хуже, соответственно поменяем значения в зависимости от объема хранилища. И единицы измерения гигабайты.

После проверки и обновления видим, что получилось успешно, информативно и удобно.

Займемся установкой базы данных – установим MariaDB. Для этого переходим в магазин аддонов и выбираем базу MariaDB – отображено на рисунке 15.

Она довольно быстрая, удобная и несложно настраиваемая. Нажимаем установить и ждем. После установки идем в раздел конфигурации - тут изменим пароль. Сохраняем и запускаем.

Рисунок 15 – База MariaDB

Затем перейдем в file editor в основной config - configuration.yaml и изменим его, чтобы настройки записывались во внешние файлы и папки.

Создадим для конфигов отдельную папку includes. Добавим путь к ней от уже созданных файлов и создадим в ней файлы объявленных разделов: scenes, groups, scripts,automations.

Проверяем нет ли ничего в старых файлах и если нет, то удаляем их из внешних упоминаний, если есть, то переносим в новые файлы. Повторяем для всех разделов.

В файле configuration.yaml их ссылки выглядят так: #Вынос во внешние файлы group: !include includes/groups.yaml; automation: !include includes/automations.yaml; script: !include includes/scripts.yaml; scene: !include includes/scenes.yaml.

На примере созданного нами раздела сенсоров рассмотрим вынос в отдельную папку.

Сначала создадим ее. Внутри нее создаем новый текстовый файл. Назовем его system.yaml. В configuration.yaml пропишем ссылку на ту папку – к нему будет относиться содержимое всех файлов в этой папке: #Вынос во внешние папки sensor: !include_dir_merge_list includes/sensor.

File editor сейчас показывает ошибку. Вырезаем данные с созданными сенсорами и переносим их в файл сенсоров, чтобы не было ошибки. Сохраняем config configuration.yaml.

Открываем файл в папке сенсоров, куда мы перенесли ранее вырезанное – сам раздел sensor тут объявлять не нужно, т.к. он указан в главном конфиге.

Возвращаемся к базе данных – в configuration.yaml создаем раздел recorder с выносом во внешний файл и создаем его в папке includes, а внутри него указываем путь к базе данных и количество дней, которые она хранит историю db_url: mysql://hass:hass@core mariadb / homeassistant ? charset = utf8; purge_keep_days: 3.

Проверяем корректность сделанных изменений. Перед перезагрузкой добавим сенсор объема базы данных. Открываем папку includes, далее sensor.

В файл с сенсорами добавим еще один, указав тот же путь как и в разделе recorder. Проверяем конфигурацию и перезагружаем.

После перезагрузки появится новый сенсор, показывающий объем базы данных в мегабайтах.

Выведем его в интерфейс, указав имя, иконку и сущность. Наши карты уже разошлись на два столбца.

Для того, чтобы столбцы делились так, как нам надо, добавим блок горизонтали – подпишем карты используя ввод форматированного текста.

Обратим внимание на уровень вложенности. Затем, чтобы карты на столбцы делились именно там, где нам нужно, используем блок объединения по вертикали. Его уровень вложенности должен быть выше, чем у описанных карт, но ниже чем у объявленного раздела cards.

Добавим график для базы – чтобы показать, что она работает и наполняется. Это карта исторических значений, которая своим ростом покажет, что база работает и наполняется – представлено на рисунке 16.

Рисунок 16 – График базы MariaDB

Вернемся в configuration.yaml и закончим настройку сенсора погоды. Найдем его в панели состояний и создадим погодную карту на ранее созданной странице погоды – показано на рисунке 17.

```
- title: Погода
  icon: mdi:weather-partly-snowy-rainy

  cards:

      - type: weather-forecast
        entity: weather.home_assistant
```

Рисунок 17 – Код для сенсора погоды

Обновляем и получаем ту же карту, что была в автоматическом режиме, но теперь она находится там, где мы для нее прописали место – на второй вкладке – представлена на рисунке 19, а также отображение иконок вкладок в НА – их пока 2. На рисунке 18 – первая вкладка интерфейса.

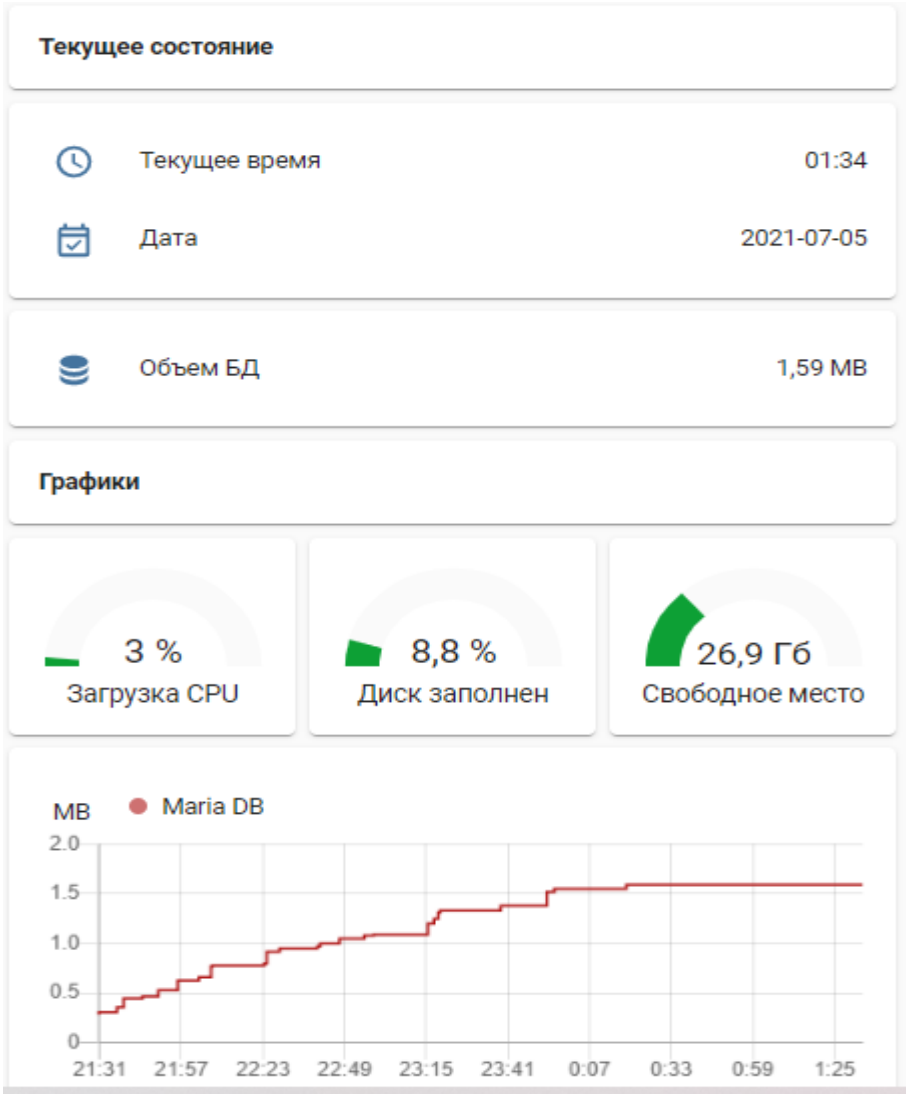

Рисунок 18 – Главная страница интерфейса

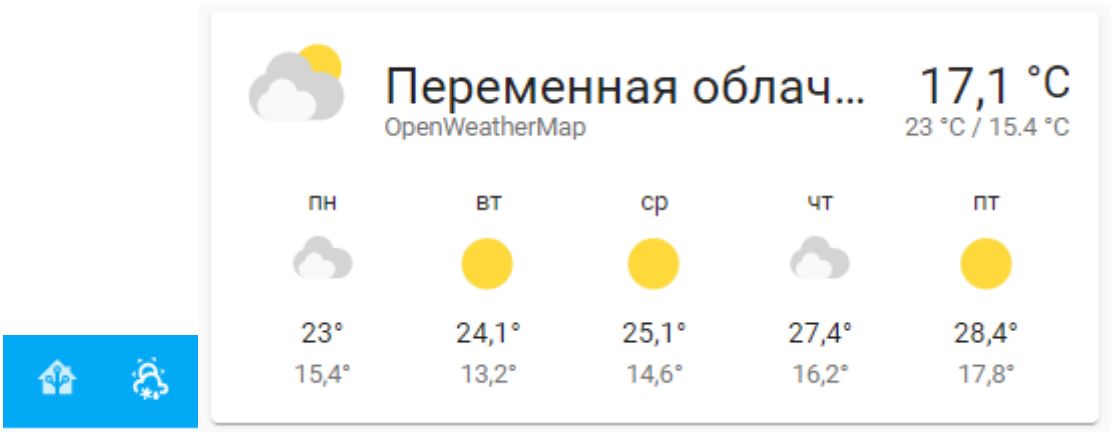

Рисунок 19 – Вкладка «Погода» и иконки интерфейса

После чего продолжим про настройку HA, в частности для добавления в него ZigBee-устройств. Для этого настроим MQTT.

Установим аддон Mosquitto Broker – он находится в списке стандартных интеграций. Во вкладке конфигурации укажем логин и пароль. Сохраняем и запускаем.

После чего переходим в «настройки» - «интеграции», где добавляем новую интеграцию и находим MQTT. В строке брокер укажем core-mosquitto, т.к. у нас подключается локально.

Затем нажав на 3 точки увидим репозиторий и вставим следующую ссылку[11]. Появился новый репозиторий ZigBee2mqtt – установим его. После чего во вкладке конфигурация укажем ранее введенные логин и пароль. Сохраняем и запускаем интеграцию – представлено на рисунке 20.

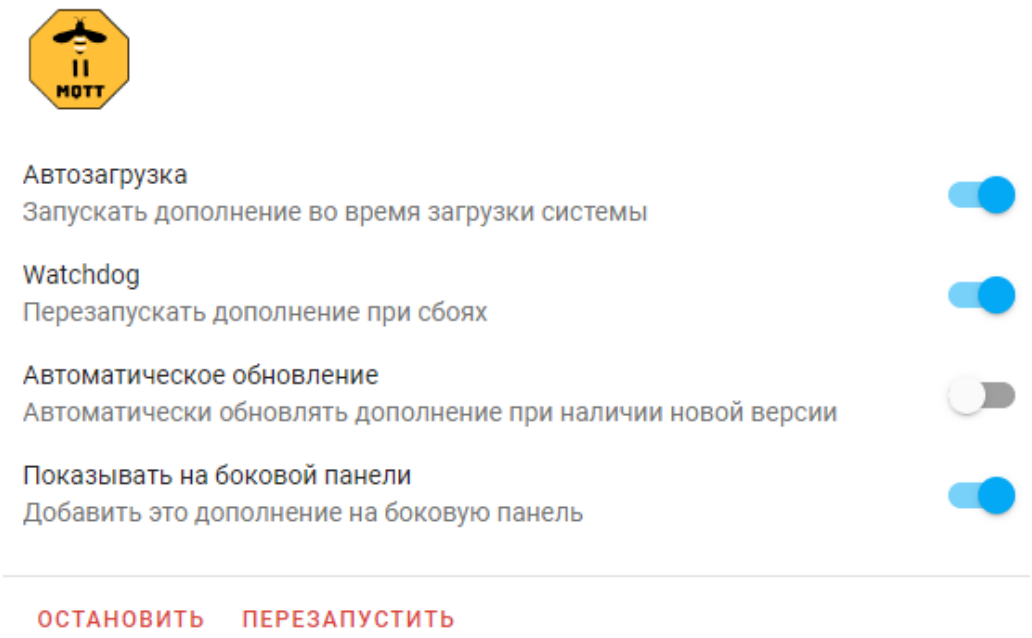

Рисунок 20 – Аддон ZigBee2mqtt

Во вкладке «информация» нажмем «watchdog» и «показать в панели сбоку». Переходим в нее и видим меню с несколькими кнопками: setting – можно узнать версию стика, devices – подключаемые устройства, а самая правая permit join – позволит нам подключать устройства – отображено на рисунке 21.

Если все прошло успешно, то можно увидеть его в строке устройств. Процесс сопряжения может занять некоторое время. Во вкладке «карта» можно увидеть подключенные устройства, как конечные (зеленые линии) так и роутеры (синие линии).

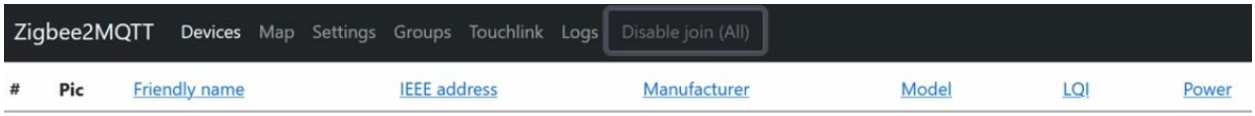

Рисунок 21 – Интерфейс ZigBee2mqtt

Для удаления устройств нужно нажать красную кнопку и тогда оно исчезнет. Стоит отметить опцию форсированного удаления. Если ее выбрать, то устройство удалится и из интеграции MQTT.

Вторая опция при необходимости внесет устройство в блок-лист и запретит подключаться вновь – представлено на рисунке 22.

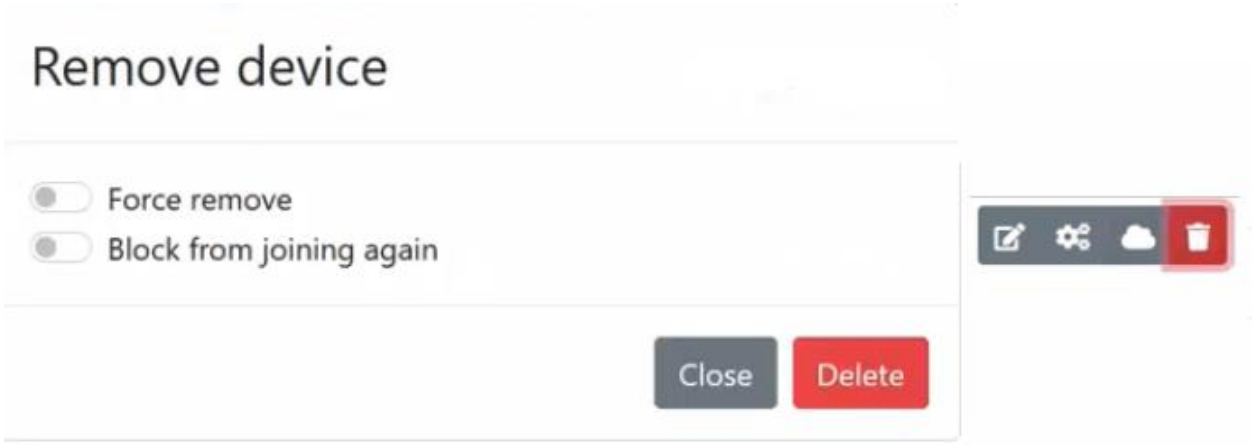

Рисунок 22 – Удаление устройств из ZigBee2mqtt

В итоге, как можно увидеть по представленной информации выше, то предварительный процесс для создания дальнейшей программной части исследования проведен успешно. Создана основная часть для помощника, на базе которой дальше будет строиться следующее программирование.

4 Необходимые компоненты для работы

4.1 ConBee ii

Универсальный USB-шлюз ZigBee. Предназначен в качестве посредника между вашими ZigBee-устройствами и ПК или ноутбуками [13],[43].

4.1.1 Особенности

4.1.1.1 Работает без облака

Показано на рисунке 23.

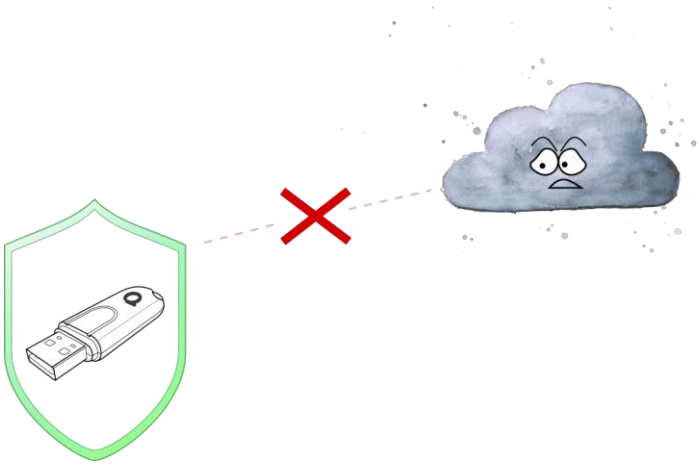

Рисунок 23 – ConBee ii работа локально

- не требуется регистрация;
- локальное управление устройствами, без интернета;
- все данные, такие как состояние освещения, датчики и переключатели, остаются в локальной сети;

4.1.1.2 Совместимость

Совместим с широким спектром источников света, переключателей и датчиков – представленных на рисунке 24 от:

- Philips Hue;
- IKEA TRÅDFRI;

– Xiaomi Aqara;
– OSRAM SMART+;
– Busch-Jaeger;
– GIRA;
– JUNG;
– Paulmann;
– Paul Neuhaus;

Рисунок 24 – Совместимость ConBee ii

4.1.1.3 Независимая платформа

Превращает мини-компьютеры, такие как Raspberry PI и Intel NUC, а также ПК и ноутбуки в универсальные шлюзы ZigBee

4.1.1.4 Простая миграция платформы

Используя функцию резервного копирования приложения Phoscon, можно перенести установку Zigbee в новую систему. Например, вы можете перенести существующую установку с Raspberry Pi на более мощный Intel NUC.

4.1.1.5 Системы домашней автоматизации

ConBee II был интегрирован сообществом в самые популярные системы умного дома.

Мощная интеграция [48] позволяет реализовать захватывающую автоматизацию и функции.

- Homebridge-hue;
- Home Assistant;
- deCONZ Docker Container;
- Node-RED;
- Node-RED;
- FHEM;
- ioBroker;
- Jeedom;
- JowiHue for HomeSeer;
- Domoticz;
- openHAB;
- Pimatic;
- Homey;
- AI-Speaker;
- WebThings;
- Nymea;
- Symcon;
- HOOBS;

4.1.1.6 Диапазон сигнала

- До 30 метров в зданиях;

– До 200 метров в прямой видимости;

Благодаря усилителю мощности [45] сигнал способен проходить через 2-3 этажа или помещения – показано на рисунке 25.

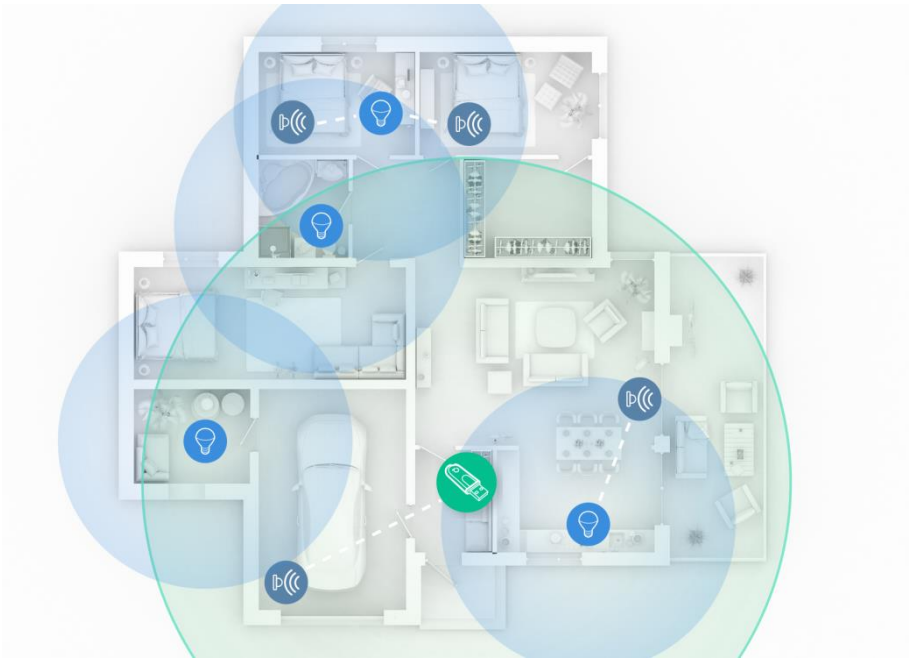

Рисунок 25 – Распространение сигнала ConBee ii

Устройства дальше подключены через сеть Zigbee Mesh . В этой сети все устройства Zigbee [51] с питанием от сети, такие как светильники и розетки, действуют как повторители и могут маршрутизировать сигнал.

4.2 Arduino Pro Micro

Китайский клон Arduino Micro [20],[46] со встроенным USB. Опознается как Arduino Leonardo – представлено на рисунке 26 . Принципиальная схема в приложении Б.

Рисунок 26 - Arduino Pro Micro

Технические характеристики устройства приведены в таблице 4.

Таблица 4 – Технические характеристики

Микроконтроллер	ATMega32u4
Напряжение питания	5 В
Цифровые входные/выходные выводы	12
ШИМ каналы	5
Аналоговые входные выводы	4
Постоянный ток через входные/выходные выводы	40 мА
Флеш-память	32 Кб (ATmega32u4), из которых 4 Кб используются загрузчиком
Оперативная память SRAM	2,5 Кб
Энергонезависимая память EEPROM	1 Кб
Тактовая частота	16 МГц
Длина	33 мм
Ширина	18 мм
Вес	5 г

Расположение выводов [54] Arduino Pro Micro показано на рисунке 27.

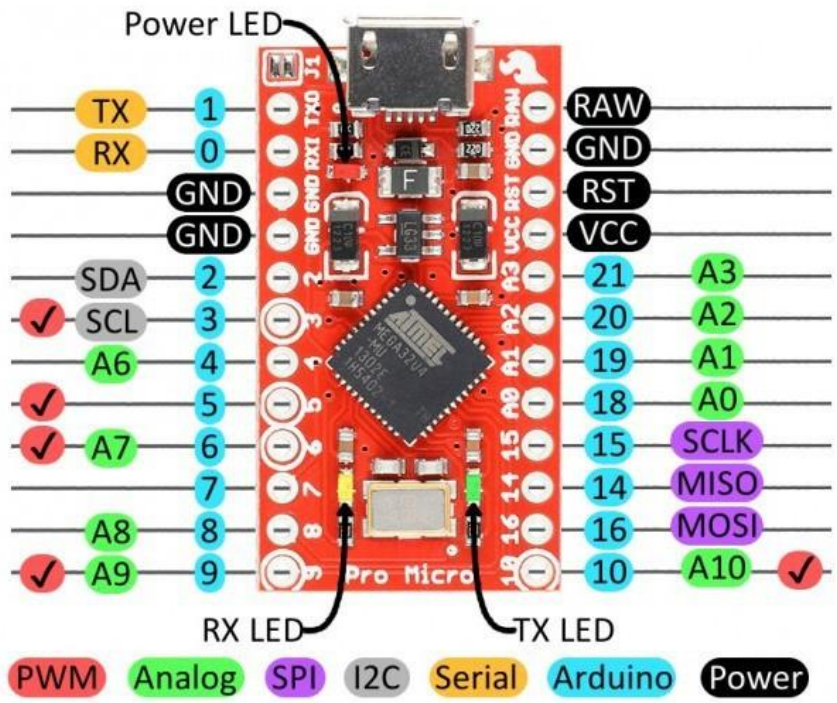

Рисунок 27 – Расположение выводов

4.3 Air Conditioner

В качестве испытуемого объекта был взят кондиционер Hitachi, модель RAS-LJ22Y, год выпуска 2009. Он был выбран по причине своей доступности. Он оснащен системой домашней автоматизации, которая позволяет дистанционно включать или выключать его, а также знать его состояние питания. Именно к нему мы присоединим наше устройство, для того чтобы дистанционно им управлять.

Таким образом, все необходимые компоненты аппаратной части были собраны в одном месте и описаны выше.

5 Соединение программной и аппаратной части

Сняв крышку с описанного в пункте выше устройства, и пользуясь документацией (14) присоединили провода к пинам его материнской платы, чтобы по 4-хпроводной схеме UART тот понимал нас, а мы через Arduino передавали эту информацию на E-18.

Затем используя инструмент с графическим интерфейсом для создания собственного ZigBee-устройства с гибкой конфигурацией входов и выходов (15). Внимательно изучили его успешно отправили, а затем и получили данные от Home Assistant.

Для начала перешли с deConz на ZigBee2mqtt с использованием брокера Mosquitto. Затем, используя описанный выше инструмент создали прошивку на UART со скоростью 9600 бод. После чего залили данную прошивку на наш модуль.

В homeassistant на панели инструментов нажимаем разрешить обнаружение устройств, «присоединение (все)». Затем подключили модуль ZigBee по UART к hitachi, включили его и открыли termite на своем ПК.

Через некоторое время (около половины минуты) ZigBee2mqtt обнаружил новое устройство. Переименовыввем его в climate и добавляем в homeassistant. Затем в разделе конфигурации – интеграции – брокер Mosquitto – настроить и в разделе опубликовать пишем «тест». Тоже самое появляется в окне на ПК

После чего в homeassistant в разделе прослушать нажали на начать прослушивание. На своем ПК в окне пишем «тест» и этот же самый текст появляется в homeassistant. Затем через графический инструмент создаем прошивку со следующими настройками:

Тип платы = CC2530, Тип устройства = Конечное устройство без маршрутизации, Светодиод состояния = P22, отправка отчетов, Выход 1 = P03, UART, скорость передачи = 9600, конец пакета = 0x0D, ID модели =

RAS-LJ22Y_ADPT (настройка уникального имени), Установите интервал (ы) отчетности по умолчанию = 60, Обновить временную метку прошивки

После чего сохранили пользовательский конвертер для Z2M, чтобы создать внешний конвертер соответствующий для ZigBee2mqtt. После чего загрузив скетч на модуль, включили его, а затем в панели инструментов Z2M в HA добавили его. Изначально оно появилось как неподдерживаемое, но следуя инструкции (16) успешно его добавили.

Затем ранее созданный внешний преобразователь переносим в config\zigbee2mqtt вернувшись на панель инструментов Z2mqtt заходим в настройки – внешние конвертеры, вводим свое имя конвертера, жмем отправить и перезапускаем. Результат показан на рисунке 28.

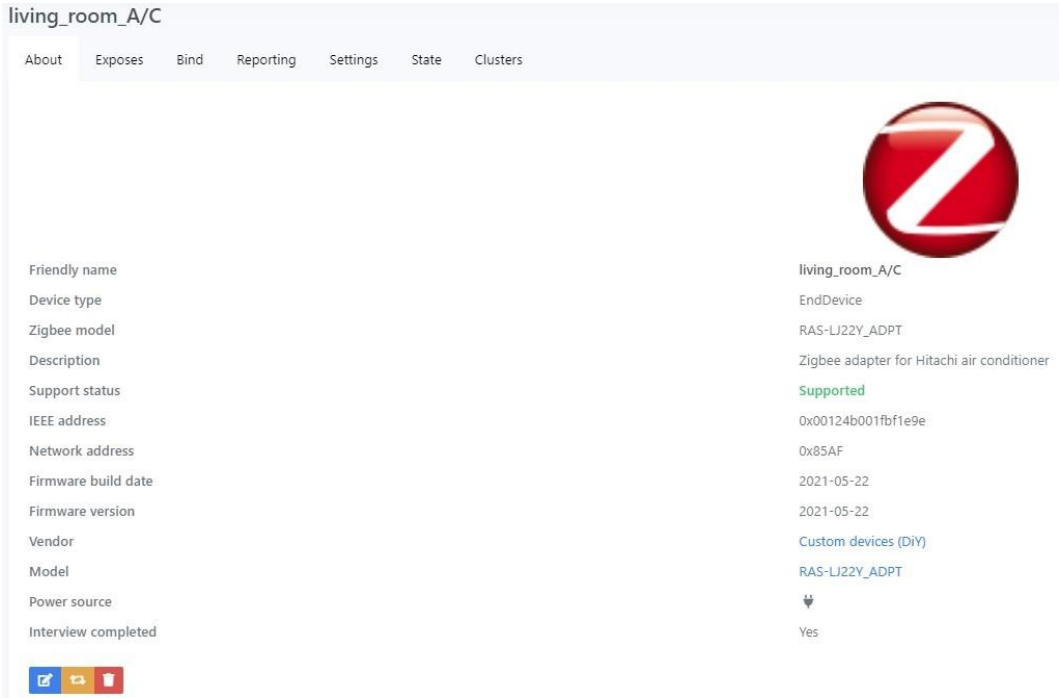

Рисунок 28 – Результат загрузки конвертера

После чего, следуя документации (17), добавляем климатическую платформу в свой configuration.yaml. И видим там следующий код:

```
# Climate settings for HITACHI HVAC
climate:
  - platform: mqtt
    name: HITACHI_HVAC
    modes:
      - "off"
      - "heat"
      - "cool"
      - "dry"
      - "fan_only"
    fan_modes:
      - "auto"
      - "high"
      - "medium"
      - "low"
    max_temp: 32
    min_temp: 16
    payload_off: 0
    payload_on: 1
    fan_mode_command_topic: "zigbee2mqtt/living_room_A/C/set/action"
```

fan_mode_command_template: "FAN:{{ value }}"

fan_mode_state_topic: "zigbee2mqtt/living_room_A/C/action"

fan_mode_state_template: "{{ value_json.FAN }}"

mode_command_topic: "zigbee2mqtt/living_room_A/C/set/action"

mode_command_template: "MODE:{{ value }}"

mode_state_topic: "zigbee2mqtt/living_room_A/C/action"

mode_state_template: "{{ value_json.MODE }}"

temperature_command_topic: "zigbee2mqtt/living_room_A/C/set/action"

temperature_command_template: "TMP:{{ value }}"

temperature_state_topic: "zigbee2mqtt/living_room_A/C/action"

temperature_state_template: "{{ value_json.TMP }}"

current_temperature_topic: "zigbee2mqtt/living_room_T/H_sensor"

current_temperature_template: "{{ value_json.temperature }}"

send_if_off: false

precision: 1.0

После чего была написана программа, которая получает состояние кондиционера каждые 2 секунды , ищет разницу с текущим значением, и если есть различия, то шлет это значение в HA. Кроме того, данная программа отправляет команды настройки кондиционеру в соответствии с тем, что было получено от HA. Код ниже

#include <SoftwareSerial.h>

```
SoftwareSerial ZigBee_serial(0, 1); // 0->RX, 1->TX

const int ZigBee_LED = 2;

char HVAC_data[17];

char ZigBee_data[15];

char ZigBee_data_cmd[5];

char ZigBee_data_value[10];

char payload_to_HVAC[23];

char payload_to_ZigBee[60];

char *index_t;

unsigned long start_millis;

unsigned long current_millis;

unsigned long intervall = 2000;

int size_data2;

boolean value_changed = false;

boolean init_state = false;

byte temp_value = 0;

const char mode_list[6][9] = {

  {"off"},      //0x0000

  {"heat"},     //0x0010

  {"dry"},      //0x0020
```

```
  {"-"},
  {"cool"},     //0x0040
  {"fan_only"}, //0x0050
};
const char fan_list[4][7] = {
  {"auto"},   //0x00
  {"high"},   //0x01
  {"medium"}, //0x02
  {"low"},    //0x02
};

uint16_t address_list[]=
{
  0x0000, //power (ON, OFF)
  0x0001, //mode (hot, dry, cool, fan)
  0x0002, //fan speed (auto, high, medium, low)
  0x0003, //temperature (16~32°C)
  0x0100  //inside temperature
};
struct HVAC_status {
```

```
  byte power;
  byte mode;
  byte fan_speed;
  byte temperature;
  byte inside_temp;
} Hitachi_status;
void LED_ISR(){
  noInterrupts();
  digitalWrite(LED_BUILTIN, !digitalRead(ZigBee_LED));
  interrupts();
}
void setup() {
 pinMode(LED_BUILTIN, OUTPUT);
 pinMode(ZigBee_LED, INPUT);
 digitalWrite(LED_BUILTIN, !digitalRead(ZigBee_LED));
 attachInterrupt(digitalPinToInterrupt(ZigBee_LED), LED_ISR, CHANGE);
 ZigBee_serial.begin(9600);
 Serial1.begin(9600, SERIAL_8O1); // baud raute: 9600, data size: 8 bits, parity: odd, stop bits: 1
 size_t a = Serial1.readBytesUntil('\r', HVAC_data, 17); // discard the message [NG P=01 C=FFFE\r] received from HVAC after initialization of Serial1
```

```
 delay(20);
Hitachi_status.power = inquire_HVAC_status(address_list[0]);
 Hitachi_status.mode = inquire_HVAC_status(address_list[1]);
 Hitachi_status.fan_speed = inquire_HVAC_status(address_list[2]);
 Hitachi_status.temperature = inquire_HVAC_status(address_list[3]);
 Hitachi_status.inside_temp = inquire_HVAC_status(address_list[4]);
 send_HVAC_status();
start_millis = millis();
}
void loop(){
  if(ZigBee_serial.available() > 0) {
    size_data2 = ZigBee_serial.readBytesUntil('\r', ZigBee_data, 15);
    ZigBee_data[size_data2] = '\0';
    index_t = strchr(ZigBee_data, ':');
    *index_t = '\0';
    strcpy(ZigBee_data_cmd, ZigBee_data);
    strcpy(ZigBee_data_value, index_t + 1);
    if(strcmp(ZigBee_data_cmd,"MODE") == 0){
      if(strcmp(ZigBee_data_value, mode_list[0]) == 0){
        Hitachi_status.power = 0x00;
```

```
    set_HVAC_status(address_list[0],0x00);
  }
  else{
    for(int i = 1; i < 6; i++){
      if(strcmp(ZigBee_data_value, mode_list[i]) == 0){
        Hitachi_status.mode = i << 4;
        Hitachi_status.power = 0x01;
        set_HVAC_status(address_list[1],Hitachi_status.mode);
        set_HVAC_status(address_list[0],1);
      }
    }
  }
}
else if(strcmp(ZigBee_data_cmd,"FAN") == 0){
  for(int j = 0; j < 4; j++){
    if(strcmp(ZigBee_data_value, fan_list[j]) ==0){
      Hitachi_status.fan_speed = j;
      set_HVAC_status(address_list[2],Hitachi_status.fan_speed);
    }
  }
```

```
  }
  else if(strcmp(ZigBee_data_cmd,"TMP") == 0){
    index_t = strchr(ZigBee_data_value, '.');
    *index_t = '\0';
    Hitachi_status.temperature = atoi(ZigBee_data_value);
    set_HVAC_status(address_list[3],Hitachi_status.temperature);
  }
  send_HVAC_status();
}
current_millis = millis();
if(current_millis - start_millis >= intervall){
  start_millis = millis();
  temp_value = inquire_HVAC_status(address_list[0]);
  if(temp_value != Hitachi_status.power){
    value_changed = true;
    Hitachi_status.power = temp_value;
  }
  temp_value = inquire_HVAC_status(address_list[1]);
  if(temp_value != Hitachi_status.mode){
    value_changed = true;
```

```
  Hitachi_status.mode = temp_value;
}
temp_value = inquire_HVAC_status(address_list[2]);
if(temp_value != Hitachi_status.fan_speed){
  value_changed = true;
  Hitachi_status.fan_speed = temp_value;
}
temp_value = inquire_HVAC_status(address_list[3]);
if(temp_value != Hitachi_status.temperature){
  value_changed = true;
  Hitachi_status.temperature = temp_value;
}
temp_value = inquire_HVAC_status(address_list[4]);
if(temp_value != Hitachi_status.inside_temp){
  value_changed = true;
  Hitachi_status.inside_temp = temp_value;
}
if(value_changed == true){
  send_HVAC_status();
  value_changed = false;
```

```
    }
    temp_value = 0;
  }
}
byte inquire_HVAC_status(uint16_t address_to_send){
  int size_data;
  byte cs_buffer;
  byte status_value;
  char buffer1[5];
  char *ptr;
  strcpy(payload_to_HVAC,"MT P=0");
  if(address_to_send == 0x0100){
    strcat(payload_to_HVAC,"100");
  }
  else{
    strcat(payload_to_HVAC,"00");
    itoa(address_to_send, buffer1, 16);
    strcat(payload_to_HVAC, buffer1);
  }
  strcat(payload_to_HVAC," C=FF");
```

```
if(address_to_send == 0x0100){
  cs_buffer = 0xFF - 1;
  }
else{
  cs_buffer = 0xFF - address_to_send;
  }
buffer1[0]=0;
itoa(cs_buffer,buffer1,16);
strcat(payload_to_HVAC, buffer1);
strupr(payload_to_HVAC);
strcat(payload_to_HVAC,"\r");
Serial1.print(payload_to_HVAC);
Serial1.flush();
size_data = Serial1.readBytesUntil('\r', HVAC_data, 17);
ptr = strtok (HVAC_data, "=");
ptr = strtok (NULL, " ");
status_value = (byte) strtoul(ptr, NULL, 16);
delay(20);
return status_value;
```

```
}
void send_HVAC_status(){
  char convert_buffer2[5];
  strcpy(payload_to_ZigBee, "{\"MODE\":\"");
  if(Hitachi_status.power == 0){
    strcat(payload_to_ZigBee, mode_list[0]);
    }
  else {
    strcat(payload_to_ZigBee, mode_list[Hitachi_status.mode >> 4]);
    }
  strcat(payload_to_ZigBee, "\",\"FAN\":\"");
  strcat(payload_to_ZigBee, fan_list[Hitachi_status.fan_speed]);
  strcat(payload_to_ZigBee, "\",\"TMP\":");
  itoa(Hitachi_status.temperature, convert_buffer2, 10);
  strcat(payload_to_ZigBee,convert_buffer2);
  strcat(payload_to_ZigBee,",\"IN_TMP\":");
  convert_buffer2[0]=0;
  itoa(Hitachi_status.inside_temp, convert_buffer2, 10);
  strcat(payload_to_ZigBee,convert_buffer2);
  strcat(payload_to_ZigBee,"}\r");
```

```
  ZigBee_serial.print(payload_to_ZigBee);
  ZigBee_serial.flush();
}
void set_HVAC_status(uint16_t address_to_send, byte value_to_send){
  char convert_buffer[5];
  byte value_buffer;
  strcpy(payload_to_HVAC, "ST P=000");
  itoa(address_to_send, convert_buffer, 10);
  strcat(payload_to_HVAC, convert_buffer);
  strcat(payload_to_HVAC, ",0");
  if(address_to_send == 0x0001 or address_to_send == 0x003){
    strcat(payload_to_HVAC, "0");
  }
  convert_buffer[0]=0;
  itoa(value_to_send, convert_buffer, 16);
  strcat(payload_to_HVAC, convert_buffer);
  strcat(payload_to_HVAC, " C=FF");
  value_buffer = 0xFF - address_to_send - value_to_send;
  convert_buffer[0]=0;
  itoa(value_buffer, convert_buffer, 16);
```

```
  strcat(payload_to_HVAC, convert_buffer);
  strupr(payload_to_HVAC);
  strcat(payload_to_HVAC, "\r");
  Serial1.print(payload_to_HVAC);
  Serial1.flush();
  size_t z = Serial1.readBytesUntil('\r', HVAC_data, 17);
  delay(20);
}
```

Затем добавляем термостатную карту в lovelace и устанавливаем «сущность» на созданную климатическую сущность – показано на рисунке 29.

Рисунок 29 – Отображение термостатной карты

Окончательный вариант схемы подключения устройств выглядит на рисунке 30. Нижняя часть схемы подключается к кондиционеру.

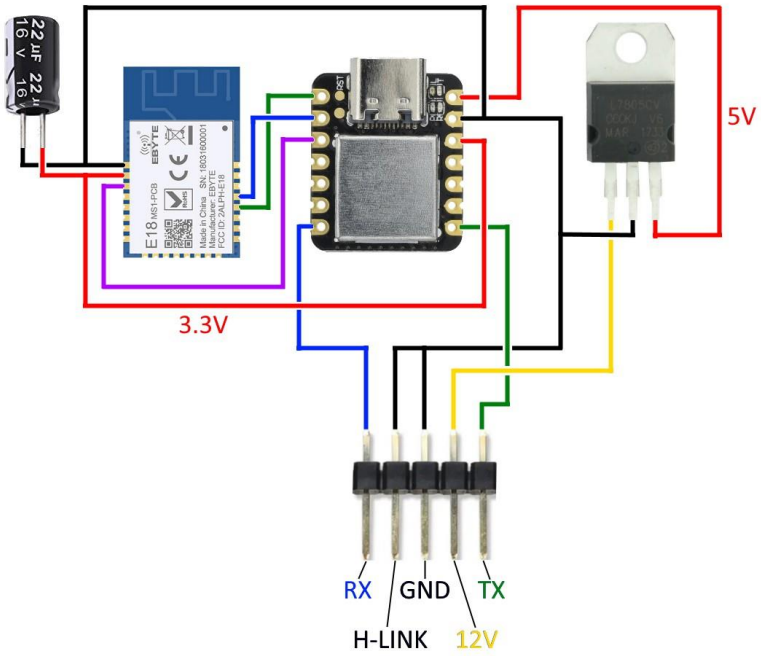

Рисунок 30 – Подключение устройств друг к другу

В результате вышеописанных действий получаем готовое устройство, которое способно не только отслеживать и передать значения в режиме реального времени, но и реагировать на желание пользователя по изменению своего состояния в заданных пределах.

6 Экономическая часть

Произведем подсчет средств на данные устройства в таблице 5

Таблица 5 – Стоимость компонентов

Наименование	Цена
Универсальный USB-шлюз Conbee ii с учетом бесплатной доставки	2320.07р
Беспроводной приемо-передатчик E-18-MS1-PSB с чипом cc2530	222.69р
Arduino Pro Micro	94.49
Конденсатор 22 uF 16В	6.5р
Стабилизатор напряжения +5В	45р

Таким образом, самой дорогостоящей [57],[58] частью проекта, не считая кондиционер, является usb-шлюз. Что может являться одной из причин усовершенствования проекта, кроме того, что с каждым годом появляется большое количество разных новинок, некоторые из которых по своему соотношению цена – качество могут составить конкуренцию выбранным на данный момент.

Заключение

Таким образом, модуль технологии ZigBee вполне соответствует заявленным характеристикам для использования в домашней автоматизации. И по праву занимает свою нишу среди малопотребляющих устройств, способных к самовосстановлению своей сети в случае неполадок, а также для ее масштабирования.

Как можно увидеть по представленной информации выше, то была создана основная часть для помощника, на базе которой было построено дальнейшее программирование.

Описаны составляющие аппаратной части, а также их совокупная сборка показанная в приложении В.

В результате вышеописанных действий получаем готовое устройство, которое способно не только отслеживать и передать значения в режиме реального времени, но и реагировать на желание пользователя по изменению своего состояния в заданных пределах.

Самой дорогостоящей частью проекта, не считая кондиционер, является usb-шлюз. Что может являться одной из причин по усовершенствования проекта, кроме того, что с каждым годом появляется большое количество разных новинок, некоторые из которых по своему соотношению цена – качество могут составить конкуренцию выбранным на данный момент.

Соответственно, данная технология крайне нетребовательна к питанию, особенно конечные устройства, многие из которых могут на одной батарейке проработать до 3 лет, находясь в режиме сна и периодически включаясь.

Список используемой литературы и используемых источников

1. Аникин А.П. Обзор современных технологий беспроводной передачи данных в частотных диапазонах ISM. //Беспроводные технологии №4, 2011.
2. Барашко, О.Г. Проектирование систем домашней автоматизации: учеб. пособие. / О. Г. Барашко, А. В. Овсянников. – «Белорусский государственный технологический университет», 2016 – 57с
3. Коптев Д.С., Щитов А.Н., Шевцов А.Н. Сравнительный анализ наиболее перспективных стандартов беспроводных сетей связи // Международный журнал гуманитарных и естественных наук. 2016. №1. URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-naibolee-perspektivnyhstandartov-besprovodnyh-setey-svyazi
4. [Электронный ресурс]: URL - Alex Kvazis: Video Lessons Home Assistant
5. [Электронный ресурс]: URL - https://github.com/kvazis/training/blob/master/lessons/unit%205/unit_5_0_2021.tx
6. [Электронный ресурс]: URL - https://ptvo.info/zigbee-configurable-firmware-features/
7. [Электронный ресурс]: URL - https://phoscon.de/en/conbee2
8. [Электронный ресурс]: URL - https://www.jst-mfg.com/product/pdf/eng/ePA-F.pdf
9. https://ptvo.info/zigbee-configurable-firmware-features/
10. [Электронный ресурс]: URL - https://www.zigbee2mqtt.io/how_tos/how_to_support_new_devices.html
11. [Электронный ресурс]: URL - https://www.home-assistant.io/integrations/climate.mqtt/
12. Research on Smart Home System Based on SC2440 and ZIGBEE Protocol 2018, Gong Lanfang

13. [Электронный ресурс]: документация BLE112 Datasheet, 2020.
14. Официальный сайт фирмы Arduino Software [Электронный ресурс] URL: http://arduino.cc
15. Используем ZigBee в Home assistant // indahomekit.ru URL: https://www.indahomekit.ru/2019/01/10/ispolyzuem-zigbee-v-home-assistant/
16. [Электронный ресурс]: URL - https://habr.com/ru/company/iobroker/blog/495926/
17. RELIABLE DATA BROADCAST FOR ZIGBEE WIRELESS SENSOR NETWORKS 2010, Tien-Wen Sung, Ting-Ting Wu, Chu-Sing Yang, Yueh-Min Huang3
18. [Электронный ресурс]: URL - https://github.com/diyruz/diyruz_rt
19. [Электронный ресурс]: URL - https://ptvo.info/zigbee-switch-configurable-firmware-v2-210/
20. ZigBee's Received Signal Strength and Latency Evaluation under Varying Environments 2016, H. H. R. Sherazi, Razi Iqbal, Sana Ul Hassan, M. H. Chaudary, Syed Asfandyar Gilani
21. [Электронный ресурс]: URL - https://ptvo.info/zigbee-switch-configurable-firmware-v2-210/
22. Design of Intelligent Vehicle Monitoring System Based on ZigBee 2018, Ma Jiandong
23. [Электронный ресурс]: URL - https://ptvo.info/zigbee-configurable-firmware-features/
24. A Review on Communications Perspective of Flying Ad-Hoc Networks: Key Enabling Wireless Technologies, Applications, Challenges and Open Research Topics 2020, Fazal Noor, Muhammad Asghar Khan, Ali Al-Zahrani, Insaf Ullah and Kawther A. Al-Dhlan
25. [Электронный ресурс]: URL - https://www.cnx-software.com/2020/05/25/zigbee-firmware-news-ti-z-stack-3-0-zigbee-for-cc2530-ptvo-zigbee-fw-configuration-tool/

26. [Электронный ресурс]: URL - https://cnx-software.ru/2020/05/25/novosti-o-proshivke-zigbee-ti-z-stack-3-0-zigbee-dlya-cc2530-i-nastraivaemaya-proshivka-ptvo-zigbee-fw/
27. SMART INTERNET CONNECTED MOBILE PHONE REMOTE FOR MONITORING AND CONTROLLING OF HOUSE AND HOUSEHOLD APPLIANCES 2017, Pandiaraj R, Rani Hemamalini R
28. [Электронный ресурс]: URL - https://e2e.ti.com/support/wireless-connectivity/zigbee-thread-group/zigbee-and-thread/f/zigbee-thread-forum/610754/cc2530-uart-and-blinky-code
29. Performance Analysis of Smart Energy Monitoring Systems in Real-time 2020, R. Govindarajan, S. Meikandasivam, D. Vijayakumar
30. [Электронный ресурс]: URL - https://www.youtube.com/watch?v=qKZhKKZO-GU&ab_channel=%D0%A1%D0%B5%D1%80%D0%B6%D0%92%D0%B0%D1%81%D0%B8%D0%BB%D1%8C%D0%B5%D0%B2
31. [Электронный ресурс]: документация https://www.manualslib.com/manual/1509997/Ebyte-E18-Series.html
32. [Электронный ресурс]: URL - https://github.com/Koenkk/zigbee-herdsman/issues/32
33. [Электронный ресурс]: документация - WizFi210. User’s Manual. V. 1.0.
34. Wireless Home Energy Management System with Smart Rule-Based Controller 2020, Hussain Shareef, Eslam Al-Hassan, Reza Sirjani
35. [Электронный ресурс]: документация https://www.ebyte.com/en/product-view-news.aspx?id=122
36. [Электронный ресурс]: документация https://doc.platan.ru/pdf/datasheets/ebyte/E18-MS1PA2-PCB.pdf
37. [Электронный ресурс]: URL - https://modkam.ru/2019/04/16/koordinator-zigbee-v2/
38. [Электронный ресурс]: документация https://www.manualslib.com/manual/1581346/Ebyte-E18.html

39. https://aliexpress.ru/item/1005001826162194.html?sku_id=12000017825104253
40. https://gio-dot.github.io/Z1-Mini/
41. Wireless Sensor Network Energy Model and Its Use in the Optimization of Routing Protocols 2020, Carolina Del-Valle-Soto, Carlos Mex-Perera, Juan Arturo Nolazco-Flores, Ramiro Velázquez, Alberto Rossa-Sierra
42. [Электронный ресурс]: URL - https://aliexpress.ru/popular/conbee-ii.html
43. [Электронный ресурс]: документация https://www.manualslib.com/manual/2409535/Ebyte-E18-Series.html?page=3#manual
44. [Электронный ресурс]: документация - ZigBee Cluster Library Specification. October, 2007.
45. [Электронный ресурс]: документация Zigbee Application Framework Developer's Guide, 52c
46. [Электронный ресурс]: URL - https://aliexpress.ru/wholesale?catId=&SearchText=e18%20zigbee
47. The Design and Realization of Household Intelligent Security System 2016, Huang Sheng-Bo, Fan Tong-Liang
48. [Электронный ресурс]: документация - ZigBee Cluster Library Specification Revision 6 Draft Version 1.0
49. [Электронный ресурс]: URL - https://aliexpress.ru/wholesale?catId=&SearchText=arduino%20pro%20micro
50. [Электронный ресурс]: документация Z-Stack Sample Applications, 2011 – 24c
51. [Электронный ресурс]: документация CC253x System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee® Applications
52. [Электронный ресурс]: URL - https://www.chipdip.ru/catalog/popular/l7805cv?from=suggest_popular

53. [Электронный ресурс]: URL - https://aliexpress.ru/wholesale?catId=&SearchText=l7805cv
54. [Электронный ресурс]: документация CC2540/41 System-on-Chip Solution for 2.4- GHz Bluetooth® low energy Applications
55. [Электронный ресурс]: документация - CC2530 Datasheet, 2011.
56. [Электронный ресурс]: документация - Manual Ebyte-E18-Series, 2019.
57. [Электронный ресурс]: URL - https://www.virtualbox.org/wiki/Downloads
58. [Электронный ресурс]: URL - https://www.home-assistant.io/installation/windows

Приложение А

Принципиальная электрическая схема модуля E18-MS1-PCB

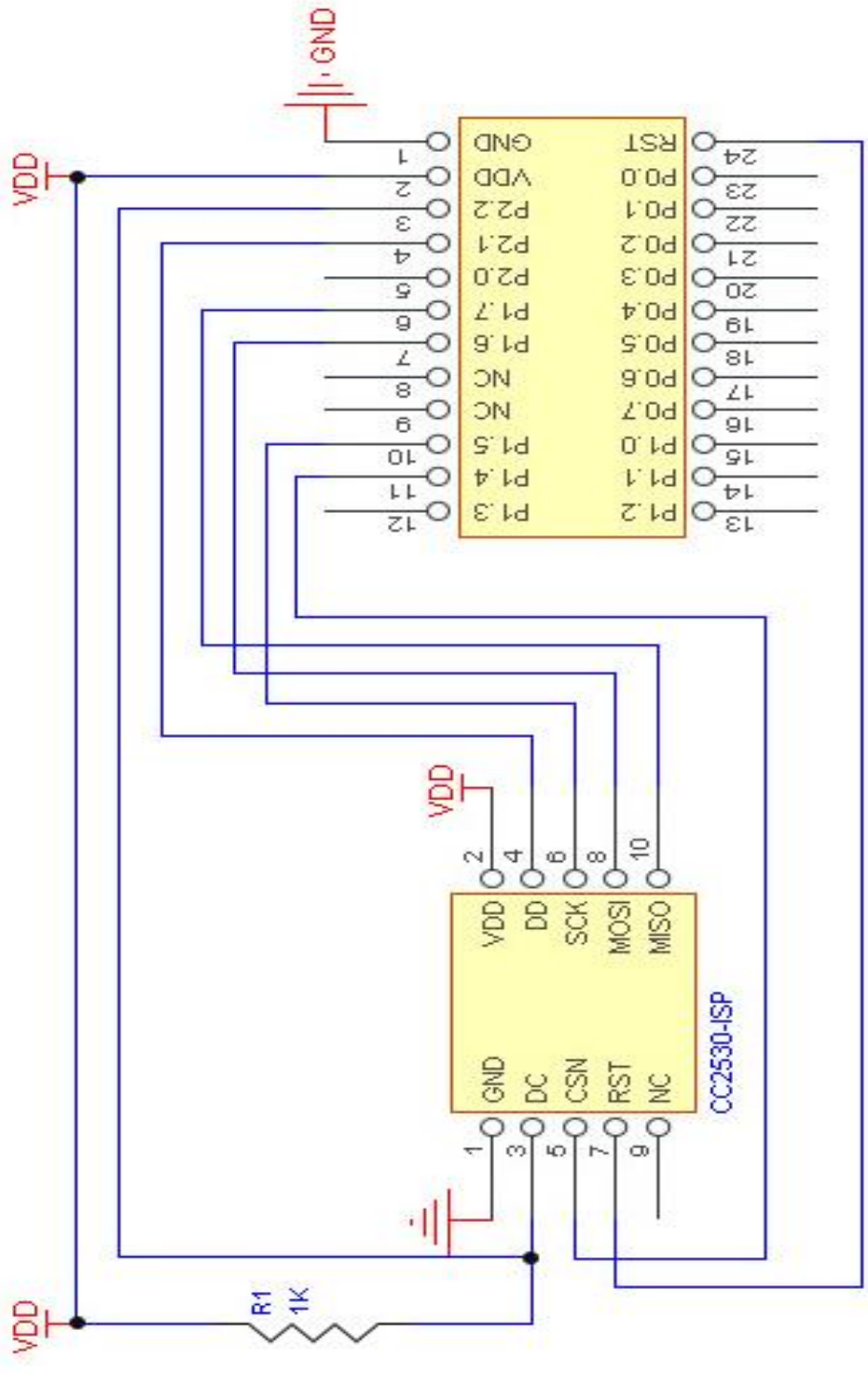

Рисунок А.1 - Принципиальная электрическая схема модуля E18-MS1-PCB

Приложение Б

Принципиальная электрическая схема Arduino Pro Micro

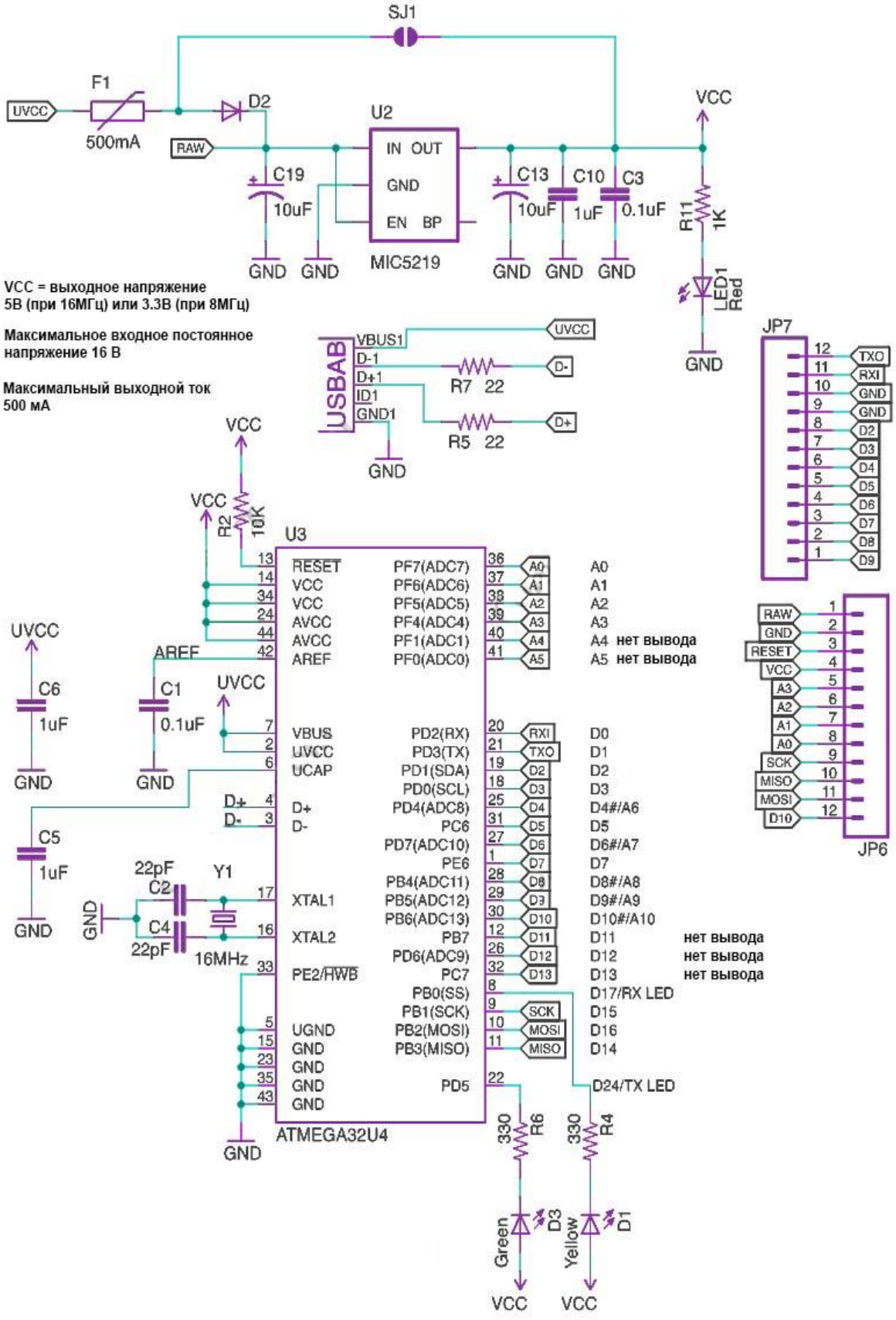

Рисунок Б.2 - Принципиальная электрическая схема Arduino Pro Micro

Приложение В

Схема работы устройства

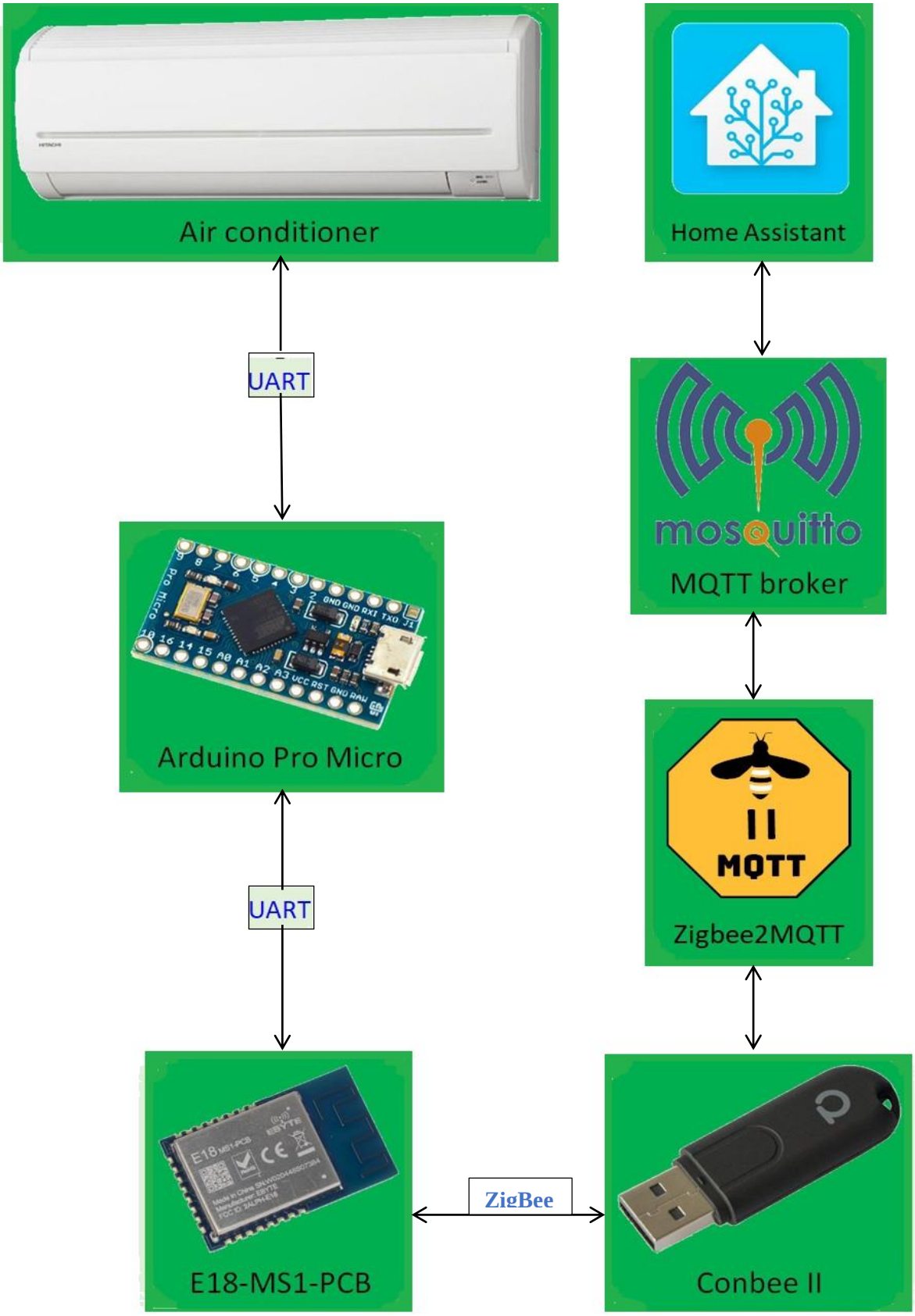

Рисунок В.3 - Схема работы устройства