МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2019 г.

Программно-аппаратный модуль интернета вещей для обнаружения углекислого газа в окружающей среде

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ П.О. Шабуров
«___»_____2019 г.

Автор работы,
студент группы КЭ-222
_____ О.В. Рябцева
«___»_____2019 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2019 г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2019 г.

ЗАДАНИЕ

на выпускную квалификационную работу магистра

студенту группы КЭ-222
Рябцевой Ольги Викторовны
обучающемуся по направлению
09.04.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Программно-аппаратный модуль интернета вещей для обнаружения углекислого газа в окружающей среде» утверждена приказом по университету от 25 апреля 2019 г. № 899.

2. **Срок сдачи студентом законченной работы:** 1 июня 2019 г.

3. **Исходные данные к работе:** статьи, книги, техническое задание.

– ГН 2.2.5.2100-06. Предельно допустимые концентрации (ПДК) вредных веществ в воздухе рабочей зоны (дополнение N 2 к ГН 2.2.5.1313-03 Предельно допустимые концентрации (ПДК) вредных веществ в воздухе рабочей зоны). – 2006.
– Selonen, P. IoT Cloud Environment for Enabling the Programmable World. Proc. 42nd Euromicro Conf. Software Eng. And Advanced Applications (SEAA 16). — 2016. — P. 250–257.

4. **Перечень подлежащих разработке вопросов:**

- рассмотрение научных проектов, доказывающих негативное влияние повышенного содержания углекислого газа в окружающей среде на человека;
- анализ существующих решений по выявлению содержания углекислого газа в окружающей среде;
- разработка собственного программно-аппаратного комплекса интернета вещей для контроля уровня углекислого газа;
- оценка работоспособности разработанного программно-аппаратного комплекса интернета вещей в различных режимах и внешних условиях.

5. **Дата выдачи задания:** 1 декабря 2018 г.

Руководитель работы _____/ П.О. Шабуров /

Студент _____/ О.В. Рябцева /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2019	
Разработка модели, проектирование	01.04.2019	
Реализация системы	01.05.2019	
Тестирование, отладка, эксперименты	15.05.2019	
Компоновка текста работы и сдача на нормоконтроль	24.05.2019	
Подготовка презентации и доклада	30.05.2019	

Руководитель работы _____/ П.О. Шабуров /

Студент _____/ О.В. Рябцева /

Аннотация

О.В. Рябцева. Программно-аппаратный модуль интернета вещей для обнаружения углекислого газа в окружающей среде. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 132 с., 27 ил., библиогр. список – 33 наим.

В ходе работы был разработан программно-аппаратный комплекс, состоящий из физического модуля и программной реализации для управления на микропроцессорных средствах. Комплекс основан на микроконтроллере ESP32, отвечающий современным требованиям эффективности и надежности.

Назначение физического модуля состоит в выявлении содержания концентрации углекислого газа в окружающей среде и передачи данного показателя другим модулям, входящих в состав Интернета вещей для принятия определенных действий.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Атмосферный воздух – смесь различных газов. В его составе 78,08% азота, 20,9% кислорода, 0,93% аргона, 0,03% углекислого газа. [1] Кроме того небольших количествах содержится водород, гелий и другие газы. Процентная доля воды и углекислого газа (CO_2) может изменяться в зависимости от окружающих условий, в то время как концентрация большинства газов в атмосфере практически остается постоянной.

В помещении основным источником углекислого газа является человек. В разнообразных местах, где находятся люди существует вероятность превышения нормы углекислого газа из-за дыхания людей.

Тема выпускной квалификационной работы выбрана неслучайно, этому способствовали следующие факторы. Во-первых, при высоких концентрациях CO_2 в воздухе, проявляется значительное понижение внимания и возникает хроническая усталость.

Во-вторых, углекислый газ становится причиной, повышенной заболеваемости людей. Первым делом проявляется недомогание дыхательных путей и носоглотки, увеличивается число астматических приступов. В случае длительного воздействия углекислого газа на организм человека, в крови начинают происходить биохимические изменения, что приводит к гипертонии, ослаблению сердечнососудистой системы и т. д.

В-третьих, повышенное содержание концентрации углекислого газа в квартире вызывает головную боль и бессонницу. Поэтому контролировать углекислый газ нужно не только в школах, детских садах и офисах, но и в квартирах.

В-четвертых, концепция Интернета вещей при помощи объединения в единую сеть сенсоров, устройств и людей, предоставляет возможность беспрепятственного взаимодействия человека и машины, программного обеспечения и оборудования. При наличии подключения Интернета,

устройство может быть настроено для автоматического выполнения определенных действий. Таким образом, изделие самостоятельно адаптируется к поставленным задачам с помощью облачных ресурсов.

Для того, чтобы концентрация углекислого газа в воздухе не превышала норму, помещения должны быть оснащены вентиляционными системами и регулярно проветриваться. Наличие пластиковых стеклопакетов в современных зданиях осложняет процесс естественной вентиляции. Во время перемены за счет «сквозного проветривания», которое учителя вынуждены делать, понижается концентрация углекислого газа, но уровень быстро возрастает. Следовательно, данный метод не эффективен. И данный показатель влияет на состояние организма детей.

Изучив данные проблемы, было решено разработать программно-аппаратный модуль по оценке углекислого газа в окружающей среде, который позволит измерять температуру и контролировать содержание CO_2 в воздухе посредством взаимодействия с другими устройствами интернета вещей.

1. АНАЛИЗ РЕШАЕМОЙ ЗАДАЧИ

1.1. АКТУАЛЬНОСТЬ ЗАДАЧИ

За последние 50 лет содержание углекислого газа в атмосферном воздухе увеличилось на 20% и данный показатель продолжает расти. В первую очередь в крупных городах из-за выхлопов автомобилей и промышленных выбросов [2].

Средний объем жизненной ёмкости лёгких человека составляет 4−4,5 л. [2]. При нахождении одного человека в комнате площадью 10 м² с высотой потолков 2,70 м, одним окном с закрытой форточкой и при температуре воздуха 20 °C, концентрация O_2 за час снижается приблизительно на 1%. Ощущение гипоксии и гиперкапнии появится через 6−8 часов, если он будет спать, и через 4−5 часов при активной деятельности. Таким образом работоспособность начинает снижаться через 2−3 часа, при этом еще нет симптомов нехватки O_2, но повышение CO_2 стимулирует дышать чаще. Поэтому потребление кислорода в замкнутом помещении имеет не линейную, а параболическую характеристику, которая тем круче уходит вверх, чем больше человек работает [2]. Следовательно, проветривание помещения является постоянной необходимостью.

Для обоснования актуальности проблемы, были проведены замеры уровня углекислого газа в школьном классе МАОУ «Лицей № 102» г. Челябинска. На рисунке 1 показан график уровня концентрации углекислого газа в классе во время урока. Максимум содержания CO_2 приходится на конец учебного занятия и составляет более 1500 ppm, из-за его накопительного эффекта. Данный уровень является очень высоким показателем. За время перемены 15 минут концентрация CO_2 снижается, но затем возрастает.

Даже после проветривания помещения, концентрация CO_2, начинает быстро расти, вследствие дыхания людей, находящихся в здании.

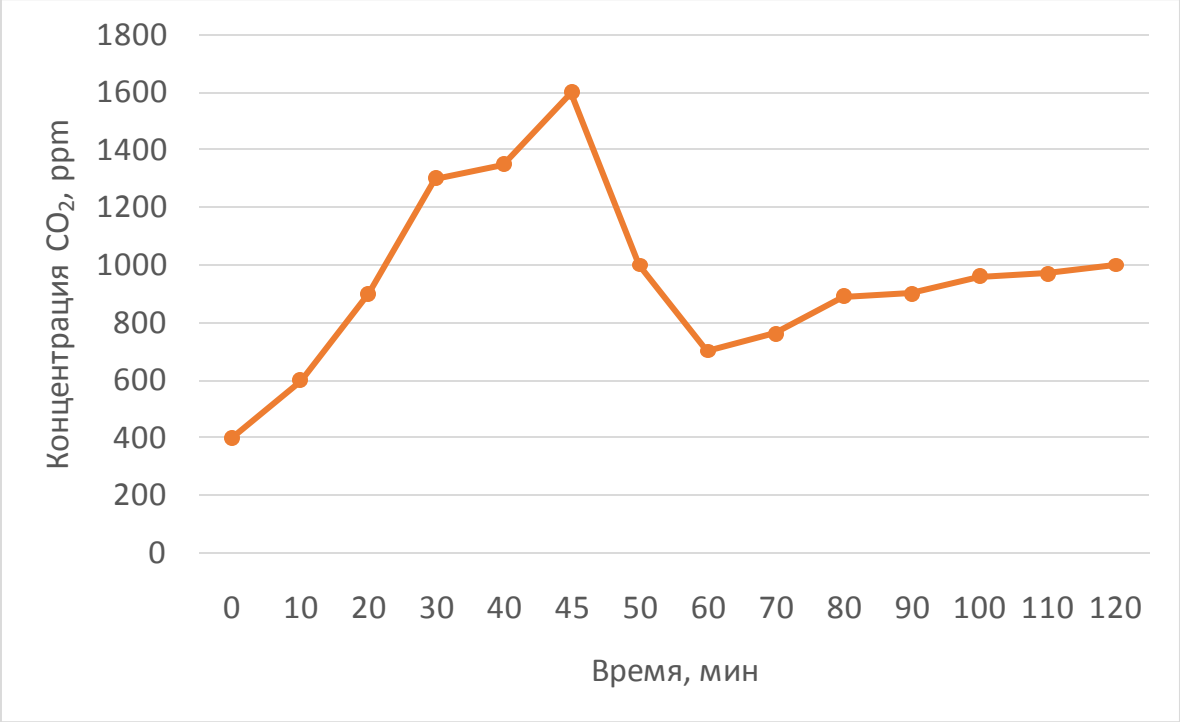

Рисунок 1 – Уровень углекислого газа в школьном классе

Данный показатель возрастает из–за отсутствия или неисправной принудительной и естественной вентиляции, также пластиковые окна не пропускают воздух. Как показывают исследования для того, чтобы сохранялся здоровый микроклимат в помещении необходимо держать окна открытыми в течение всего урока. Но это осложняется наличием шума за окном, холодных сквозняков, неприятных запахов, дыма и других факторов

С 2011 года в Европе и Центральной Азии реализуется исследовательская программа «Внутришкольная среда и заболеваемость органов дыхания у детей» («School Environment and Respiratory Health of Children» (SEARCH) [3]. Благодаря этой программе был разработан план урока, который демонстрирует важность свежего воздуха и вентиляции.

В России с 1 октября 2008 года действует ГОСТ РЕН 13779-2007 «Вентиляция в нежилых зданиях. Технические требования к вентиляции и кондиционированию» [4], приводящий уровни загрязнения наружного воздуха. Данные показатели приведены в таблице 1.

Таблица 1 – Примеры содержания загрязнений в наружном воздухе

Окрестность	**Концентрация частиц**					
	, ppm	, мг/	, мкг/	, мкг/	Общая концентрация, мг/	, мкг/
Сельская местность, существенные источники отсутствуют	350	1	5-35	5	0,1	20
Небольшой город	375	1-3	15-40	5-15	0,1-0,3	10-30
Загрязненный центр большого города	400	2-6	30-80	10-50	0,2-1,0	20-50

Данные значения являются среднегодовыми.

Были проведены исследования в рамках ЕАС. Мотивацией для данного действия послужила гипотеза, высказанная в 2004 году: углекислый газ является причиной, повышенной заболеваемости людей. В результате на ежегодной конференции Европейского Респираторного Общества (European Respiratory Society – ERS) [5], проходившая 2-6 сентября 2006 года в Мюнхене, было подтверждено данное предположение.

Корейские ученые провели исследование «Влияние концентрации CO2 в помещении на приступы астмы у детей». В нем приняли участие 181 ребенок моложе 14 лет [6]. В ходе исследования были проведены замеры уровня содержания веществ, которые считаются основными загрязнителями воздуха в помещении , и аллергены. В результате, выяснилось, что только уровень концентрации в жилом помещении влияет на увеличение частоты приступов астмы у детей.

В свою очередь американские ученые провели тесты. Результаты подтвердили предположения, уровень углекислого газа в классе влияет на состояние школьников, при росте данного показателя, снижается внимание и это отражается на качестве обучения.

В школах Финляндии регулярно проводится контроль за уровнем углекислого газа в школьных классах. Те школы, где уровень CO2

превышает допустимые нормы, вынуждены решать проблему качества воздуха, поскольку могут оказаться под угрозой закрытия.

Нормальная кислотность крови человека составляет 7,35 – 7,45 по шкале pH. На живое существо даже небольшие постоянные изменения кислотности крови оказывают негативное влияние. При повышении концентрации в воздухе, увеличивается парциальное давление в альвеолах, его растворимость в крови повышается, и образуется слабая угольная кислота, которая рассчитывается по формуле (1)

, (1)

распадающаяся на H+ и -. Происходит ацидоз, то есть смещение кислотно-щелочного баланса организма в сторону увеличения кислотности (уменьшению pH). Следовательно, при повышении концентрации в воздухе, снижается pH крови и кислотная реакция (рис. 2).

В последствии ацидоза человек становится перевозбужденный, у него учащается сердцебиение и повышается давление. Также может проявляться беспокойство.

Карл Шафер является сотрудником медицинской научно-исследовательской лаборатории военно-морского подводного флота США. Он проводил исследования влияния концентрации углекислого газа на морских свинок. В результате воздействия в течение восьми недель на грызунов концентрации равной 5000 ppm, у них была выявлена кальцификация почек. После чего уровень углекислого газа был снижен и составлял 3000 ppm, но данная особенность все равно наблюдалась. Были замечены структурные изменения в легких. Таким образом, был сделан вывод, что организм грызунов проходил адаптацию к постоянному воздействию углекислого газа. Данные признаки исчезали при уменьшении воздействия.

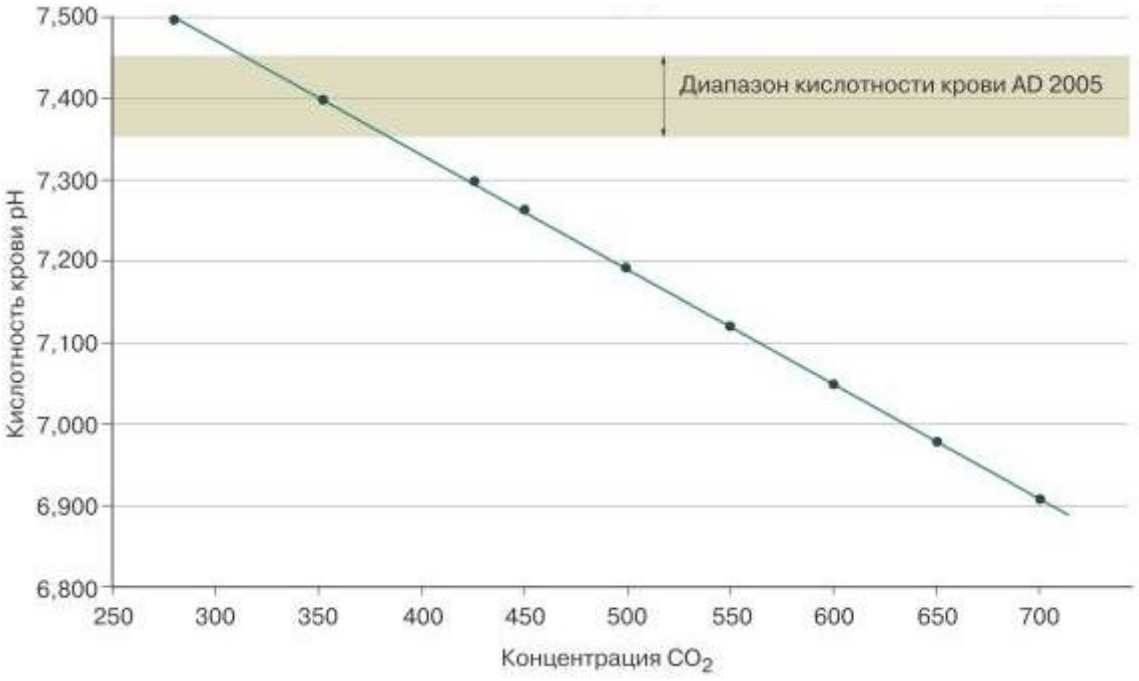

Рисунок 2 – Изменения pH крови с увеличением концентрации углекислого газа

В городе Тайбэй (Тайвань) было проведен эксперимент. Реакция организма офисных работников была обследована на влияние углекислого газа. В результате повышения концентрации выше 800 ppm был замечен рост маркеров окислительного стресса. Зависимость такова, что содержание маркеров увеличивается с возрастанием духоты помещения.

Также проводились исследования в Европе. Будапештский университет технологии и экономики принялись разрабатывать специальную технологию, которая призвана уменьшить уровень загрязнения другими веществами. Углекислый газ является главным загрязнителем. Были проведены исследования, в которых концентрация поднималась до 3000 ppm. В результате молодые и здоровые люди, чувствовали ухудшение самочувствия. Таким образом можно сделать вывод, что офисные работники ощущают еще большее недомогание.

В России ученные не занимались исследованиями на данную тему. Но были проведены замеры в офисах города Москва. В результате уровень поднимается до отметки 2000 ppm. Однако О.В. Елисеева в 1960 году

написала диссертацию, в которой были отражены исследования по обоснованию ПДК CO_2 в окружающей среде. Подобно европейским ученым, она наблюдала за влиянием углекислого газа в разных концентрациях на организм человека. Выводы оказались таковыми, что повышенное содержание вызывает отчетливые сдвиги в функции внешнего дыхания, кровообращении и электрической активности головного мозга. Основываясь на своих выводах, она разработала нормы, если верить которым, содержание CO_2 в воздухе жилых и общественных зданий не должно превышать 1000 ppm, а среднее содержание CO_2 должно быть около 500 ppm. Но несмотря на данные исследования, не были приняты никакие нормативные акты. Нет подобных норм для учебных, офисных и жилых помещений в СНиПах (строительных нормах) и СанПинах (санитарные правила и нормы).

Для обоснования актуальности проблемы, были проведены замеры уровня углекислого газа в среднестатистическом офисе в центре г. Челябинска. Площадь помещения составляет 15 м², численность сотрудников 4 человека. В течение восьмичасового рабочего дня были проведены замеры, результаты продемонстрированы на рисунке 3. Из графика видно, что на момент начала рабочего дня уровень углекислого газа представляет максимальную отметку. В результате проветривания помещения и бесперебойной работы кондиционера концентрация понижается и в среднем держится на отметке 700 ppm.

Также были проведены измерения содержания углекислого газа в квартире спального района г. Челябинска, площадь помещения 68 м², проживающих 2 человека. Распорядок был таков:

1. С 9 до 10 часов производилось проветривание. Открыты окна во всех комнатах, концентрация CO_2 спадает с 2000ppm до 600ppm.
2. С 10 до 15 часов окна в комнатах закрыты, на кухне открыта форточка. В квартире 1 человек. Концентрация CO_2 в норме.

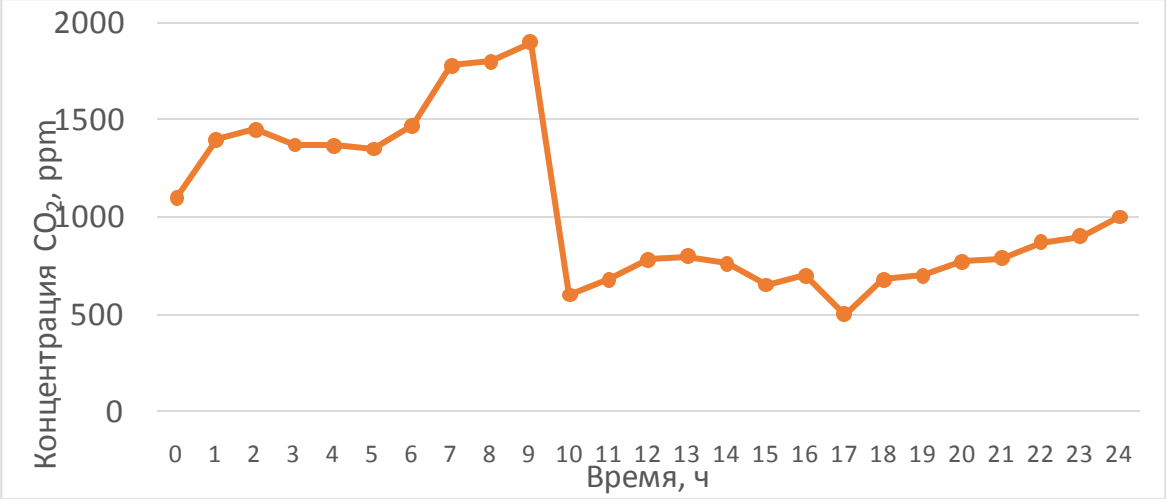

Рисунок 3 – Уровень углекислого газа в помещении офиса

3. С 15 до 18 часов открыты форточки во всех комнатах. В квартире 2 человека. Концентрация CO_2 всё еще в норме.
4. С 18 до 21 часа открыты форточки во всех комнатах. В квартире 3 человека. Концентрация CO_2 начинает нарастать.
5. С 21 до 22-30 часов проветривание с открытыми окнами. В квартире 3 человека. Концентрация CO_2 приходит в норму, но начинает повышаться при закрытии окон.

Результаты отражены на рисунке 4. Основываясь на проведенные эксперименты, можно сделать следующие выводы о том, что постоянное воздействие повышенного содержания уровня углекислого газа на организм человека ведет к гиперкапнии.

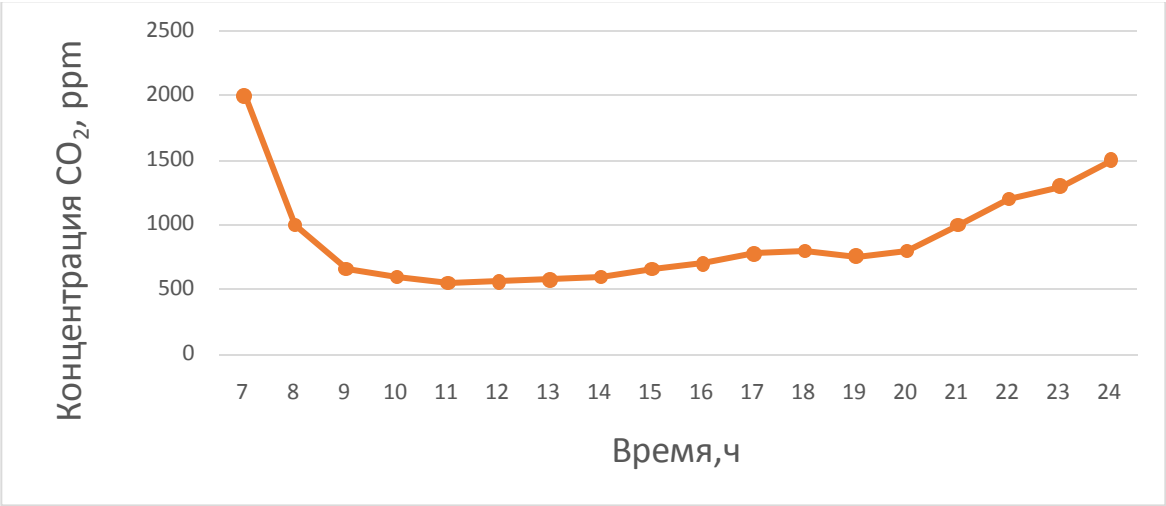

Рисунок 4 – Уровень углекислого газа в квартире

Исследования Джесси Браун, сотрудницы медицинского факультета «Школа медицины им. Файнберга» Северо-западного университета Чикаго, подтверждают, что гиперкапния, вызванная повышением уровня CO2 в крови и тканях, обычно развивается у пациентов с запущенными заболеваниями легких и тяжелыми легочными инфекциями и связана с высокой смертностью [7].

Также было выяснено, для поддерживания уровень содержания углекислого газа в воздухе, необходимо постоянное проветривание помещения. Данное условие осложняется рядом факторов, но поскольку в последнее время большую популярность приобрели устройства Интернет вещей, призванные автоматизировать управление устройствами и тем самым облегчить задачу человечеству, то данные ресурсы необходимо использовать при решении изученной проблемы.

1.2. ЦЕЛЬ ДИПЛОМНОЙ РАБОТЫ

Целью дипломной работы является разработка программно-аппаратного модуля, являющегося частью системы умного дома с возможностью автономной работы, который обнаруживает содержание в воздухе углекислого газа и измеряет температуру и уровень влажности. Позволяющий при помощи взаимодействия с другими устройствами Интернета вещей полностью контролировать содержание CO2, управлять и производить мониторинг окружающей среды, поддерживать оптимальный микроклимат, отдавая команды управления кондиционерам, приточной вентиляции и очистителям воздуха.

Для достижения поставленной цели необходимо сделать программно-аппаратное устройство, аппаратная часть которого будет состоять из модуля, выдающего управляющие сигналы, необходимые для выполнения функциональных требований комплекса, иметь встроенный АКБ, а также содержать все необходимые разъемы для взаимодействия пользователя.

С помощью программной части комплекса будет существовать возможность интеграции с системой умного дома. Блок обмена информацией будет позволять использовать такой интерфейс взаимодействия с пользователем, как USB.

1.3. ВЫЯВЛЕНИЕ НЕОБХОДИМОГО ФУНКЦИОНАЛА

Содержание углекислого газа измеряется в миллионных долях (ppm - parts per million), равная 1 от базового показателя. В России проектировщики многоквартирных и частных домов берут за основу ГОСТ 30494-2011 под названием «Здания жилые и общественные. Параметры микроклимата в помещениях» [8]. Этот документ оптимальным для здоровья человека уровнем CO_2 считает 800 - 1000 ppm. Предел допустимого содержания углекислого газа в помещении - 1400 ppm. Однако уже 1000 ppm не признается вариантом нормы целым рядом исследований, посвященных зависимости состояния организма от уровня CO_2 [6]. Их данные свидетельствует о том, что на отметке 1000 ppm больше половины испытуемых ощущают последствия ухудшения микроклимата: учащение пульса, головную боль, усталость. В таблице 2 показана разница между строительными нормативами и санитарно-гигиеническими рекомендациями.

Таким образом, необходимо чтобы разрабатываемый комплекс имел высокочувствительный датчик, с помощью которого определял значение углекислого газа в окружающей среде в миллионных долях. Имел дисплей для отображения показателей уровня CO_2 и температуры окружающей среды.

Поскольку существует проблема постоянного контроля и проветривания помещения, которая описана в 1.1, то необходимо внедрить данное устройство в систему умного дома. Умный дом – это единая система управления в доме, офисе, квартире или здании, включающая в себя датчики, управляющие элементы и исполнительные устройства. [8]

Таблица 2 – Физиологические проявления при различных уровнях углекислого газа в воздухе

Концентрация CO_2 (ppm)	**Строительные нормативы (согласно ГОСТ 30494-2011)**	**Влияние на организм (согласно санитарно-гигиеническим исследованиям)**
менее 800	Воздух высокого качества	Идеальное самочувствие и бодрость
800 – 1 000	Воздух среднего качества	На уровне 1 000 ppm каждый второй ощущает духоту, вялость, снижение концентрации, головную боль
1 000 - 1 400	Нижняя граница допустимой нормы	Вялость, проблемы с внимательностью и обработкой информации, тяжелое дыхание, проблемы с носоглоткой
Выше 1 400	Воздух низкого качества	Сильная усталость, безынициативность, неспособность сосредоточиться, сухость слизистых, проблемы со сном

1.4. ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ

Для достижения цели выпускной работы необходимы выполнить несколько этапов:

1. Анализ существующих решений.
2. Выявление достоинств и недостатков существующих устройств.
3. Разработка программно-аппаратной части комплекса:

– разработка структуры устройства;
– выбор компонентов схемы;
– синтез принципиальной схемы;
– разводка печатной платы устройства;
– изготовление опытного образца;
– интеграция с системой умного дома;
– тестирование работы комплекса в реальных условиях.

1.5. ОБЗОР АНАЛОГОВ

На рынке, существует ряд родственных устройств, имеющих схожий функционал с разрабатываемым комплексом, но не являющихся модулями интернета вещей. В таблице 3 представлены основные параметры существующих датчиков измерения углекислого газа в окружающей среде.

Таблица 3 – Сравнительные характеристики детекторов углекислого газа

	AirVisual Pro	Мастеркит MT8057S	Awair	HT-2000	NetAmo Smart Home	uHoo	ТИОН	Xiaomi Mijia Air Detector
Датчик CO2	+	+	+	+	+	+	+	+
Датчик PM2.5	+	-	+	-	-	+	-	+
Экран	LCD 5″	LCD + LED	LED	LCD	LED	LED	LED	3,97″
Измерение температуры	+	+	+	+	+	+	+	+
Измерение влажности	+	-	+	+	+	+	+	+
Питание	USB 5V	USB 5V	DC12V	DC6V	USB 5V	USB 5V	USB 5V	USB 5V
Встроенный аккумулятор	1900 мАч	-	-	4xAA	-	-	-	2000 мАч
WIFI	+	-	+	-	+	+	+	+
Приложение	+	-	+	-	+	+	+	+
Язык	английский	русский	английский	английский	английский	английский	русский	английский, китайский
Интеграция с системой умного дома	-	-	-	-	-	-	частично	MiHome
Стоимость, руб	21990	5950	13000	7000	11000	20790	15000	5000

Из таблицы можно заметить, что рынок может предложить разнообразие устройств с датчиком CO_2. Этот факт еще раз доказывает обеспокоенность населения и желание контролировать данный показатель, что объясняет актуальность разработки.

Ряд устройств выполняют базовые функции, но не являются устройствами системы умный дом. Таким образом, нельзя назвать данные устройства прямыми аналогами. Но 11 ноября 2018 года китайский производитель смартфонов и аксессуаров презентовал Xiaomi MIJIA Air Quality Detector, который помимо выполнения основных функций имеет возможность подключения к системе Mi Home. Но несмотря на расширенную линейку устройств, поддерживающих систему Mi Home, в России возникают

проблемы с их интеграцией. Устройство еще не представлено на российском рынке. Отечественный производитель микроклиматических устройств ТИОН предлагает свой продукт – MAGICAIR. Он измеряет качество воздуха, отображает данные на смартфоне и может управлять климатической техникой, но только из линейки устройств, которые предложит сам производитель, она достаточно ограничена.

Таким образом, изучив предлагаемые альтернативные варианты, было решено разработать программно-аппаратный комплекс интернета вещей по оценке углекислого газа в окружающей среде, который должен выполнять ряд требований:

- измерять содержание CO_2, температуру и влажность окружающей среды;
- иметь возможность автономной работы;
- поддерживать русский язык;
- иметь простой интерфейс пользователя;
- отвечать современным требованиям микроклимата окружающей среды;
- быть интегрирован в систему умного дома.

2. СТРУКТУРА ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА

2.1. ПОСТАНОВКА ЗАДАЧИ

Задача состоит в разработке программно-аппаратного модуля, реализованного на микропроцессорных средствах, позволяющего производить измерение содержания CO_2 в окружающей среде и управлять устройствами, интегрированными в систему умный дом. Программно-аппаратный комплекс позволит расширить функциональные возможности существующих устройств, поддерживать здоровый микроклимат окружающей среды, экономить электроэнергию и ресурс приборов.

2.2. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Поскольку в результате обзора аналогов в пункте 1.5 были выявлены недостатки существующих устройств, то исходя из этого были учтены рекомендуемые параметры для разрабатываемого комплекса. Должны обеспечиваться:

- измерение параметра CO_2 в диапазоне 0 – 4000 ppm с погрешностью ± (50ppm+5%);
- измерение температуры от -40 до +60°C с погрешностью (+- 0.5%);
- измерение влажности от от 0 до 100% с погрешностью (+- 5%);
- цветной экран;
- возможность автономной работы;
- управление по Wi-Fi;
- поддержка облачной платформы для интеграции с системой умного дома;
- понятный интерфейс пользователя на русском языке;
- доступность покупки.

Конструктивно устройство должно иметь сравнительно небольшие габариты, обеспечиваться разъёмами для присоединения к внешним устройствам.

2.3. ПУТИ РЕШЕНИЯ ЗАДАЧИ

Исходя из п. 2.1. и п. 2.2. для выполнения поставленной задачи был реализован данный функционал:

- измерение параметра CO2 в диапазоне 0 – 5000 ppm с погрешностью ± (50ppm+5%);
- измерение температуры от -40 до +80°C с погрешностью (+- 0.5%);
- измерение влажности от от 0 до 100% с погрешностью (+- 5%);
- цветной LCD экран 320 на 240 точек;
- аккумулятор емкостью 700 мАч;
- поддержка WI-FI, Bluetooth;
- встроенный динамик;
- поддержка облачной платформы для интеграции с системой умного дома.

2.4. ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ ФУНКЦИОНАЛА

Выявленный в пункте 2.3. функционал может быть реализован с помощью микроконтроллера с интегрированным WI-FI и Bluetooth контроллерами и антеннами, датчика CO_2, температуры и влажности, модулей обмена информацией. Данное решение подходит для бытового, офисного использования, в детских учреждениях.

2.5. ОБЩАЯ СТРУКТУРА АППАРАТНОЙ ЧАСТИ КОМПЛЕКСА

Аппаратное обеспечение состоит из различных электронных элементов. Для их выбора необходимо выделить основные функциональные

модули программно-аппаратного комплекса. В его состав входят следующие модули:

- система управления;
- питание информационной части;
- датчик температуры, углекислого газа, влажности;
- обмен информацией с системой.

Каждый из которых обладает определенным функционалом, благодаря которому программно-аппаратный комплекс имеет возможность действовать как единый механизм. На рисунке 5 представлена структурная схема, которая позволяет наглядно отразить устройство рабочих изменений в комплексе, а также демонстрирующая процессы, протекающие в комплексе в целом.

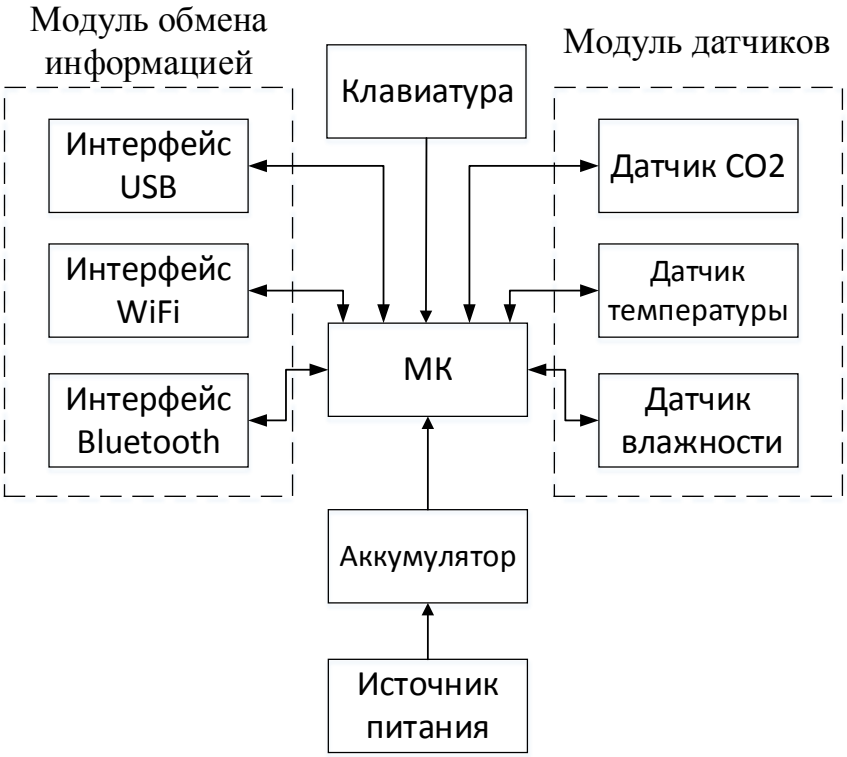

Рисунок 5 – Структурная схема

3. ПОДБОР КОМПОНЕНТОВ И РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ СХЕМЫ АППАРАТНОЙ ЧАСТИ КОМПЛЕКСА

3.1. ОБЩИЕ ПОЛОЖЕНИЯ

Следующим этапом разработки аппаратной части модуля является преобразование структурных схем модулей в схемы электрические принципиальные. В этом разделе рассмотрены основные моменты выбора элементов модуля и некоторые особенности работы аппаратной части комплекса. Основным критерием разработки принципиальных схем является минимум стоимости конечного продукта, не в ущерб надежности работы. Эти критерии накладывают отпечаток на выбор компонентов схемы.

Схема электрическая принципиальная данного комплекса представлена в приложении А.

3.2. ВЫБОР ДАТЧИКОВ

Программно-аппаратный комплекс содержит датчики, которые оценивают значения неэлектрических параметров и преобразуют их в электрические сигналы. Преобразованные сигналы передаются в блок управления, который в соответствии с заложенной программой приводит в действие исполнительные механизмы. Поскольку в соответствии техническими требованиями к устройству, описанные в пункте 2.2, необходимо:

- измерять параметр CO_2 в диапазоне 0 – 4000 ppm с погрешностью ± (50ppm+5%);
- измерять температуру от -40 до +60°C с погрешностью (+- 0.5%);
- измерять влажность от от 0 до 100% с погрешностью (+- 5%).

Был проведён обзор датчиков:

– углекислого газа;
– температуры;
– влажности.

3.2.1. ВЫБОР ДАТЧИКА УГЛЕКИСЛОГО ГАЗА

Принципиальным требованием для контроля углекислого газа в окружающей среде является наличие датчика. Существуют разнообразные варианты реализации данного требования.

Компания Winsen предлагает широкий спектр разнообразных газовых датчиков, предназначенных для контроля качества воздуха в помещениях. Для контроля CO2 компания Winsen представляет ассортимент продукции, которое позволяет проводить необходимые измерения. В зависимости от конкретной задачи можно выбрать как простые датчики, формирующие электрический сигнал, пропорциональный анализируемому параметру, так и полноценные готовые модули, осуществляющие комплексный контроль качества воздуха с передачей информации по распространенным интерфейсам, UART или I2C, сопровождающиеся световой и звуковой сигнализацией.

Модули для измерения концентрации углекислого газа, используют инфракрасный метод и представляют собой малогабаритные измерители с высокой селективностью, которая не зависит от содержания кислорода в воздухе. Благодаря эффективной технологии обнаружения углекислого газа, прецизионной оптической системы и оригинальной схемотехники, модули обладают высоким качеством. Результаты измерений могут быть получены в аналоговом, дискретном (ШИМ) или цифровом виде по интерфейсу UART или Modbus. На рисунке 6 представлены сенсоры измерения концентрации углекислого газа данной компании.

Рисунок 6 – Датчики CO_2

В основе работы инфракрасных газоанализаторов лежит явление поглощения фотонов молекулами вещества, которое используется в спектроскопии. Поскольку большинство газов способно поглощать световое излучение определенных длин волн, то их комбинация позволяет судить о химическом составе газа.

Количество поглощаемого излучения пропорционально количеству молекул, следовательно, концентрация определяется измерением интенсивности света после прохождения через измерительную камеру. С помощью инфракрасной спектроскопии можно регистрировать наличие большого количества веществ, однако наибольшую эффективность этот метод имеет при определении концентрации CO_2.

Углекислый газ поглощает свет с длинной волны 4,26 мкм, так как эта длина волны уникальна для CO_2, то измерения ее интенсивности вполне достаточно для определения его количества в воздухе. Это значительно упрощает устройство регистратора, позволяя использовать простой недисперсионный инфракрасный (Nondispersive Infrared, NDIR) метод измерения. В качестве источника света в газоанализаторе используется миниатюрная лампа накаливания или светодиод, представленная на рисунке 7. Луч света, проходя через измерительную камеру, освещает фотоэлемент, перед которым устанавливается светофильтр, пропускающий свет только одной длины волны, для CO_2 - это 4,26 мкм.

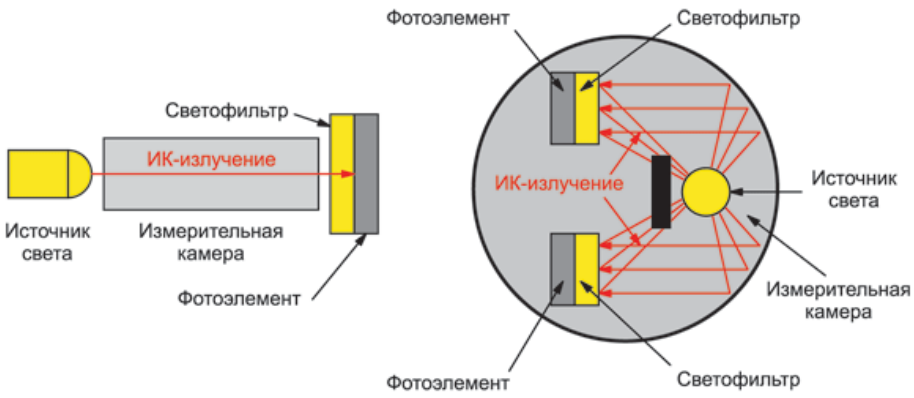

Рисунок 7 – Устройство недисперсионного инфракрасного газоанализатора

Зависимость такова, что чем выше концентрация CO_2, тем меньше освещенность фотоэлемента. Для обеспечения высокой чувствительности длина светового пути должна быть как можно больше, поэтому при создании компактных газоанализаторов часто используют многократное отражение луча от внутренних стенок измерительной камеры.

Датчики из серии MH-Z1x предназначены для использования в системах отопления, вентиляции и кондиционирования, в холодильном оборудовании, приборах для контроля качества и очистки воздуха, интеллектуальных системах, например, умный дом. В таблице 4 представлены сравнительные характеристики датчиков CO_2.

Датчики MH-Z19 небольшие габариты и вес, это позволяет создавать на их основе компактные устройства, что актуально для портативных приложений. Датчики MH-Z14A имеет большую точность измерений, в то время как модуль MH-Z14 имеет более широкий рабочий диапазон. Особенностью модуля MH-Z16 является наличие измерительной камеры большого объема, благодаря которой уменьшается время отклика, что актуально для учебного оборудования.

Функционал обоих датчиков идентичен, но у MH-Z19 существенно снижен уровень потребления и уменьшен размер корпуса. Точности датчика вполне достаточно для выполнения требований.

Таблица 4 – Сравнительные характеристики датчиков углекислого газа

Наименование	MH-Z16 NDIR CO2 SENSOR	MH-Z14 NDIR CO2 SENSOR	MH-Z14A NDIR CO2 SENSOR	MH-Z19B NDIR CO2 SENSOR
Диапазон измерений, ppm	0…5000		0…10000	0…5000
Точность, ppm	±(100+6%)		±(50+3%)	±(50+5%)
Напряжение питания, В	4,5…5,5			
Средний потребляемый ток, мА	< 85		< 60	< 20
Выходной сигнал	ШИМ, UART	Аналоговый, ШИМ, UART		
Время готовности, мин	3			
Время отклика, с	< 30	< 90	< 120	< 120
Рабочая температура, °C	0…50			
Рабочая влажность, %	0…95			
Размеры, мм	97×20×17	57,5×34,7×17	57,5×34,7×17	32,5x20,5x9
Срок службы, лет	> 5			

В результате, были рассмотрены два варианта MH-Z14 и MH-Z19. В таблице 5 представлены их сравнительные характеристики.

Таблица 5 – Сравнительные характеристики MH-Z14 и MH-Z19

Наименование	Диапазон измерений, ppm	Точность, ppm	Выходы	Питание	Размер, мм
MH-Z14	0…5000 (0…2000)	±50+5%	Аналоговый, UART, PWM	5 В, < 85 мА (в среднем)	57,2×34,7
MH-Z19			UART, PWM	3,6…5,5 В DC, < 18 мА	32,6×19,5

Все сенсоры поставляются калиброванными и имеют внутреннюю систему термостабилизации, что вместе с различными вариантами выходов позволяет быстро интегрировать их в разрабатываемое устройство. При интеграции следует учесть следующие особенности:

– вход и выход последовательного интерфейса работают с уровнями 3,3 В, и превышение этого напряжения может привести к выходу датчика из строя;
– после включения датчику требуется около трех минут на прогрев, во время которого выходные данные могут является недостоверными;
– хотя в передаваемом датчиком пакете данных и есть данные о температуре, их не стоит использовать из-за невысокой достоверности;

– датчик имеет встроенную автоматическую калибровку. Возможна также и ручная калибровка, но она требует специального оборудования.

Таким образом, в качестве датчика углекислого газа был выбран MH-Z19, имеющий следующие характеристики:

– рабочее напряжение: 3.6 ~ 5.5В;
– потребление тока: < 18мА;
– уровень TTL: 3.3В;
– диапазон измерения: 0 - 5000 ppm (0 ~ 0.5%);
– точность измерений: ± (50ppm+5%);
– сигнал на выходе: UART (TX, RX), PWM;
– время разогрева: до 3 мин;
– рабочая температура: 0 ~ 50 °C;
– рабочая влажность окружающей среды: 0 ~ 95% RH;
– габариты: 33мм*20мм*9мм (ДхШхВ) ;
– вес: 21 г.

3.2.2. ВЫБОР ДАТЧИКА ТЕМПЕРАТУРЫ И ВЛАЖНОСТИ

В результате обзора были выбраны два датчика со схожими характеристиками – DHT11 и DHT22. Датчики состоят из двух частей – емкостного датчика температуры и гигрометра. Первый используется для измерения температуры, второй – для влажности воздуха. Находящийся внутри чип может выполнять аналого-цифровые преобразования и выдавать цифровой сигнал, который считывается посредством микроконтроллера.

В итоге был выбран совмещенный цифровой датчик DHT22 (AM2302). Внешний вид датчика изображен на рисунке 8. Преимуществом датчика является то, что ему необходим только один провод для подключения. Его недостатки: при высокой влажности (более 80%) датчик быстро становится неработоспособным, он требователен к качеству питания и имеет достаточно большую фактическую погрешность при измерении влажности.

Рисунок 8 – Внешний вид датчика DHT22

Для измерения температуры и влажности был проведен анализ совмещенных датчиков, результат представлен в таблице 6.

Таблица 6 – Сравнительные характеристики датчиков температуры и влажности

	AM2302	**AM2320**	**DHT11**	**HTU21D**	**Si7021**	**BME280**
Диапазон измерения	0-100	0-100	0-60	0-100	0-100	0-100
Точность при 25°C (%RH)	±3% (10-90)	±3% (10-90%)	±3% (10-90%)	±3% (20-80%)	±3% (0-80%)	±3% (20-80%)
Повторимость (%)	±0,3	±0,1	±0,3	-	±0,025	-
Частота обновления в сек.	5	5	10	5	18	1
Рабочее напряжение (V)	3,3–5,5	3,3–5,5	3,3–5,5	1,5–3,6	1,9–3,6	1,71–3,6

Таким образом, в качестве датчика температуры и влажности был выбран DHT22, имеющий следующие характеристики:

- питание – от 3 до 5 Вольт;
- максимальный ток при преобразовании – 2,5 мА;
- измерение влажности в интервале от 0% до 100%. Точность измерений колеблется от 2% до 5%;
- минимальная измеряемая температура – 40, максимальная +125 градусов по Цельсию (точность измерений – 0,5);

– устройство способно совершать одно измерение за 2 секунд. Частота до 0,5 ГЦ;
– габаритные размеры: 15,1 мм длина, 25 мм широта, 5,5 мм высота.

3.3. ВЫБОР МИКРОКОНТРОЛЛЕРА

Центральной частью аппаратного обеспечения является микроконтроллер, предназначенный для управления. Функции управления сводятся к обработке и последующему использованию цифровой двоичной информации, поступающей от объектов управления по линиям связи от различных устройств сопряжения микроконтроллера с объектом. В качестве таких устройств выступают датчики различных аналоговых физических параметров и связанные с ними нормирующие преобразователи электрических сигналов, аналого-цифровые преобразователи, датчики цифровой информации и др. [9]. Со стороны вывода информации МК взаимодействует с цифровыми индикаторами, исполнительными механизмами, дисплеями, цифропечатающими устройствами и другими средствами запоминания, хранения и использования результатов обработки информации.

Необходимо определить требования к выбираемому микроконтроллеру. Главные из них – это возможность работы с пониженным энергопотреблением, наличие соответствующей периферии для реализации передачи данных и ее стоимость. Поскольку согласно изложенным техническим требованиям в пункте 2.2 необходимо обработанную информацию с датчиков передавать на сервер умного дома, то для этой цели может использоваться контроллер с программируемой логикой.

В марте 2014 года был предложен тест ULPBench, который помог стандартизировать параметры микроконтроллеров (МК) с ультрамалым потреблением, предложив методологию для надежного и объективного измерения энергоэффективности МК. До этого момента способов сравнения

эффективности микроконтроллеров с ультрамалым потреблением не существовало. Рабочая группа ULPBench в рамках консорциума Embedded Microprocessor Benchmark Consortium разработала и опубликовала набор стандартных тестов, который позволил бы разработчикам сравнивать микроконтроллеры разных производителей с различными особенностями.

Появление теста ULPBench CoreProfile для микроконтроллеров с ультрамалым потреблением стало помогло облегчить процесс выбора МК. Данный инструмент позволяет измерять рейтинг эффективности при заданных условиях. Чем выше балл теста, тем меньше энергопотребление. ULPBench помог определить лидеров в сегменте микроконтроллеров с ультрамалым потреблением.

Были выделены три основных фактора, определяющих уровень потребления микроконтроллеров:

- производительность;
- память;
- ток утечки.

Поставщики микроконтроллеров предлагают разработчикам несколько режимов пониженного потребления, при которых определяющий вклад в общее потребление вносят токи утечки. В то же время при использовании дополнительных рабочих режимов оценка тока потребления в мА/МГц и тока утечки значительно усложняется.

Для оценки влияния каждого из трех перечисленных выше факторов ULPBench использует два основных компонента микроконтроллера: процессорное ядро и часы реального времени RTC (real-time clock). При проведении испытания микроконтроллер выполняет определенную фиксированную задачу с периодичностью один раз в секунду, а оставшуюся часть этого секундного интервала он находится в спящем состоянии. Все тестируемые процессоры выполняют одинаковую задачу. Во многих приложениях, в которых требуется обеспечить минимальное потребление,

используется аналогичный импульсный режим работы, при котором система активна в течение коротких интервалов времени, а оставшуюся часть периода проводит в состоянии сна.

В таких приложениях учитываются три основных параметра:

- время пробуждения;
- энергия пробуждения и пиковый ток;
- время исполнения.

Для сравнительной оценки эффективности используются различные тесты с одинаковой рабочей нагрузкой и возможностью запуска на 8-битных, 16-битных, 32-битных микроконтроллерах. Эффективность работы МК оценивается как при активном режиме, так и при нахождении в режимах пониженного потребления, которые используются с учетом особенностей реальных приложений. Тестирование подразумевает использование аппаратной системы EnergyMonitor от EEMBC и набора программного обеспечения ULPBench. Функционал EnergyMonitor позволяет запускать ядро тестового ПО ULPBench на исследуемом микроконтроллере. Исходный код программного ядра ULPBench поставляется EEMBC.

В таблице 7 представлены значения рейтинга ULPBench, полученные для следующих микроконтроллеров:

- STMicroelectronics: STM32L433, STM32L0;
- Texas Instruments: MSP432 режим DC/DC; MSP432 режим LDO, MSP430FR5969 FRAM, MSP430FG4618 Flash;
- SiliconLabs: EFM32 WonderGecko, Giant Gecko, Pearl Gecko, Zero Gecko;
- Atmel: SAML21 rev A, rev B режим LPEFF On, rev B режим LPEFF Off;
- NXP: Kinetis KL27Z;
- Microchip: 16-битный PIC24FJ64GA202;
- Ambiq Micro: Apollo.

STM32L433 с ядром ARM Cortex-M4 демонстрирует максимальный рост значения ULPBench при уменьшении напряжения: от 172,58 баллов при 3,0 В до 325,78 баллов при 1,8 В, то есть больше на 153,2 балла.

Кроме того, STM32L433 является лидером рейтинга при напряжении питания более 2,1 В. Контроллер STM32L0 с ядром ARM Cortex-M0+ отметился ростом от 129,99 баллов до 226,67 баллов, тем самым заняв четвертое место в условиях пониженного напряжения.

Таблица 7 – Полученные значения рейтинга ULPBench для тестируемых микроконтроллеров

Наименование	Версия	Vdd, В												
		3	2,9	2,8	2,7	2,6	2,5	2,4	2,3	2,2	2,1	2	1,9	1,8
STM32L433	–	172,58	180,54	188,44	198,01	208,42	217,35	229,34	242,14	254,97	270,26	287,25	305,97	325,78
STM32L476	–	149,48	156,87	164,72	172,42	181,03	188,82	199,23	209,81	223,32	234,01	246,85	261,45	280,15
SAML21	Rev A	147,08	155,3	163	168,57	174,9	180,16	186,21	196,25	202,28	211,46	220,41	229,81	240,11
STM32L0	–	129,99	135,74	140,92	147,47	154,84	161,04	167,41	175,93	184,55	192,99	203,35	214,75	226,67
SAML21	Rev B LPEFF off	127	136	142,24	148,49	154,88	161,71	167,2	172,78	180,89	189,28	199,71	207,31	215,79
MSP432	LDO	117,88	124,17	128,4	134,88	140,59	147,09	154,45	161,41	169,42	178,64	188,03	197,58	208,82
Kinetis KL27	–	79,77	83,27	84,3	87,97	92,3	96,41	100,43	104,78	108,82	113,94	119,97	126,09	133,56
MSP430FR5969	–	120,5	125,17	129,96	135,25	140,73	146,5	152,89	159,41	167,06	175,17	184,07	194,32	N.A.
PIC24FJ64GA202	–	70,84	73,93	77,42	80,56	83,7	87,18	90,84	95,11	0,69	104,15	110,39	121,84	N.A.
SAML21	Rev B LPEFF on	137,33	142,68	151,43	154	158,41	167,06	170,95	178,39	183,21	187,16	186,01	N.A.	N.A.
EFM32 ZeroGicko	–	98,01	103,63	108,65	113,99	118,49	124,19	129,99	136,51	142,59	149,79	157,06	N.A.	N.A.
EFM32 GiantGicko	–	58,68	61,55	64,33	67,02	70,01	73,46	76,91	80,42	84,51	88,73	94,14	N.A.	N.A.
MSP430FG4618	–	32	43	36	37,6	40	42	45	48	50	53	57,5	N.A.	N.A.
Apollo	–	329,92	339,3	354,95	367,05	384,94	395,18	410,63	436,62	459,51	472,42	N.A.	N.A.	N.A.
EFM32 PearlGicko	–	106,25	110,44	112,29	113,51	114,85	115,82	115,63	119,31	122,33	127,06	N.A.	N.A.	N.A.
EFM32 WonderGicko	–	74,59	79,21	82,91	86,44	90,01	94,02	98,27	102,81	107,8	113,42	N.A.	N.A.	N.A.
MSP432	DC/DC	153,99	161,97	166,6	171,85	178	183	190,11	193,05	199,68	N.A.	N.A.	N.A.	N.A.

При исследовании характеристик SAML21 было выявлено, что если версия контроллера SAML21 rev A с ядром ARM Cortex-M0+ показала рост от 147,08 до 240,11 баллов, то для версии SAML21 rev B для работы с напряжениями питания до 1,8 В необходимо перевести контроллер в режим LPEFF со сбросом соответствующего бита управления в состояние «0». Как видно из таблицы 4, значение ULPBench оказалось ниже официально заявленного значения. Регулятор напряжения демонстрирует большую

надежность, если указанный бит установлен («1»), но вместе с этим при напряжениях ниже 2,1 В (по документации 2,5 В) контроллер работает нестабильно.

Характеристики микроконтроллеров производства компании EnergyMicro с ядром ARM Cortex-M0+, ядром ARM Cortex-M3+ и ядром ARM Cortex-M4 оказались ниже, чем у конкурентов. По сравнению с другими ARM-контроллерами они демонстрируют показатели ниже как по уровню прироста эффективности при снижении напряжения, так и по абсолютным показателям в целом. При уменьшении напряжения питания менее 2 В наблюдается падение эффективности.

Также по сравнению с лидерами рейтинга низкие результаты показали контроллеры NXP Kinetis KL27Z с ядром ARM Cortex-M0 и PIC24 с фирменным 16-битным ядром. Однако они позволяют использовать напряжения питания вплоть до 1,8 В (NXP) и 1,9 В (Microchip) при приемлемом повышении эффективности как минимум в процентном отношении.

В тестировании приняли участие три представителя от Texas Instruments. При работе с включенным DC/DC-регулятором микроконтроллер MSP432 с ядром ARM Cortex-M4 демонстрирует вполне достойные значения рейтинга ULPBench, однако минимальное напряжение питания при этом ограничено 2,2 В. Дальнейшее снижение напряжения при использовании DC/DC-преобразователя невозможно. Для работы с напряжениями вплоть до 1,8 В необходимо активировать LDO-регулятор, а это приводит к тому, что значение ULPBench сразу сокращается на 25%, что автоматически отбрасывает MSP432 в конец рейтинга среди контроллеров с ядром ARM Cortex-M4.

На рисунке 9 в виде графиков представлены данные из таблицы 7. В диапазоне напряжений 3,0-2,2 В значение ULPBench с различной скоростью увеличивалось для всех микроконтроллеров. Чтобы подчеркнуть, какие

контроллеры способны работать при малых значениях напряжений питания, основное внимание уделяется диапазону ниже 2,3 В. Первым рассматриваемым вариантом стали высокопроизводительные микроконтроллеры из серии STM32WB.

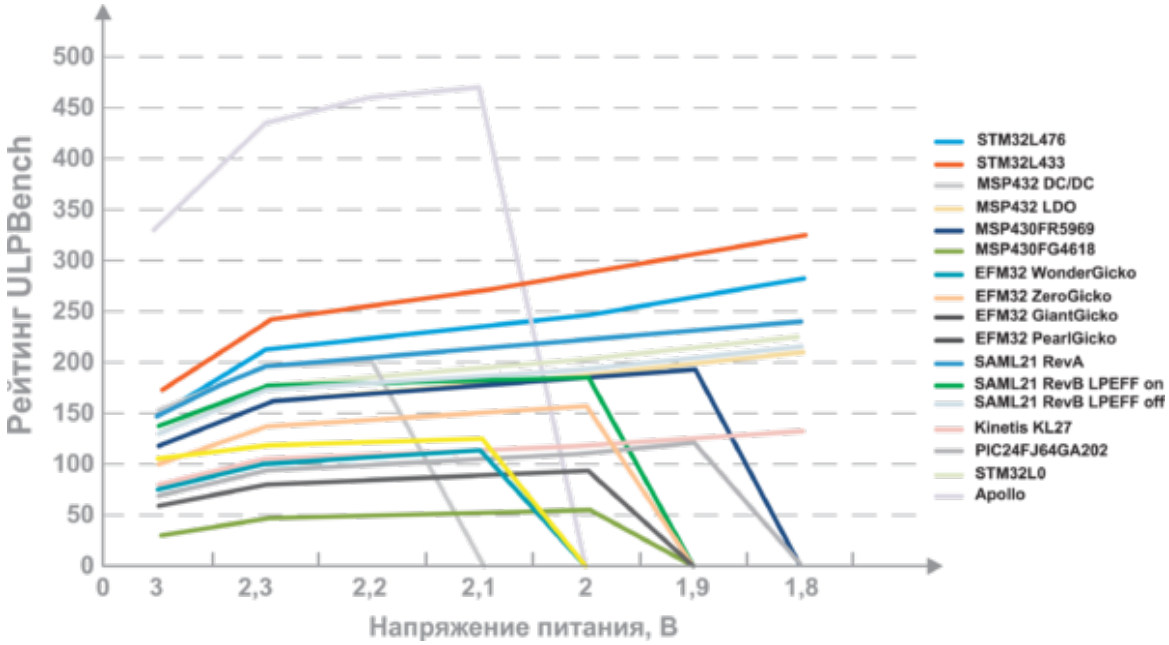

Рисунок 9 – График отображает результаты, полученные при проведении испытаний

Поскольку имелся опыт работы с данным семейством при написании выпускной бакалаврской работы. В первую очередь был проведен обзор возможностей

Это двухъядерный микроконтроллер с беспроводной связью. В дополнение к высокопроизводительному ядру ARM Cortex-M4, предназначенному для обработки приложений, микроконтроллеры STM32 включают в себя независимое ядро ARM Cortex-M0+ для управления мультипротокольным передатчиком 2,4 ГГц. Дополнительное ядро M0+ реализует технологии Bluetooth 5.0, Thread и IEEE 802.15.4 в параллельном режиме и обеспечивает управление безопасностью. Микроконтроллер STM32WB имеет энергопотребление от 50 мкА/МГц в активном режиме. Интегрированный балун обеспечивает простое подключение к антенне. Встроенное радио 2,4 ГГц контролируется ULP-ядром Cortex-M0 и

потребляет 3,8 мА в режиме приема и 5,5 мА в режиме передачи (при 0 дБм). Максимальная выходная мощность может достигать +6 дБм, обеспечивая увеличенную дистанцию связи. Радиоконтроллер содержит сертифицированные стеки протоколов, включая ST OpenThread и Bluetooth 5 с поддержкой Mesh 1.0.

STM32WB, как часть системы STM32 микроконтроллеров, предлагает широкий набор средств разработки, включая программное обеспечение STM32Cube, беспроводные стеки, библиотеки, STM32 Nucleo/Discovery наборы и платы для прототипов. Стоимость STM32WB составляет примерно 500 руб. и большой срок поставки, при мелкосерийном производстве данный показатель критичен.

Мощность потребления зависит от особенностей работы конкретного устройства. Например, сетевой Wi-Fi-процессор может быть постоянно подключен к сети, а может выполнять лишь периодические подключения для передачи данных. Во многих приложениях подключение к сети выполняется только при наступлении заданных событий, и в этом случае процессор проводит большую часть времени в спящем режиме. Следующим в качестве используемого микроконтроллера был рассмотрен беспроводной микроконтроллер SimpleLink CC3220. Он имеет два процессорных ядра: ARM Cortex-M4 для выполнения пользовательских программ и сетевой процессор для выполнения всех логических уровней протоколов WI-FI и интернета.

Технические параметры CC3220 в части пользовательского процессорного ядра имеют следующие особенности:

- микросхема CC3220R обладает 256 кбайт RAM;
- наличие таких дополнительных коммуникационных интерфейсов как двухканальный I²S, SD, SPI, I²C, UART;
- наличие 12-битного четырехканального АЦП;
- 8-битный интерфейс камеры;

– четыре таймера общего назначения с 16-разрядными ШИМ-выходами;
– до 27 входов-выходов общего назначения;
– отладочные интерфейсы JTAG, cJTAG, SWD.

На рисунке 10 представлена функциональная схема CC3220. На рынке данный микроконтроллер представлен в небольших количествах, стоимость составляет 761 рубль, что является дорого при условии серийного производства.

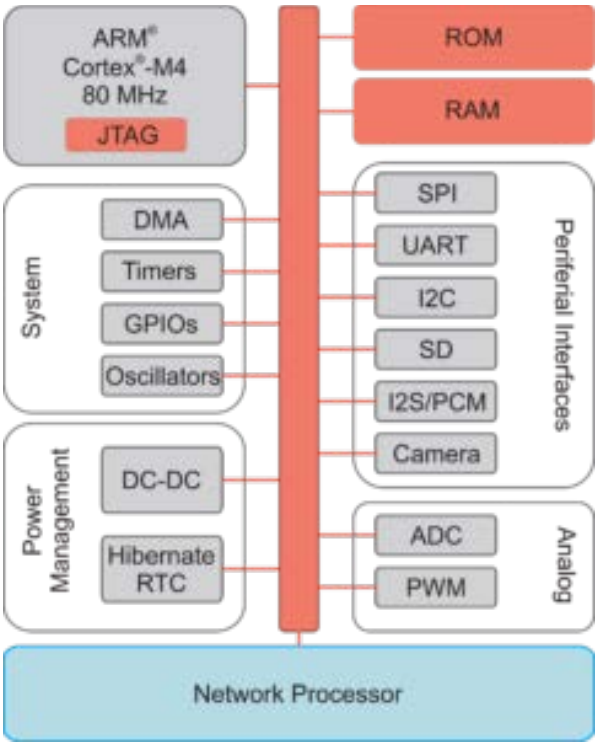

Рисунок 10 – Функциональная схема CC3220

3.4. ВЫБОР МОДУЛЯ ИНТЕРНЕТА ВЕЩЕЙ

Поскольку подбор микроконтроллера происходит с учетом использования его для устройства интернет вещей, то следующими для обзора были рассмотрены платформы интернета вещей. Под платформой понимают некоторое готовое интегрированное решение из нескольких компонентов, помогающее ускорить процесс вывода IoT-продукта на рынок.

По мнению авторов «IoT Analytics» [13], полноценной IoT-платформой следует считать такую платформу, которая позволяет разрабатывать

соответствующие приложения/решения. Они выделили восемь компонентов полноценной платформы Интернета вещей:

1. Связь и нормализация (Connectivity & normalization): за счет сведения различных протоколов и форматов данных в один «программный» интерфейс, гарантируется точная передача данных и взаимодействие со всеми устройствами.
2. Управление устройствами (Device management): то есть конфигурирование Интернет вещей и их исправная работа.
3. База данных (Database) – это масштабируемое хранилище данных.
4. Обработка и управление действиями (Processing & action management): платформа должна обеспечивать построение процессов, «триггеров событий» на основе конкретных данных сенсоров.
5. Аналитика (Analytics): анализ данных.
6. Визуализация (Visualization): то есть обеспечение построения графиков, моделей, наличие удобного интерфейса.
7. Дополнительные инструменты (Additional tools): наличие набора инструментов.
8. Внешние интерфейсы (External interfaces): одна из главных возможностей это возможность интеграции с помощью платформы. IoT-платформа должна иметь интерфейсы прикладного программирования (API), комплекты разработки программного обеспечения (SDK) и шлюзы.

В таблице 8 представлены данные показатели.

На рынке уже имеется большое количество платформ. IoT Analytics насчитали их более 450 и число продолжает расти. При этом различные платформы появляются из-за различных стратегий входа различных компаний, видения развития, подходов и начальных условий. Но на данный момент не все платформы Интернета вещей обладают перечисленными компонентами.

В 2016 году компанией Samsung было выпущено семейство Artik, в которое на данный момент входят:

– программная платформа - Artik Cloud;

– аппаратная платформа – микрокомпьютеры Artik.

Таблица 8 – Восемь компонентов полноценной IoT-платформы

База данных Репозиторий, в котором хранятся важные наборы данных	**Внешние интерфейсы** API, SDK и шлюзы, которые служат интерфейсами для сторонних систем (ERP, CRM)	
	Аналитика Алгоритмы для сложных вычислений и машинного обучения	**Вспомогательные инструменты** Прототипирование приложений, управление доступом, отчёты
	Визуализация данных Графическое отображение показаний датчиков (возможно, в реальном времени)	
	Обработка данных и управляющие воздействия Механизм правил, который позволяет в реальном времени совершать действия, основываясь на показаниях сенсоров и состояниях устройств	
	Менеджер устройств Инструмент для управления статусами устройств, удаленной развертки ПО и обновлений	
	Связность и нормализация данных Агенты и библиотеки, которые позволяют поддерживать постоянную целостность и связность данных	

Рассмотрев Samsung Artik Cloud с точки зрения перечня функциональных компонентов (таблица 8), получились данные, отраженные в таблице 9.

Таблица 9 – Обзор Samsung Artik Cloud

База данных Есть, работает.	**Внешние интерфейсы** Поддержка 4 наиболее распространенных API: REST, Websockets, MQTT, COAP	
	Аналитика На данный момент отсутствует	**Вспомогательные инструменты** Из полезных инструментов можно назвать симулятор данных, утилиту для обмена данными со сторонними облаками Cloud Connector, а также тесную интеграцию с устройствами SmartThings
	Визуализация данных Есть инструмент для просмотра 2D-графиков	
	Управление действиями Есть механизм правил – Rules – доступный из веб-интерфейса, похожий на IFTTT (If This Then That)	
	Менеджер устройств Есть, позволяет узнавать статус устройств и удаленно выполнять функции перезагрузки, обновления и пр.	
	Связность и нормализация данных Внутреннее представление – в формате JSON. Для пользователя есть возможность произвести нормализацию данных средствами самой системы: это делается на языке Java в процессе написания манифеста (документа о структуре данных)	

Из ключевых компонентов на данный момент отсутствует компонент аналитики и машинного обучения. Компания Samsung предоставляет три платы для разработчиков (Artik 1, Artik 5, Artik 10). Инфраструктура формиркется благодаря наличию программных инструментов и облачных сервисов.

У модуля Artik 1 имеется двухъядерный процессор, Bluetooth Low Energy, гироскоп и акселерометр.

У модуль Artik 5 имеется двухъядерный чип ARM, WI-FI, Zigbee, Bluetooth Low Energy. Данному решению было уделено больше всего внимания. На рисунке 11 представлен внешний вид ARTIK 520 Module.

ARTIK 520 Module имеет следующие характеристики:

- Высокопроизводительный 4-ядерный 32-разрядный процессор ARM Cortex-A9 с частотой 1,2 ГГц.
- ARM MALI GPU для мультимедиа, графических приложений.
- 512 МБ ОЗУ, 4 ГБ флэш-памяти (eMMC).
- Поддержка WLAN, Bluetooth, ZigBee, Thread.

Рисунок 11 – Внешний вид ARTIK 520 Module

Данный комплект разработчика имеет ряд преимуществ, но его стоимость по данным [14] составляет более 10 000 рублей. При этом не учитываются другие компоненты разрабатываемого комплекса.

Далее был произведен обзор модулей и платформы быстрой разработки электронных устройств Arduino. Платформа пользуется популярностью во всем мире из-за удобства и простоты языка программирования. Еще одним из преимуществ является открытая архитектура и программный код. Программирование устройства производится через USB без использования программаторов.

С помощью языка Arduino, который был основан на языке Wiring, можно программировать микроконтроллер, расположенный на плате. Программирование происходит в среде разработки Arduino. Отличительной чертой является автономность работы проектов, основанных на Arduino, но данный факт не исключает взаимодействие с ПО на компьютере. Все ПО можно скачать бесплатно. Также в открытом доступе имеются чертежи схем.

Ещё одной отличительной особенностью Arduino является наличие плат расширения (shields). Это дополнительные платы, которые ставятся подобно поверх Arduino, чтобы дать ему новые возможности. Из линейки плат Arduino для разработки программно-аппаратного комплекса был выбран WI-FI модуль ESP8266. Он предназначен для того, чтобы управлять устройством дистанционно или чтобы снимать показания с сенсоров через интернет. На рисунке 12 представлен внешний вид устройства.

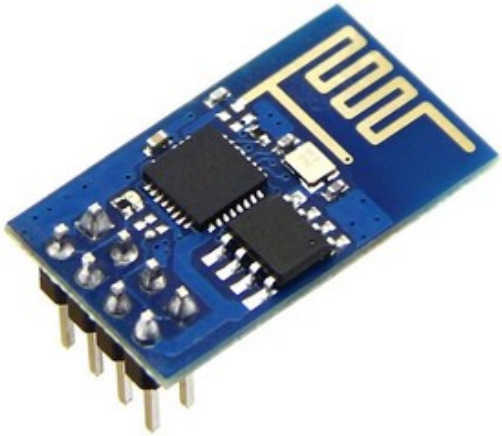

Рисунок 12 – Внешний вид WI-FI модуля ESP8266

Характеристики:

– модификация: ESP-01 V090;

– беспроводной интерфейс: WI-FI 802.11 b/g/n 2,4 ГГц;
– номинальное напряжение 3,3 В;
– максимальный потребляемый ток 220 мА;
– 2 шт. портов ввода-вывода свободного назначения;
– частота процессора составляет 80 МГц;
– объём памяти для кода 64 КБ;
– объём оперативной памяти 96 КБ;
– габариты: 21×13 мм.

Управляющее устройство общается с ESP8266 через UART (Serial-порт) с помощью набора АТ-команд. Модуль можно перепрограммировать через Arduino IDE.

Поскольку напряжение модуля 3,3 вольта, то его пины не толерантны к 5 вольтам. Поэтому для передачи данных на модуль с 5-вольтовых управляющих плат необходимо использовать делитель напряжения, чтобы перевести напряжение в допустимый диапазон.

Для работы с модулем необходим Arduino контроллер. Линейка контроллеров разнообразна, но необходимо обратить внимание на характеристики платы. Стоимость WI-FI модуля ESP8266 по данным [12] составляет 390 рублей. Но в линейке продукции Arduino появилось устройство, объединяющее в себе вычислительное ядро и WI-FI модуль, поэтому при его использовании не потребуется дополнительный контроллер.

Arduino MKR WI-FI 1010 – это платформа для разработки на базе микроконтроллера ATSAMD21G18 с вычислительным ядром ARM CortexM0. На рисунке 13 представлен внешний вид устройства. Используется 32-разрядное ядро ARM, Arduino MKR WI-FI, которое превосходит типичные платы на базе 8-разрядных микроконтроллеров. Наиболее существенные отличия заключаются в следующем:

– 32-битное ядро позволяет обрабатывать четырёх байтовые данные всего за один такт.

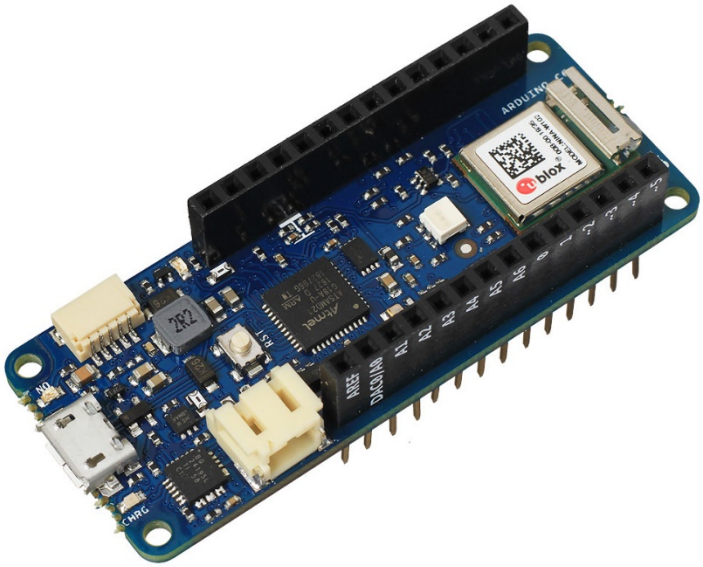

Рисунок 13 – Внешний вид Arduino MKR WI-FI 1010

– Тактовая частота составляет 48 МГц.
– Объем памяти программ Flash 256 КБ.
– Объем оперативной памяти SRAM 32 КБ.

На платформе предусмотрен JST SH-разъём (5 pin) для подключения дополнительных модулей по интерфейсу «I²C» (рис. 14).

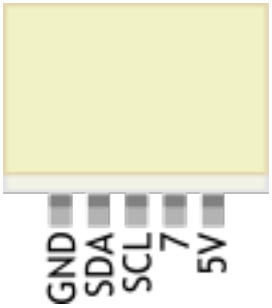

Рисунок 14 – Распиновка JST SH-разъёма

Коннектор включает в себя:

– пины питания 5V и GND;
– пины шины I²C SDA и SCL;
– дополнительный цифровой пин 7.

Линейный понижающий регулятор напряжение AP7215-33 с выходом 3,3 вольта обеспечивает питание микроконтроллера. Максимальный выходной ток составляет 600 мА.

Пины питания:

- VIN: пин для подключения внешнего источника напряжения в диапазоне от 5 до 6 вольт.
- 5V: при подключении платы через USB на вход поступает 5 В. При питании платформы через пин VIN или разъём для внешнего аккумулятора на пине может быть напряжение в диапазоне от нуля до входного.
- VCC: пин от стабилизатора напряжения с выходом 3,3 вольта и максимальных током 600 мА. Регулятор обеспечивает питание микроконтроллера ATSAMD21G18. В любом варианте питания платформы на пине будет присутствовать 3,3 вольта.
- GND - это выводы земли.
- AREF: пин для подключения внешнего опорного напряжения АЦП относительно которого происходят аналоговые измерения при Порты ввода/вывода:
- входы/выходы: 22 пина; 0–14 и A0–A6.
- ШИМ: 12 пинов; 0–8, 10, A3 и A4. Позволяет выводить аналоговые значения в виде ШИМ-сигнала.
- АЦП: 7 пинов; A0–A6. Позволяет представить аналоговое напряжение в виде цифровом виде. По умолчанию разрядность «АЦП» установлена в 10 бит.
- ЦАП: пин DAC/A0. Аналоговый выход цифро-аналогового преобразователя, который позволяет формировать 10-битные уровни напряжения.
- TWI/I²C: пины (11)SDA и (12)SCL.
- SPI: пины 8(MOSI), 10(MISO) и 9(SCK).
- UART/Serial: Serial: пины D+ и D− и Serial1: пины 12(RX) и 11(TX).

– I²S: пины 2(SCK/BCLK), 3(WS/LRCLK/FS) и A6(SD/SDATA/SDIN/SDOUT).

Основные характеристики:

– беспроводной модуль: NINA-W10 на чипе ESP32 с WI-FI и Bluetooth;
– микроконтроллер: ATSAMD21G18;
– ядро: 32-битный ARM Cortex M0+;
– тактовая частота: 48 МГц;
– flash-память: 256 КБ (8 КБ занимает загрузчик) ;
– SRAM-память: 32 КБ;
– 22 пинов ввода-вывода;
– 7 пинов с АЦП;
– разрядность АЦП: 8/10/12 бит (по умолчанию 10 бит);
– разрядность ШИМ: 8/10/12 бит (по умолчанию 8 бит);
– 1 аппаратный интерфейс SPI;
– 1 аппаратный интерфейс I2C / TWI;
– 1 аппаратный интерфейс UART / Serial;
– номинальное рабочее напряжение: 3,3 В;
– максимальный выходной ток пина 3V3: 600 мА;
– максимальный ток с пина или на пин: 7 мА;
– входное напряжение через пин Vin: 5–6 В;
– входное напряжение через разъём для аккумулятора: 3,7 В;
– габариты: 62×25 мм.

Стоимость данного модуля согласно [14] составляет 3 390 руб. Но для того, чтобы он удовлетворял требованиям, описанными в пункте 2.2, необходимо выбрать дисплей для отображения информации. Был выбран встраиваемый RGB OLED дисплей с интерфейсом SPI вынесенным на угловой штыревой разъем. На рисунке 15 представлен внешний вид устройства.

Рисунок 15 – Внешний вид RGB OLED дисплея

Основные характеристики:

- чип драйвера SSD1331;
- интерфейс SPI;
- разрешение 96x64;
- размер дисплея 0,95 дюйма;
- размер 31,7мм * 37мм;
- цвета RGB;
- угол обзора> 160 °C;
- рабочая температура (℃)-20 ~ 70.

Стоимость данного дисплея согласно [14] составляет 1 040 рублей.

В случае выбора данного варианта помимо элементов периферийной части необходим корпус и кнопки для управления. Поскольку прибор будет иметь бытовое назначение, то с точки зрения пользователя данный вариант не является удобным и эстетически привлекательным. Таким образом, можно сделать вывод, что в качестве прототипа данный вариант подходит, но если рассматривать мелкосерийное производство, то необходимо рассмотреть другие варианты.

Ранее был рассмотрен WI-FI модуль ESP8266, поскольку он имеет мощные характеристики, то было решено рассмотреть модуль ESP32. Это единый комбинированный чип WI-FI и Bluetooth 2,4 ГГц, разработанный с использованием сверхмалого энергопотребления. Он предназначен для достижения максимальной мощности и радиочастотных характеристик,

демонстрируя надежность и универсальность. ESP32 разработан для мобильных устройств, носимой электроники и приложений Интернета вещей (IoT). Низкий рабочий цикл используется для минимизации количества энергия, которую расходует чип. На рисунке 16 представлен внешний вид устройства.

Рисунок 16 – Внешний вид модуля ESP32

Далее был произведен обзор основных характеристик:

- двухядерный 32-битный микропроцессор Xtensa LX6 160-240МГц;
- ОЗУ - 520Кб;
- ПЗУ - 448Кб;
- наличие RTC таймера с 16Кб ОЗУ;
- внешняя флэш память 4-16Мб;
- питание 2.2 - 3.6В;
- WI-FI 802.11n 2.4Гц с максимальной скоростью 150Мбит/сек.
- Периферийная часть:
- 12- битный АЦП на 18 портах.
- 8-битный ЦАП на 2 портах.
- 10-портов в режиме сенсорных кнопок.
- Встроенный температурный датчик.
- 4 x SPI, 2 x IS1, 2 x I2C, 3 x UART.
- 1 host (SD/eMMC/SDIO).
- 1 slave (SDIO/SPI).
- CAN 2.0, IR (TX/RX).

Энергопотребление:

- максимальный ток при передаче WiFi - 160-260мА;
- потребление без включенного WiFi и Bluetooth - 20мА.

Данный модуль отличается большим разнообразием модификаций, доступностью к покупке и при достойных характеристиках недорогой стоимостью, согласно [14] стоимость составляет 140 рублей. Несмотря на то, что микроконтроллер ESP32 был выпущен компанией Espressif в 2015 году, он до сих пор остается наиболее востребованным вариантом для построения устройств интернета вещей. Такая крупная компания как Xiaomi используют его в своих разработках. С учетом ранее подобранного RGB OLED дисплея, можно сделать вывод, что данный вариант можно считать наиболее экономичным и доступным к покупке, но требуется тщательное конструирование корпуса и комплектация дополнительными электронными компонентами, что является минусом перед платформами Интернета вещей. Поэтому был рассмотрен еще один вариант, основанный на ESP32, являющийся полноценной платформой Интернета вещей.

Платформа M5Core GRAY построена на чипе ESP32, поэтому имеет большой функционал и все возможности этого микроконтроллера. Работает на частоте 240 МГц и объем памяти составляет 2 Мб. На рисунке 17 представлен внешний вид устройства.

На лицевой части расположен цветной LCD экран 320 на 240 точек, на него можно выводить как текстовую информацию, так и рисовать. Под экраном находятся три кнопки, на них можно назначать выполнение разных функции и обработки. В составе имеется встроенный WI-FI для подключения к сети, Bluetooth, встроенные динамики, картридер для микро SD карт памяти.

Платформа сразу поставляется с модулем автономного питания на 150 миллиампер, который уже установлен в съемную плату расширения.

Рисунок 17 – Внешний вид M5Core GRAY

Основные характеристики:

- наличие ESP32, 240 МГц, два ядра, 600 DMIPS, 520 КБ SRAM, WI-FI, двухрежимный Bluetooth;
- память Flash 16MB;
- напряжение 5В при 500мА;
- ЖК-дисплей 2 дюйма, цветной цветной TFT-дисплей 320x240;
- динамик 1W-0928;
- микрофон MEMS аналоговый BSE3729;
- аккумулятор 550mAh;
- размер 54 x 54 x 12,5 мм;
- вес 120 г.

Ток потребления при работе платформы в демонстрационном режиме 185 миллиампер. Таким образом получается, что аккумулятора хватит на 45 минут автономной работы. Базовая комплектация включает в себя:

- 1x модуль M5Core GRAY.
- 1x модуль M5Core Bottom.
- Type-C USB кабель.

Стоимость данного набора согласно [17] составляет 2 594 рубля. Модуль является расширяемым и может быть дополнен другими блоками. Так, например, для увеличения автономной работы устройства достаточно докупить M5Stack BATTERY Module с емкостью 700 мАч. Его стоимость [17] составляет 872 рубля.

Таким образом, в результате обзора микроконтроллеров, выполняющих требования пункта 2.2. была составлена таблица 10, которая отражает стоимость каждого микроконтроллера из рассмотренных. Но данные показатели нельзя считать за конечные, поскольку в сравнении участвуют как отдельные микросхемы микроконтроллеров, так и готовые платформенные решения. Поэтому с учетом комплектации был произведен пересчет итоговой стоимости базового набора разработки. Для этого с наименования 1-5 необходимо добавить стоимость дисплея и корпуса.

Таблица 10 – Обзор стоимости микроконтроллеров

№	Наименование	Стоимость, руб.
1	STM32WB55VGY6TR	500,00
2	CC3200R1M2RGC	761,00
3	ARTIK 520 Module	10 180,00
4	Arduino MKR Wi-Fi 1010	3 390,00
5	Модуль ESP32	141,00
6	M5Core GRAY	2 594,00

Дисплей был выбран ранее, его характеристики удовлетворяют пункту 2.2 и его стоимость составляет 1 040 рублей.

Далее необходимо выбрать корпус. Большой ассортимент пластиковых корпусов и доступность покупки может предложить компания «Планар-СПб», которая поставляет на российский рынок корпуса для радиоэлектронной аппаратуры различного назначения. Корпуса в зависимости от назначения подразделяются на приборные, корпуса для пультов управления и т.д. Для выбора корпуса необходимо учитывать габаритные размеры устройства и условия эксплуатации. В результате обзора [18] были взяты модели корпусов, отраженные в таблице 11.

Таблица 11 – Обзор стоимости корпусов

№	Наименование	Стоимость, руб.
1	G968B	478,55
2	G999B	667,78
3	G1183B	307,84
4	PL2943BK	280,30
5	G413	271,00

Для дальнейшего подсчета взято среднее значение стоимости пластиковых корпусов, которое составило 401 рубль. Следовательно, таблицу 8 можно преобразовать с учетом этих дополнений. В результате получится таблица 12.

Таблица 12 – Обзор стоимости базовых комплектов разработки

№	Наименование	Стоимость, руб.	Итого, руб.
1	STM32WB55VGY6TR	500,00	1 941,00
2	CC3200R1M2RGC	761,00	2 202,00
3	ARTIK 520 Module	10 180,00	11 621,00
4	Arduino MKR WI-FI 1010	3 390,00	4 831,00
5	Модуль ESP32	141,00	1 582,00
6	M5Core GRAY	2 594,00	2 594,00

Необходимо учитывать стоимость изготовления плат и компонентов обвязки вариантов 1 и 2, поэтому данные варианты не подлежат дальнейшему рассмотрению. Из таблицы 9 можно заметить, что наибольшую стоимость составляет ARTIK 520 Module, что является критичным при дальнейшем производстве программно-аппаратного комплекса.

Для сравнения трех последних вариантов, были введены три новых характеристики:

- эстетическая составляющая;
- доступность к покупке;
- сложность реализации.

Таким образом, был выбран M5Core GRAY, поскольку все три параметра сравнения имеют положительный ответ, а также платформа легко расширяется дополнительными модулями, соединяясь один поверх другого. Такая сцепка не требует дополнительных проводов и можно быстро

переделать проект из одного под другую задачу используя один и тот же контроллер и модули.

В результате были получены данные, отраженные в таблице 13.

Таблица 13 – Обзор стоимости базовых комплектов разработки

Наименование	Стоимость, руб.	Эстетика	Простота покупки	Реализация
Arduino MKR WI-FI 1010	4 831,00	-	+	+
Модуль ESP32	1 582,00	-	+	+
M5Core GRAY	2 594,00	+	+	+

В базовую комплектацию M5Core GRAY входит батарея с емкостью 250 мАч. Для выполнения условий в п.2.2, то есть обеспечения автономной работы устройства, данного показателя недостаточно. Поэтому был использован модуль расширения аккумулятора Module BATTERY с емкостью 700 мАч, который подключается к основной плате через разъем.

3.5. ПРОЕКТИРОВАНИЕ ЭЛЕКТРИЧЕСКОЙ ПРИНЦИПИАЛЬНОЙ СХЕМЫ ДАТЧИКОВ

После выбора главных составляющих компонентов устройства, была спроектирована электрическая принципиальная схема с датчиками, представленная в приложении А.

Поскольку модуль M5Core GRAY имеет разъем для подключения платы расширения, то данная особенность использовалась для связи данных частей устройства.

В качестве пинов для сбора и обработки информации с датчиков, на модуле ESP32 были выбраны: GPIO16, GPIO17, GPIO2, GPIO5. В таблице 14 представлено их назначение.

Таблица 14 – Назначение контактов

Порт МК	**Контакт разъема**	**Функциональное назначение**
GPIO16	15	Аппаратный UART
GPIO17	16	
GPIO2	19	GPIO
GPIO5	20	1-Wire

Спецификация выбранных компонентов представлена в приложении Б.

4. РАЗВОДКА ПЕЧАТНОЙ ПЛАТЫ

4.1. ВЫБОР И ОБОСНОВАНИЕ КОНСТРУКЦИИ ПЕЧАТНОЙ ПЛАТЫ

Поскольку содержится небольшое количество корпусов (7 штук), то необходимо выбрать одностороннюю печатную плату (ОПП).

Размещение элементов производится таким образом, чтобы электрические соединения были минимальной длины. Элементы необходимо располагать как можно более равномерно по площади печатной платы для обеспечения равномерности масс элементов.

Разводка печатной платы осуществлялась в САПР DipTrace. Спроектированная плата является односторонней, ее можно отнести к третьему классу точности, так как минимальное расстояние между проводниками составляет 0.3 мм.

Учитывая габаритные размеры корпуса M5Core GRAY и расположение разъема для соединения, были выбраны габаритные размеры печатной платы равные 50*50 мм, представленные на рисунке 18. Схема полученной печатной платы, представлена в приложении В. Взят текстолит СФ-1-1,5 мм.

В левой части расположен разъем DS1021-2x10, служащий для соединения платы с основной частью M5Core GRAY. В верхней части датчик углекислого газа MH-Z19. Его расположение было выбрано так, чтобы в последствии при проектировании корпуса учесть вентиляционные отверстия, необходимые для его исправной работы. В нижней части расположен датчик влажности и температуры AM2302. Конденсаторы C1 и C2 находятся как можно ближе к выводам питания и земли.

Для линии питания +3,3В выбрана ширина проводника равная 0,6 мм, для линий данных 0,4 мм. В приложении В представлены изображения разведенной печатной платы.

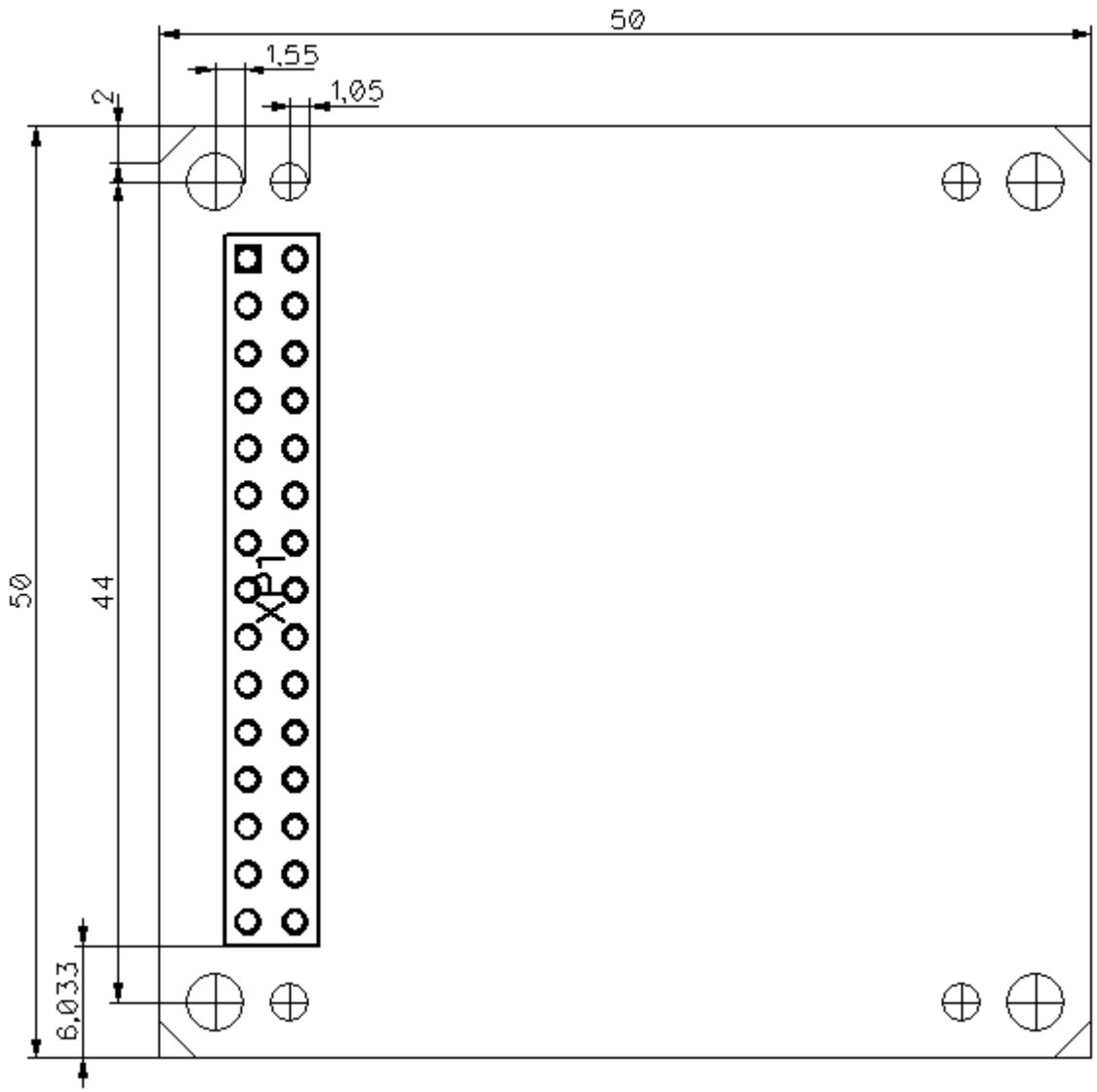

Рисунок 18 – Габаритные размеры платы

5. ПРОЕКТИРОВАНИЕ КОРПУСА УСТРОЙСТВА

С учетом габаритов модуля M5Core GRAY и конструкторских особенностей устройства, отвечающих функционалу, был спроектирован корпус. Поскольку модуль M5Core GRAY имеет возможность расширяемости, то необходимо учитывать расположение монтажных отверстий, а также форму и габариты корпуса. На рисунке 19 представлена модель полученного корпуса в САПР SolidWorks.

Рисунок 19 – Корпус

На боковых панелях расположены прорези для проникания воздуха из окружающей среды в устройство. Для этой же цели сделаны отверстия диаметром 1,0 мм на задней поверхности корпуса. Чертеж корпуса представлен в приложении Г.

6. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

6.1. АРХИТЕКТУРА ИНТЕРНЕТА ВЕЩЕЙ

Согласно исследованиям Российской ассоциации электронных коммуникаций (РАЭК): Интернет вещей представляет собой не одну технологию, а целую систему технологических решений. Это глобальная сетевая инфраструктура, которая состоит из вычислительных сетей физических объектов, традиционной IP сети Интернет и различных устройств (Gateway, Border router и т.д.), объединяющих эти сети [24]. В набор элементов экосистемы интернета вещей входят следующие составляющие:

- поставщики технологических решений (компании, производящие оборудование и ПО, предоставляющие облачные сервисы, провайдеры);
- потребители технологических решений;
- инфраструктура (технологии, платформы, аналитика, данные, решения в сфере безопасности);
- регулирование и стандарты.

Международная компания-разработчик программного обеспечения для двухмерного и трёхмерного проектирования «PTC, Inc» выделяет свои модули IoT [25]. По их мнению Интернет вещей состоит из трех основных компонентов:

- набор интеллектуальных, поддерживающих сетевые функции изделий;
- систем изделий;
- других «вещей», связанных через коммуникационную инфраструктуру.

Архитектура Интернета вещей демонстрирует связь друг с другом и лежащие в ее основе инфокоммуникационные технологии [26]. Она схематично показывает, каким образом устройства, сети и услуги участвуют

в процессе сбора, обработки, анализа данных и доставки результатов потребителю, давая более полное и системное представление об Интернете вещей.

На данный момент не существует общепринятой эталонной модели архитектуры. Каждая из официальных организаций предлагает свои определения. Ученые полагают, что архитектура IoT обрастает все новыми технологическими слоями, усложняясь по мере развития. Среди организаций, предложивших свои версии архитектуры, можно выделить Международный союз электросвязи (МСЭ), Всемирный форум IoT (Internet of Things World Forum, IoTWF), Европейскую комиссию, а также исследовательские агентства, подобные IoT Analytics. В представленных ими архитектурах существуют различия, это можно объяснить тем, что перед ними стояли различные задачи. Исследовательские комиссии Сектора стандартизации электросвязи МСЭ–Т разработали целую серию рекомендаций по IoT – «Глобальная информационная инфраструктура, аспекты протокола Интернет и сети последующих поколений». Вопросы архитектуры интернета вещей рассматриваются в рекомендациях МСЭ–Т Y.2060 «Обзор интернета вещей» [31].

Согласно МСЭ–Т Y.2060 эталонная модель включает в себя четыре уровня (рис.20):

- сети;
- приложения;
- устройства;
- поддержки услуг и поддержки приложений.

В модель также включены возможности управления и обеспечения безопасности, которые связаны с этими четырьмя уровнями.

Согласно Приказу Министерства цифрового развития, связи и массовых коммуникаций Российской федерации [32], в рассматриваемой

модели IoT уровень устройства состоит как из самих конечных устройства, так и из промежуточных шлюзов.

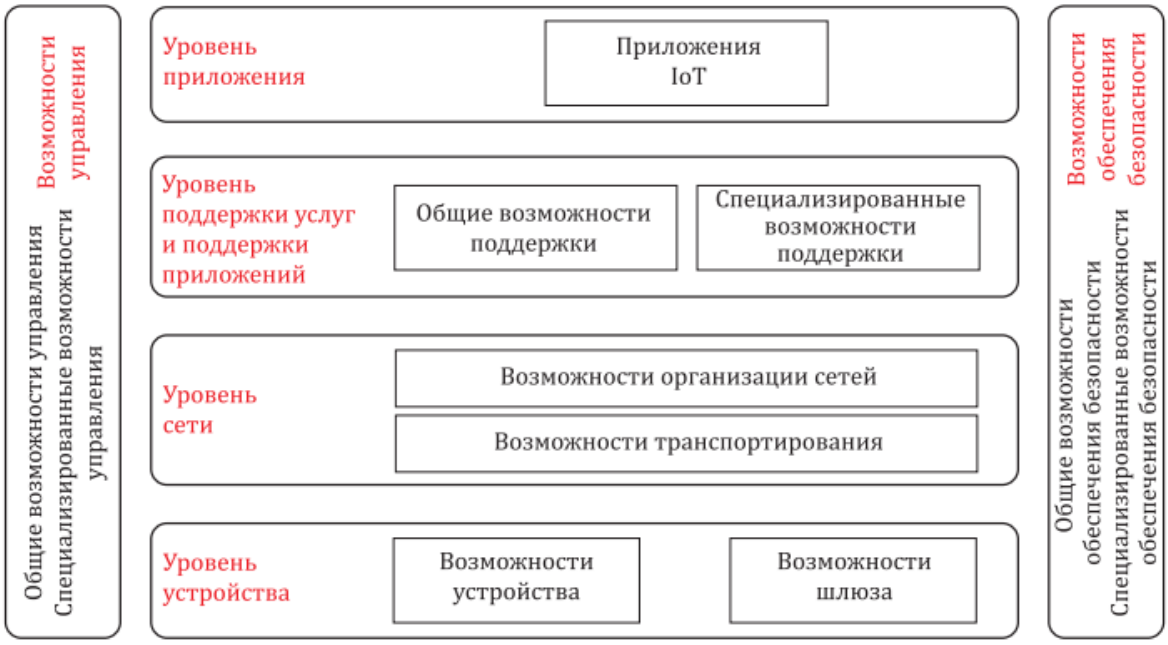

Рисунок 20 – Эталонная модель IoT, МСЭ–Т Y.2060

При этом устройства могут обладать способностью собирать и получать информацию непосредственно из сети связи, а также непрямым образом. В качестве шлюза выступают точки доступа, соединенные с оконечными устройствами с использованием различных проводных и беспроводных технологий (CAN, ZigBee, Bluetooth, Wi-Fi и др.). Шлюз необходим при наличии разнородных устройств или при осуществлении конвертации протоколов передачи. При этом шлюзы подключаются к уровню сети с использованием различных технологий: коммутируемая телефонная сеть общего пользования, сеть сотовой связи, Ethernet, линия DSL и т.д.

Уровень сети выполняет две базовых задачи:

– организация сетей;
– транспортировка информации.

В рамках организации сети предоставляются соответствующие функции управления сетевыми соединениями, такие как функции управления

доступом и ресурсом транспортирования, управление мобильностью и аутентификация, авторизация и учет.

Далее были рассмотрены каждый из уровней. На уровне поддержки услуг приложения используют предоставленные им возможности поддержки. То есть управление БД и обработка данных. Также перекрываются потребности конкретного подмножества приложений Интернета вещей.

Уровень приложений состоит из всех приложений, взаимодействующих с устройствами Интернета вещей.

Управление сетью происходит на уровне возможностей управления. Контролируется конфигурация, управляются показатели работы и безопасность.

Как видно из приведенной модели IoT, верхние уровни модели описаны максимально абстрактно без какой-либо привязки к технологической основе. Причем сетевой компонент фигурирует только на двух нижних уровнях (уровень устройства и уровень сети), а также в некоторой части возможностей управления.

В 2014 году Организационный комитет Всемирного форума IoT представил свой вариант эталонной модели (рисунок 21).

Эталонная архитектура, предложенная Всемирным форумом, имеет семь уровней.

Поскольку цель Всемирного форума IoT – ускорить развертывание Интернета вещей, то объединяя рыночных игроков, Всемирный форум решает вопросы разработки приложений, промежуточного ПО и функций поддержки для корпоративного сегмента. Отличие этой модели от архитектуры МСЭ состоит в том, что в последней акцент сделан на уровнях устройства и шлюза, а верхние слои описываются только в общих чертах.

Рисунок 20 – Эталонная модель Всемирного форума IoT

Наравне со всеми Европейский союз разработал свою архитектуру интернета вещей. Она представлена на рисунке 21.

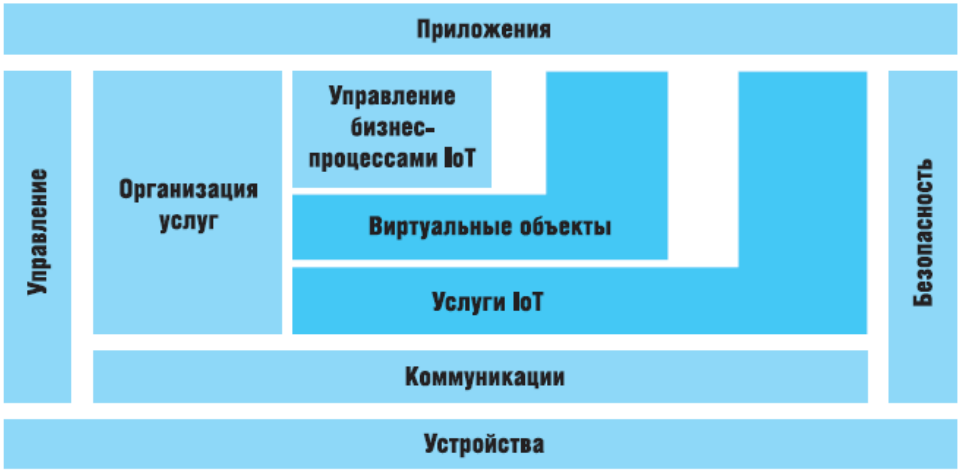

Рисунок 21 – Эталонная модель Европейского союза

Главной целью проекта было достижение совместимости систем IoT, поэтому в этой референтной архитектуре возможна полная интеграция RFID–меток и узлов беспроводных сенсорных сетей, однако большинство решений пока еще основано на системе EPCglobal.

6.2. СТАНДАРТИЗАЦИЯ ИНТЕРНЕТА ВЕЩЕЙ

Нормативно-техническое регулирование интернета вещей и смежных областей осуществляется на нескольких уровнях. В нем принимают участие международные, региональные и национальные организации по стандартизации, различные неправительственные ассоциации, союзы и консорциумы, альянсы производителей и операторов, партнерские проекты, а также отдельные исследовательские группы. Различными аспектами тематики интернета вещей, по приблизительным оценкам из открытых источников, занимаются более 100 таких организаций.

Ведущую роль в создании и разработке международных стандартов играют Международная организация по стандартизации (International Organization for Standardization, ISO, ИСО), Международная электротехническая комиссия (International Electrotechnical Commission, IEC, МЭК) и Международный союз электросвязи (International Telecommunication Union, ITU, МСЭ).

Согласно [30] подавляющее большинство беспроводных сетей для IoT возможно классифицировать в рамках шести крупных сегментов, показанных на рисунке 22.

При этом узкополосные беспроводные сети связи IoT соответствуют двум отдельным сегментам в зависимости от использования полос радиочастот в общем или упрощенном порядке. Модель использования радиочастотного спектра во многом определяет и технологическую основу сетей. Узкополосные беспроводные сети связи IoT, использующие РЧС в упрощенном порядке, представляют собой новые закрытые или открытые стандарты, разрабатываемые различными консорциумами и ассоциациями в рамках полос радиочастот, используемых устройствами малого радиуса действия.

Рисунок 22 – Классификация основных беспроводных технологий для IoT

Узкополосные беспроводные сети связи Интернета вещей в полосах радиочастот, представлены тремя стандартами: EC-GSM, eMTC (также называется LTE-eMTC) и NB-IoT. Фактически все три технологии не являются самостоятельными стандартами, а представляют собой развитие существующих стандартов сотовой подвижной связи, доработанных для удовлетворения потребностей в подключении маломощных устройств, работающих, как правило, от батареи и имеющих ограниченные потребности в пропускной способности В Российской Федерации в настоящее время существует несколько компаний, развивающих свой собственный стандарт радиоинтерфейса и экосистему всех уровней модели «Интернета вещей». В таблице 15 дано краткое описание стандартов и их основных характеристик.

Также присутствует большое количество локальных производителей оборудования и операторов, использующих международный стандарт LoRa.

На рисунке 23 изображена укрупненная схема участников работы над стандартами. Исходя из схемы, наиболее статусными и значимыми в

международном масштабе являются Международные организации в области стандартизации.

Таблица 15 – Основные параметры узкополосных беспроводных сетей связи IoT.

Характеристики	**EC-GSM**	**LTE-eMTC**	**NB-IoT**
Диапазон радиочастот, МГц	900, 1800	Диапазоны LTE	Диапазоны LTE, в том числе 450, 800, 900, 1800, 2100, 2600 (FDD)
Количество диапазонов в устройстве	1 или 2	Несколько	Многодиапазонные чипы
Ширина радиочастотного канала	200 кГц	Задействуется шесть ресурсных блоков (1,08 МГц) в канале 5 МГц и шире	180 кГц
Число устройств IoT на сектор БС на один канал, ед., не более	50000	50000	50000
Скорость передачи данных	70 или 240 кбит/с (GMSK или 8PSK)	1 Мбит/с	127 кбит/с (линия вниз) 158 кбит/с или 15.6 кбит/с (линия вверх)
Бюджет радиолинии	До 154 дБ для UE с классом мощности 23 дБ или на 10 дБ.	До 159 дБ для UE с классом мощности 23 дБ или на 15 дБ	До 164 дБ для UE с классом мощности 23 дБ или на 20 дБ лучше GPRS
Мобильность	Полная	Полная	Ограниченная
Задержка	Секунды	Миллисекунды	Секунды

К которым относятся ИСО и МЭК (с соответствующим совместным подкомитетом ИСО/МЭК СТК 1/ПК 41 «Интернет вещей и смежные технологии» (ISO/IEC JTC 1/ SC 41 «Internet of Things and related technologies»).

Отдельно можно отметить такие национальные организации как:

– Американский национальный институт стандартов (American National Standards Institute, ANSI);

Рисунок 23 – Участники разработки стандартов, рекомендаций и протоколов в сфере Интернета вещей

– Британский институт стандартов (British Standards Institution, BSI). Немецкий институт по стандартизации (Deutsches Institut fur Normung e.V., DIN);
– Немецкая Комиссия по Электротехнике, Электронике и Информационным Технологиям (DKE) при Немецком институте по стандартизации и Немецком Союзе электротехники (German Commission for Electrical, Electronic & Information Technologies of DIN and VDE);

Одним из центральных в совместной деятельности ИСО и МЭК является Совместный технический комитет № 1 «Информационные технологии» – Joint Technical Committee (СТК 1 ИСО/МЭК, JTC1 ISO/IEC). Комитет по информационным технологиям был создан для объединения деятельности двух родственных организаций. Основная цель заключается в поддержке, разработке, содействии развитию ИТ-стандартов, в которых

нуждается рынок для согласования требований производителей и потребителей по следующим направлениям:

- разработка и развитие ИТ-систем;
- производительность и качество ИТ-продуктов;
- безопасность ИТ-систем и информации;
- мобильность;
- интероперабельность ИТ-продуктов и систем (способность программного продукта или системы функционировать во взаимодействии с другими продуктами или системами при наличии информации только об интерфейсах этих программ или систем);
- унификация инструментов и средств разработки;
- согласованность ИТ-словаря;
- эргономичность дизайна пользовательских интерфейсов.

По мнению Роба Ван Краненбурга, основателя Европейского совета по Интернет вещей, IoT можно условно разделить на 4 уровня:

1. Первый уровень связан с идентификацией каждого объекта.
2. Второй уровень предоставляет с сервисом по обслуживанию потребностей потребителя.
3. Третий уровень связан с урбанизацией городской жизни.
4. Четвертый уровень – это сенсорная планета.

С 2017 года на базе подкомитета ИСО/МЭК СТК 1/ПК 41 «Интернет вещей и смежные технологии» реализуется план по разработке стандартов интернета вещей. В деятельности подкомитета принимает 25 стран – полноценных участников и 9 наблюдателей [30].

В РФ Статьи 8 и 9 Федерального закона «О стандартизации в Российской Федерации» от 29.06.2015 г. N 162-ФЗ разделяют зоны ответственности Федеральных Органов Исполнительной Власти (ФОИВ), осуществляющих функции по выработке государственной политики и

нормативно-правовому регулированию в сфере стандартизации, и ФОИВ в сфере стандартизации.

6.3. ВЫБОР ПРОГРАММНОЙ ПЛАТФОРМЫ ИНТЕРНЕТА ВЕЩЕЙ

Следующим этапом разработки является выбор платформы интернета вещей и разработка программной части. Целью платформ Интернета вещей является облегчение развития IoT приложений, их деятельности с помощью предоставления определённого функционала, который обычно требуется для их реализации [31].

На рынке уже имеются сотни платформ. IoT Analytics – это ведущая аналитическая компания, специализирующаяся на исследованиях Интернета вещей (IoT). Они насчитали более 450 платформ и число продолжает расти. При этом различные платформы появляются из-за различных стратегий входа различных компаний, видения развития, подходов и начальных условий.

Платформы Интернета вещей предоставляют разнообразный набор функциональных компонентов. Согласно [30] платформы для устройств Интернета вещей можно представить в виде прослойки, которая находится между IoT устройствами, расположенными на границе сети и IoT приложениями. Следовательно, нижняя часть стека включает в себя основные функции устройства, а верхние уровни стека обеспечивают дополнительными функциями для развития приложений и включения сервиса. В середине находятся функции, которые поддерживают более эффективное управление и эксплуатацию вещей.

Для того, чтобы вывести на рынок лучший продукт производители сетевого оборудования, корпоративного программного обеспечения и компаний по управлению мобильностью конкурируют между собой. Существует несколько стратегий:

1. Стратегия «органического роста» снизу-вверх (Organic bottom-up approach). Платформа начинает расти от «интернета вещей». То есть в первую очередь появляются новые устройства (обычно универсальные датчики), а потом начинается расширение и улучшение функционала, выстраиваются связи. Пример: платформа Ayla Networks, которая возникла при появлении контроллеров STM32F3.
2. Стратегия «сверху вниз» (Organic top-down approach). В первую очередь появляется аналитическая составляющая, затем устройства «интернет вещей» адаптируются к ней. Пример: IBM IoT Foundation.
3. Стратегия «партнерства» (Partnership approach). Основанная на создании альянсов для развития продукта. Пример: альянс GE Predix.
4. «Слияния и поглощения» (M&A approach). Стратегия объединения бизнесов. Пример: копания Amazon в 2015 году купила компанию 2lemetry, в результате чего данный альянс получил название AWS IoT.
5. «Инвестиционный подход» (Investment approach). Тактические инвестиции по всей экосистеме IoT. Пример: Cisco.

Компания IoT Analytics утверждает, что в настоящее время продажа основного продукта становится побочным процессом, а основные бизнес-интересы строятся вокруг «торговли данными» [13]. Также IoT Analytics прогнозирует, что в скором времени будут созданы абсолютно новые отрасли, то есть «множество устройств, подключенных к интернету» [28].

В результате исследования IoT Analytics выделили восемь компонентов полноценной IoT-платформы:

1. Связь и нормализация (Connectivity & normalization): сведение различных протоколов и форматов данных в один "программный" интерфейс, гарантируя точную передачу данных и взаимодействие со всеми устройствами.

2. Управление устройствами (Device management): обеспечение правильной работы подключенных "интернет-вещей", бесперебойную работу.
3. База данных (Database): наличие масштабируемого хранилища данных.
4. Обработка и управление действиями (Processing & action management): данные, полученные от устройств «интернета вещей» в конечном итоге влияют на события в реальности, следовательно, платформа должна уметь строить процессы.
5. Аналитика (Analytics): анализ данных является всенепременным требованием к платформе.
6. Визуализация (Visualization): возможность построения графиков, моделей и удобный интерфейс.
7. Дополнительные инструменты (Additional tools): набор инструментов, который позволяет разработчикам IoT создавать прототипы, тестировать и пробовать различные системы.
8. Внешние интерфейсы (External interfaces): интеграция с помощью платформы - одна из главных возможностей. IoT-платформа должна иметь интерфейсы прикладного программирования (API), комплекты разработки программного обеспечения (SDK) и шлюзы.

Поскольку у систем Интернета вещей несовместимые друг с другом интерфейсы программирования, и стандартных общеотраслевых API пока нет. В отличие от разработки мобильных и веб-приложений, где конвергенция уже произошла, API для Интернета вещей сегодня обычно предлагаются конкретными производителями для конкретных устройств. Главным критерием при выборе платформы – открытый исходный код. Также системы Интернета вещей зависимы от облака, где собираются практически все данные. Большая часть вычислений и обработки тоже происходит в облаке.

При разработке могут возникать следующие трудности:

– программирование одновременно множества устройств;
– постоянно работающая система, реагирующая на любые воздействия;
– разнообразие;
– необходимость создания отказоустойчивого, высокозащищенного кода.

Приложение Интернета вещей работает непрерывно, реагируя на изменения. Вычисления запускаются на основании показаний датчиков и приводят к выполнению действий. Для написания ПО Интернета вещей используются те же языки и инструменты программирования, что и для мобильной и веб-разработки. В частности, создание кода для популярных сегодня плат Arduino, Espruino, Tessel, Intel Edison и Galileo осуществляется на Си, C#, Java, JavaScript и Python.

Далее с целью выбора платформы для внедрения программного модуля необходимо сравнить популярные платформы Интернета вещей с точки зрения обычного пользователя и разработчика с помощью определенных критериев. Был произведен отбор критериев таким образом, чтобы они оценивали возможность решения поставленной задачи, то есть выбор платформы для разработки модуля программируемых сценариев взаимодействия устройств в рамках платформы Интернета вещей. На первоначальном этапе были выделены критерии, которые бы позволили начать работу сразу.

1. Возможность интеграции модуля в архитектуру платформы.
2. Наличие графического интерфейса конфигуратора:
– возможность добавить/настроить устройство;
– возможность добавить/настроить отображение информации;
– возможность добавить/настроить правило обработки.
3. Возможность хостинга модуля вместе с платформой.
4. Open-Source.
5. Возможность программной интеграции.
6. Стоимость использования платформы

7. Наличие облачной инфраструктуры для соединения Интернет вещей.

Критерий возможности интеграции модуля в архитектуру платформы является одним из самых важных при выборе платформы, поскольку если возможность встраивания низкая или совершенно отсутствует, разрабатываемый модуль будет непригодным при использовании с конкретной платформой. В зависимости от архитектуры платформы удобство встраивания дополнительных модулей может различаться, особенно если эти модули тесно взаимодействуют с существующими модулями платформы. Разрабатываемый модуль полностью возьмет на себе функционал взаимодействия устройств, необходимо будет слушать события с устройств и посылать сообщения.

Blynk IoT platform, Cayenne, Thinger.io предоставляют прямой доступ к своим устройствам через широкий API, их архитектуры подразумевает шаблон подписки и публикации сообщений. В Things Board для встраивания модуля в такую архитектуру потребуется написание отдельного модуля, специализированного под такую архитектуру, который будет неприменим для других платформ.

Наличие графического интерфейса конфигуратора является важным критерием, поскольку данное условие сократит время выведения IoT продукта на рынок.

Следующим критерием, выбранным для анализа, является возможность хостинга модуля вместе с платформой. Поскольку поднять сервер не является трудной задачей, то единственным плюсом является уменьшение задержки отправки и получения сообщений, что может быть критично для желания достижения работы системы в реальном времени.

Доступность библиотек и SDK для разработки взаимодействия устройств является довольно значимым, так как доступность технологий для разработки даст свободу выбора технологии и языка при программировании

модуля. В данном контексте все платформы предоставляют тот или иной доступ к управлению устройствами.

Критерий стоимость использования платформы не является критическим на данном этапе, но впоследствии, при развитии модуля и количества устройств, подключенных к нему, может сыграть важную роль.

Последним, но не менее важным критерием, можно выделить безопасность и защищенность протокола. Все платформы предоставляют современные меры защиты информации.

В таблице 15 представлены критерии и веса сравнительного анализа платформ. Выбранные веса для критериев обозначают важность и необходимость каждого критерия при решении задачи разработки модуля. Сумма весов всех критериев будет равняться единице.

Таблица 15 – Критерии и веса сравнительного анализа интеграционных платформ

Наличие конструктора веб и мобильных приложений	**Возможность интеграции в платформу**	**Стоимость**
0,4	0,4	0,2

В результате была выбрана Blynk IoT platform, имеющая следующие преимущества:

- быстрое создание прототипа до производства с контролируемым бюджетом;
- библиотека Blynk может подключать любое оборудование через Ethernet, WiFi или GSM, 2G, 3G, LTE и т.д.
- обширный API аппаратного облачного приложения;
- Blynk Cloud является открытым исходным кодом. Он может работать в выбранной среде, локально, на частном бизнес-сервере или на компьютере.

В таблице 16 представлены результаты сравнительного анализа выбранных платформ.

Таблица 16 – Результаты сравнительного анализа интеграционных платформ

Платформа	**Наличие конструктора веб и мобильных приложений**	**Возможность интеграции в платформу**	**Стоимость**	**Результат**
Blynk IoT platform	0,4	0,4	0,2	1,0
Cayenne	0,4	0,4	0,1	0,9
Thinger.io	0,4	0,2	0	0,6
ThingsBoard	0	0,2	0,2	0,4

Особенностью построения приложения в данной платформе является то, что пользователю предоставляется лимитированный набор заряда батареи. Он тратится на использование элементов интерфейса для создания своего приложения. Как только энергия закончится, добавить новые элементы в приложение будет уже нельзя. При удалении, энергия, затраченная на установку, возвращается обратно в полном объеме. Данной услугой разработчики позволяют новому пользователю оценить возможности Blynk и создать как минимум одно более-менее функциональное приложение. Кроме всего прочего Blynk состоит из сервера и приложения для iOS.

Если существует собственный сервер Blynk, то есть возможность создания связанных объектов, разработанных с помощью Blynk, из Интернета, не подвергая свой материал для взаимодействия через проект Homebridge.

Для этого необходимо Apple TV (минимум 4-го поколения) или iPad (минимум iOS 8). Другим преимуществом предоставления объектов Blynk HomeKit является гораздо лучшая интеграция с iOS.

В качестве среды разработки была использована Интегрированная среда разработки Arduino. В приложении Д представлен исходный код.

6.4. АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ

Датчик влажности и температуры DHT22 обменивается данными с ESP32 по протоколу 1-Wire. Это протокол передачи данных в обе стороны по одному проводу. Он был разработан фирмой Dallas Semiconductor. Преимущество 1-Wire – это легкость монтажа, возможность распознавания устройства при подключении к функционирующей сети, большому числу устройств в сети и т.д.

Устройства подключаются к шине по схеме с открытым стоком и подтягивающим резистором. Главная особенность шины 1-Wire в том, что она использует лишь два провода, один – сигнальный, другой – для заземления устройств. Большинство устройств 1-Wire поддерживают две скорости передачи данных: стандартную – около 15 кбит/с и повышенную (overdrive) – около 111 кбит/с.

Режим передачи данных по шине 1-Wire – полудуплексный: мастер и ведомые устройства передают данные по очереди. Каждая транзакция через интерфейс 1-Wire начинается с того, что мастер передает импульс Reset. Для этого он переводит напряжение в шине на низкий уровень и удерживает его в этом состоянии в течение 480 мкс. отпускает шину, и подтягивающий резистор возвращает напряжение к высокому логическому уровню. Все ведомые устройства, обнаружив сигнал Reset и дождавшись его окончания, передают свой сигнал – Presence. Он представляет собой сигнал низкого уровня длительностью 100–200 мс. Устройство может генерировать сигнал Presence и без импульса Reset – например, таким способом оно сообщает о себе при подключении к шине.

Когда ESP32 посылает сигнал запуска, то AM2302 переходит из режима ожидания в режим работы. В случае, когда ESP32 заканчивает

отправку, то сигнал запуска AM2302 отправит ответный сигнал 40-битных данных, которые отражают относительную влажность и температуру в MCU. Без стартового сигнала от MCU AM2302 не подаст ответный сигнал на MCU. AM2302 перейдет в режим ожидания, когда сбор данных завершен, если он не получает сигнал запуска от MCU снова. У каждого устройства 1-Wire есть 64-разрядный идентификатор (ID). Он состоит из 8-разрядного кода семейства, который идентифицирует тип устройства и поддерживаемые им функции, 48-разрядного серийного номера и 8-битного поля кода циклического избыточного контроля (CRC-8). ID вводится при изготовлении устройства и хранится в ПЗУ.

После передачи импульса Presence устройство 1-Wire готово к приему команд. Весь информационный обмен в шине происходит под управлением мастера. Для передачи каждого бита выделяется специальный временной промежуток (таймслот) длительностью порядка 80 мкс. В начале каждого таймслота мастер переводит линию на нулевой уровень. Если далее мастер хочет передать 0, он удерживает напряжение на низком уровне как минимум 60 мкс. На рисунке 24 продемонстрирован данный процесс.

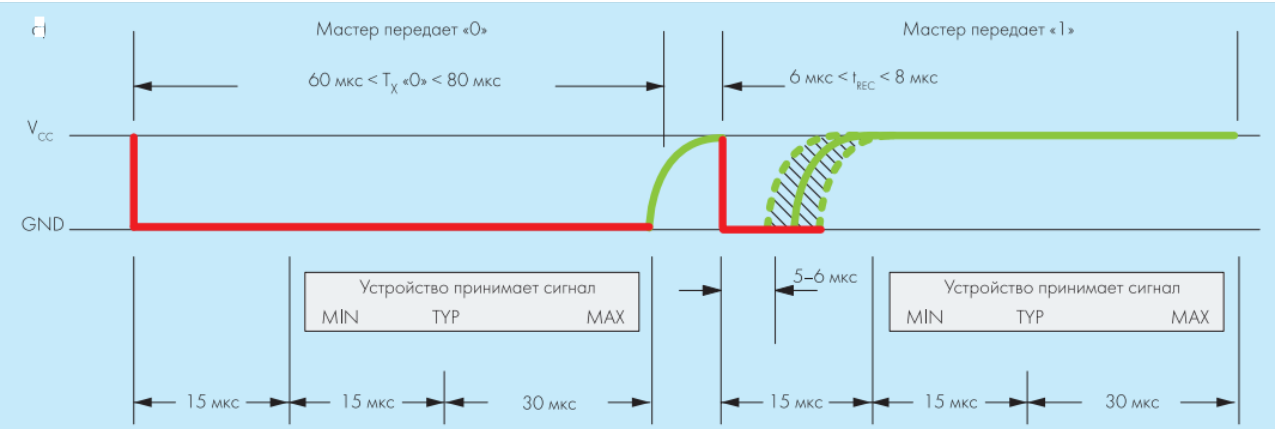

Рисунок 24 – Передача информационных битов по шине 1-Wire: мастер передает сигналы

Если мастер ожидает данные от ведомых устройств, он также обозначает начало тайм-слота, обнуляя линию на 5–6 мкс, после чего

перестает удерживать низкое напряжение и в течение короткого времени слушает линию. На рисунке 25 продемонстрирован данный процесс.

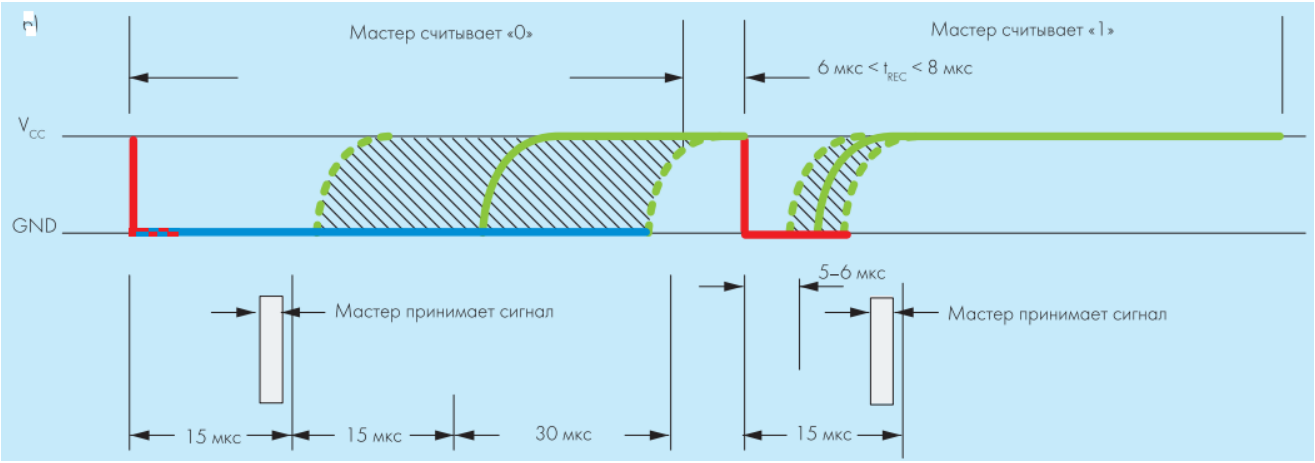

Рисунок 25 – Передача информационных битов по шине 1-Wire: мастер считывает сигналы

Для датчика влажности и температуры расчет данных можно представить в следующем виде: данные = 16-битные данные относительной влажности + 16-битные данные температуры + 8-битная контрольная сумма. Данный процесс происходит следующим образом, например, ESP32 получил 40-битные данные от AM2302: 0000 0010 1000 1100 0000 0001 0101 1111 1110 1110. Их можно разбить: 0000 0010 1000 1100 - 16 битов данных влажности, 0000 0001 0101 1111 - 16 битов температуры, 1110 1110 - контрольная сумма данных. Далее происходит конвертация 16-битных данные влажности из двоичной системы в десятичную систему: 0000 0010 1000 1100 → 652. Итого RH=652/10=65.2%RH. Такая же ситуация происходит с данными о температуре: 0000 0001 0101 1111 → 351. Итого T=351/10=35.1℃. Когда максимальный бит температуры равен 1, это означает, что температура ниже 0 градусов Цельсия.

Датчик углекислого газа MH-Z19 обменивается данными с ESP32 по протоколу UART с надстройкой Modbus. MODBUS - это протокол обмена сообщениями прикладного уровня, расположенный на уровне 7 модели OSI.

Он обеспечивает связь клиент/сервер между устройствами, подключенными к различным шинами или сетям.

Сообщение Modbus RTU состоит из адреса устройства SlaveID, кода функции, специальных данных в зависимости от кода функции и CRC контрольной суммы. На рисунке 26 показан формат сообщения.

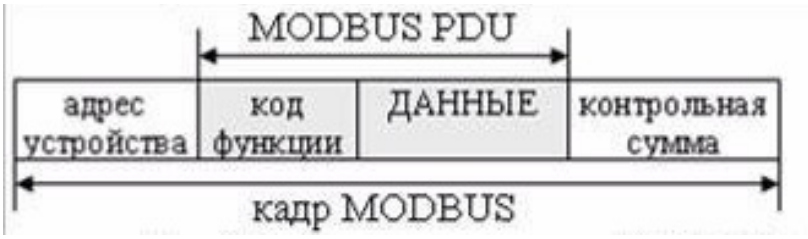

Рисунок 26 – Тип формат кадра Modbus

SlaveID – это адрес устройства, может принимать значение от 0 до 247, адреса с 248 до 255 зарезервированы. Данные в модуле хранятся в 4 таблицах.

Transaction Identifier: 2 байта устанавливаются Master, чтобы однозначно идентифицировать каждый запрос. Может быть любыми. Эти байты повторятся устройством Slave в ответе, поскольку ответы устройства Slave не всегда могут быть получены в том же порядке, что и запросы. Protocol Identifier: 2 байта устанавливаются Master, всегда будут = 00 00, что соответствует протоколу Modbus. Length: 2 байта устанавливаются Master, идентифицирующие число байтов в сообщении, которые следуют далее. Считается от Unit Identifier до конца сообщения. Unit Identifier: 1 байт устанавливается Master. Повторяется устройством Slave для однозначной идентификации устройства Slave.

Команда 03 (0x03) Read Holding Registers (16 bits read / write registers) используется для чтения содержания углекислого газа. CRC - это код циклического контроля. Контрольная сумма – это циклический проверочный код на основе полинома A001h. Передающее устройство формирует контрольную сумму для всех байт передаваемого сообщения. Принимающее устройство аналогичным образом формирует контрольную сумму для всех

байт принятого сообщения и сравнивает ее с контрольной суммой, принятой от передающего устройства. При несовпадении сформированной и принятой контрольных сумм генерируется сообщение об ошибке. Поле контрольной суммы занимает 2 байта. Контрольная сумма в сообщении передается младшим байтом вперед. Формирование контрольной суммы продемонстрировано на рисунке 27.

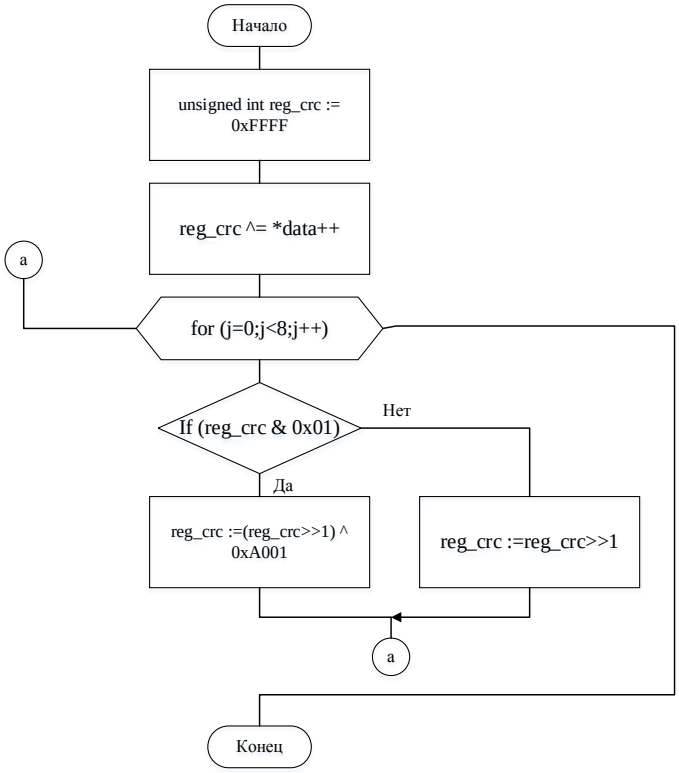

Рисунок 27 – Алгоритм подсчета контрольной суммы

С помощью циклического избыточного кода (Cyclic Redundancy Check, CRC) основан метод обнаружения ошибок. Циклические избыточные коды также называют полиномиальными кодами, так как при их вычислении битовая строка рассматривается как многочлен (полином), коэффициенты которого равны 0 или 1, и операции с этой битовой строкой можно интерпретировать как операции деления и умножения многочленов. CRC некоторой последовательности вычисляется на основании другой (исходной) битовой последовательности. Главная особенность значения CRC состоит в том, что оно однозначно идентифицирует исходную битовую последовательность.

7. ЗАКЛЮЧЕНИЕ

При разработке дипломной работы был проведен анализ существующих систем, способных с разной долей успешности выполнять данную задачу. Были выявлены недостатки и принято решение о разработке нового устройства. Проанализировав полученные данные, был составлен полный список требований ко всем частям разрабатываемого комплекса.

Исходя из списка требований, была разработана архитектура будущего комплекса и изготовлен опытный образец устройства. По итогам выполнения выпускной квалификационной работы было сделано следующее:

1. Поставлена задача.
2. Проведен анализ существующих устройств.
3. Сделан вывод о необходимости создания нового устройства.
4. Разработаны требования.
5. Осуществлено проектирование архитектуры.
6. Произведен подбор компонентов.
7. Составлены схемы.
8. Выбраны средства программной разработки.
9. Осуществлено написание программной части.
10. Произведена проверка на соответствие предъявленным требованиям.

В пояснительной записке к дипломному проекту отображены все основные этапы проектирования системы, разработка ее модулей и настройка для дальнейшего использования системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Степановских, А.С. Экология: Учебник для вузов. / А.С. Степановских // Издательство Юнти-Дана, 2001. – 703 с.
2. Звонков, А.Л. Пока едет скорая. Рассказы, которые могут спасти вашу жизнь. - Эксмо-Пресс, 2015. – 368 с.
3. ASHRAE 62–1999. Ventilation for Acceptable Indoor Air Quality. - American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., 2001. -147 p.
4. ГОСТ Р ЕН 13779-2007. Вентиляция в нежилых зданиях. Технические требования к вентиляции и кондиционированию. – М: Федеральное агентство по техническому регулированию и метрологии, 2007. – 45 с.
5. ГОСТ 12.1.005–75. ССБТ. Воздух рабочей зоны. Общие санитарно-гигиенические требования. – М.: Изд-во стандартов, 1975. – 55 с.
6. Гурина, И.В. Кто ответит за духоту в помещении. / И.В. Гурина // Сантехника, отопление, кондиционирование – 2010. – Вып. 3. - № 3 (99). - С. 84-87.
7. Immunol, J. Hypercapnia Inhibits Autophagy and Bacterial Killing in Human Macrophages by Increasing Expression of Bcl-2 and Bcl-xL. / J. Immunol // The journal of immunology. – 2015. – V 4. - № 194 (11). – P. 5388-5396.
8. ГОСТ 30494-2011. Здания жилые и общественные. - М.: Изд-во стандартов, 2011. – 12 с.
9. Пенг, Дж. Моделирование воздействия концентрации CO2 на изменения климата / Дж. Пенг, Л. Дан // Экологическое моделирование. – 2015. - Том 304. - С. 69-83.
10. ГН 2.2.5.2100-06. Предельно допустимые концентрации (ПДК) вредных веществ в воздухе рабочей зоны (дополнение N 2 к ГН

2.2.5.1313-03 Предельно допустимые концентрации (ПДК) вредных веществ в воздухе рабочей зоны). – 2006.

11. Умный дом. - http://www.dom-electro.ru/. Дата обращения: 15.02.2019.
12. Шепелев, М.С. Критерии выбора микроконтроллеров для разработки модулей модульных устройств / М.С. Шепелев // Молодой ученый. — 2016. — №9. — С. 88-91.
13. 5 Things To Know About The IoT Platform Ecosystem. - https://iot-analytics.com/5-things-know-about-iot-platform/. Дата обращения: 25.02.2019.
14. Чип и Дип. - https://www.chipdip.ru/product/sip-kitnxb001. Дата обращения: 13.03.2019.
15. Амперка. Wi-Fi модуль ESP8266. - http://amperka.ru/product/esp8266-wifi-module. Дата обращения: 15.03.2019.
16. AliExpress. - https://ru.aliexpress.com/. Дата обращения: 17.03.2019.
17. Banggood. - https://www.banggood.com/. Дата обращения: 17.03.2019.
18. Gainta. - http://gainta.ru/. Дата обращения: 12.03.2019
19. ГОСТ 21931-76. Припои оловянно-свинцовые в изделиях. Технические условия. – М.: Издательство стандартов, 1991. – 11 с.
20. AM2302. (Datasheet). - https://cdn-shop.adafruit.com/ /AM2302.pdf. Дата обращения: 01.04.2019.
21. MH-Z19B. (Datasheet). - https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf. Дата обращения: 03.04.2019.
22. M5Stack GRAY Kit. - https://docs.m5stack.com/#/en/core/ description. Дата обращения: 06.04.2019.
23. ГОСТ 2.702-2011. Межгосударственный стандарт. Единая система конструкторской документации. Правила выполнения электрических схем. – М.: Стандартинформ, 2011. – 21 с

24. Руководство для учащихся по изучению программного обеспечения SolidWorks. - https://www.solidworks.com.pdf. Дата обращения: 09.04.2019.
25. International research project report within the programme “Indoor air quality in European schools: Preventing and reducing respiratory diseases”. - http://search.rec.org/search1/doc/SEARCH%20publication_EN_final.pdf. Дата обращения: 09.04.2019.
26. Сиагри, Р. Основы архитектуры «Интернета вещей» / Р. Сиагри // CONTROL ENGINEERING РОССИЯ – 2017. - Вып. 4. - №5 (71). – С. 88-91.
27. Москаленко, Т. А. Архитектуры промышленного интернета вещей / Т. А. Москаленко, Р. В. Киричек, А. С. Бородин // Информационные технологии и телекоммуникации. - 2017. - Том 5. - № 4. С. 49–56.
28. Guth, J. A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences / J. Guth, U. Breitenbücher, M. Falkenthal1 // Institute of Architecture of Application Systems. – 2018. – V. 4. – P. 33 – 43.
29. Интернет вещей. - http://pts-russia.com/download/J4036.pdf. Дата обращения: 15.04.2019.
30. Дежина, И. Г. Перспективные рынки и технологии интернета вещей / И. Г.Дежина, А. К. Пономарев, М. А. Пукальчик. – М: ООО «Лайм», 2019. - 272 с
31. Рекомендации МСЭ - Т Y.2060 «Обзор интернета вещей». – https://www.itu.int/itu-t/recommendations/rec. aspx?rec=11559&lang=ru. Дата обращения: 16.04.2019.
32. Концепция построения и развития узкополосных беспроводных сетей связи «Интернета вещей» на территории Российской Федерации: приказ Министерства цифрового развития, связи и массовых

коммуникаций Российской федерации от 29.03.2019 г. № 113 // Вестник России. - 2019. - № 3. - С. 3–108.

33. Selonen, P. IoT Cloud Environment for Enabling the Programmable World. Proc. 42nd Euromicro Conf. Software Eng. And Advanced Applications (SEAA 16). — 2016. — P. 250–257.

ПРИЛОЖЕНИЕ

ПРИЛОЖЕНИЕ А

Схема электрическая принципиальная.

Чертеж на А3

Чертеж на А4

ПРИЛОЖЕНИЕ Б

Перечень элементов к схеме электрической принципиальной.

Спецификация на А4

ПРИЛОЖЕНИЕ В

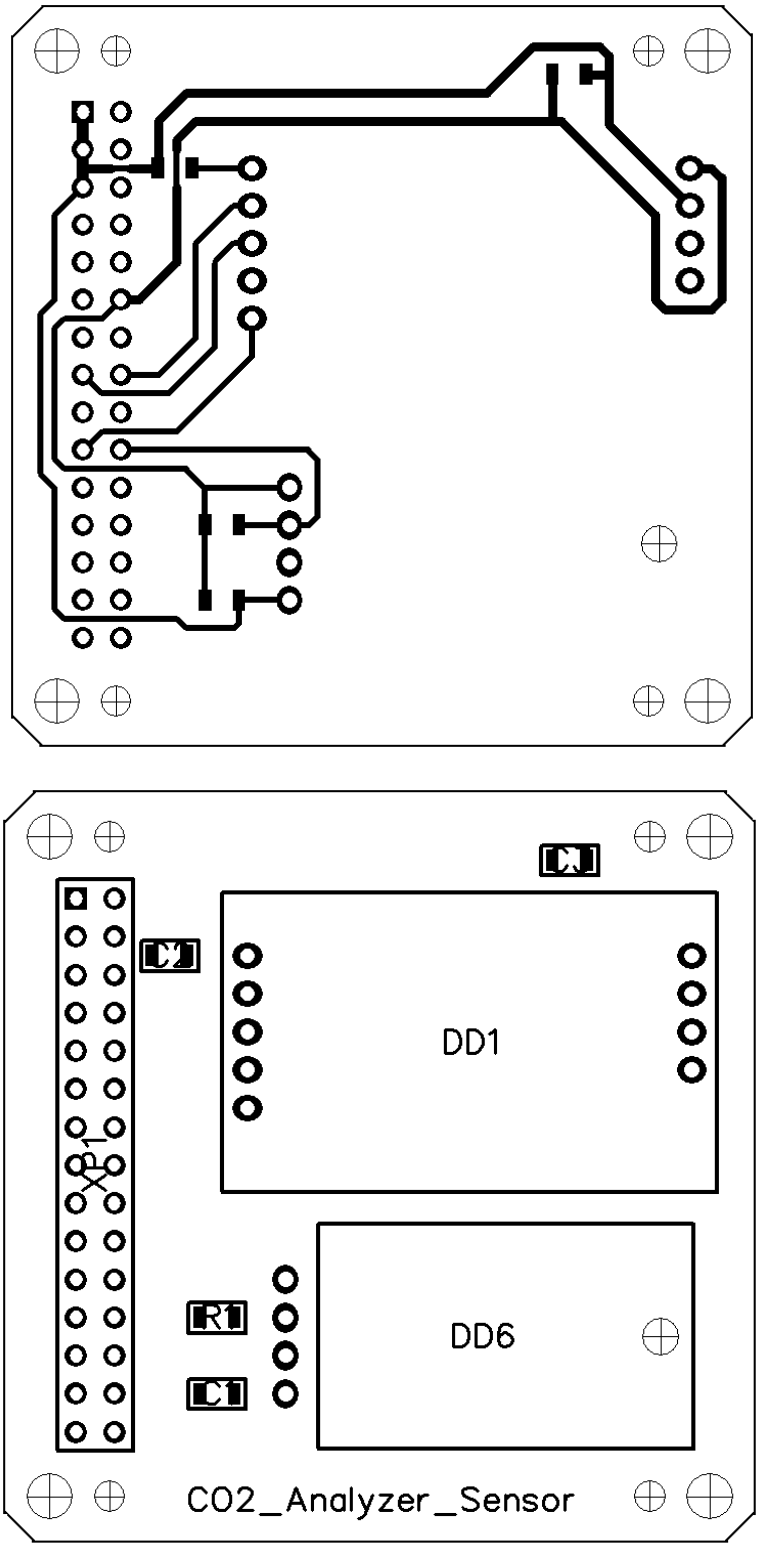

Рисунок В.1 – Печатная плата

ПРИЛОЖЕНИЕ Г

Чертеж корпуса.

Чертеж корпуса на А4

ПРИЛОЖЕНИЕ Д

Листинг Д.1 – Исходный код главного модуля.

```
#include "DHTesp.h"
void DHTesp::setup(uint8_t pin, DHT_MODEL_t model)
{
  DHTesp::pin = pin;
  DHTesp::model = model;
  DHTesp::resetTimer(); // Make sure we do read the sensor in the next readSensor()
  if ( model == AUTO_DETECT) {
    DHTesp::model = DHT22;
    readSensor();
    if ( error == ERROR_TIMEOUT ) {
      DHTesp::model = DHT11;
      // Warning: in case we auto detect a DHT11, you should wait at least 1000 msec
      // before your first read request. Otherwise you will get a time out error.
    }
  }
  //Set default comfort profile.
  m_comfort.m_tooHot_m = -0.095;
  m_comfort.m_tooHot_b = 32.85;
  //Too humid line BC
  m_comfort.m_tooHumid_m =  -56.5;
  m_comfort.m_tooHumid_b = 3981.2;
  //Too cold line DC
  m_comfort.m_tooCold_m = -0.04175;
  m_comfort.m_tooHCold_b = 23.476675;
  //Too dry line AD
  m_comfort.m_tooDry_m = -77.8;
  m_comfort.m_tooDry_b = 2364;
}
void DHTesp::resetTimer()
{
  DHTesp::lastReadTime = millis() - 3000;
}
```

```
float DHTesp::getHumidity()
{
  readSensor();
  if ( error == ERROR_TIMEOUT ) { // Try a second time to read
    readSensor();
  }
  return humidity;
}
float DHTesp::getTemperature()
{
  readSensor();
  if ( error == ERROR_TIMEOUT ) { // Try a second time to read
    readSensor();
  }
  return temperature;
}

TempAndHumidity DHTesp::getTempAndHumidity()
{
  readSensor();
  if ( error == ERROR_TIMEOUT ) { // Try a second time to read
    readSensor();
  }
  values.temperature = temperature;
  values.humidity = humidity;
  return values;
}
#ifndef OPTIMIZE_SRAM_SIZE
const char* DHTesp::getStatusString()
{
  switch ( error ) {
    case DHTesp::ERROR_TIMEOUT:
      return "TIMEOUT";
    case DHTesp::ERROR_CHECKSUM:
      return "CHECKSUM";
```

```
    default:
      return "OK";
  }
}
// SRAM by using the following method:
prog_char P_OK[]       PROGMEM = "OK";
prog_char P_TIMEOUT[]  PROGMEM = "TIMEOUT";
prog_char P_CHECKSUM[] PROGMEM = "CHECKSUM";
const char *DHTesp::getStatusString() {
  prog_char *c;
  switch ( error ) {
    case DHTesp::ERROR_CHECKSUM:
      c = P_CHECKSUM; break;
    case DHTesp::ERROR_TIMEOUT:
      c = P_TIMEOUT; break;
    default:
      c = P_OK; break;
  }
  static char buffer[9];
  strcpy_P(buffer, c);
  return buffer;
}
void DHTesp::readSensor()
{
  // Make sure we don't poll the sensor too often
  // - Max sample rate DHT11 is 1 Hz   (duty cicle 1000 ms)
  // - Max sample rate DHT22 is 0.5 Hz (duty cicle 2000 ms)
  unsigned long startTime = millis();
  if ( (unsigned long)(startTime - lastReadTime) < (model == DHT11 ? 999L : 1999L) ) {
    return;
  }
  lastReadTime = startTime;
  temperature = NAN;
  humidity = NAN;
  uint16_t rawHumidity = 0;
  uint16_t rawTemperature = 0;
```

```
  uint16_t data = 0;
 // Request sample
  digitalWrite(pin, LOW); // Send start signal
  pinMode(pin, OUTPUT);
  if ( model == DHT11 ) {
    delay(18);
  }
  else {
    // This will fail for a DHT11 - that's how we can detect such a device
    delay(2);
  }
  pinMode(pin, INPUT);
  digitalWrite(pin, HIGH); // Switch bus to receive data
  // We're going to read 83 edges:
  // - First a FALLING, RISING, and FALLING edge for the start bit
  // - Then 40 bits: RISING and then a FALLING edge per bit
  // To keep our code simple, we accept any HIGH or LOW reading if it's max 85 usecs long
#ifdef ESP32
  // ESP32 is a multi core / multi processing chip
  // It is necessary to disable task switches during the readings
  portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
  portENTER_CRITICAL(&mux);
#else
  cli();
#endif
  for ( int8_t i = -3 ; i < 2 * 40; i++ ) {
    byte age;
    startTime = micros();
    do {
      age = (unsigned long)(micros() - startTime);
      if ( age > 90 ) {
        error = ERROR_TIMEOUT;
#ifdef ESP32
        portEXIT_CRITICAL(&mux);
#else
        sei();
```

```
#endif
      return;
     }
    }
    while ( digitalRead(pin) == (i & 1) ? HIGH : LOW );

    if ( i >= 0 && (i & 1) ) {
     // Now we are being fed our 40 bits
     data <<= 1;

     // A zero max 30 usecs, a one at least 68 usecs.
     if ( age > 30 ) {
      data |= 1; // we got a one
     }
    }
    switch ( i ) {
     case 31:
      rawHumidity = data;
      break;
     case 63:
      rawTemperature = data;
      data = 0;
      break;
    }
   }
#ifdef ESP32
  portEXIT_CRITICAL(&mux);
#else
  sei();
#endif
  // Verify checksum
  if ( (byte)(((byte)rawHumidity) + (rawHumidity >> 8) + ((byte)rawTemperature) +
(rawTemperature >> 8)) != data ) {
   error = ERROR_CHECKSUM;
   return;
  }
```

```
  // Store readings
  if ( model == DHT11 ) {
    humidity = rawHumidity >> 8;
    temperature = rawTemperature >> 8;
  }
  else {
    humidity = rawHumidity * 0.1;
    if ( rawTemperature & 0x8000 ) {
      rawTemperature = -(int16_t)(rawTemperature & 0x7FFF);
    }
    temperature = ((int16_t)rawTemperature) * 0.1;
  }
  error = ERROR_NONE;
}
//boolean isFahrenheit: True == Fahrenheit; False == Celcius
float DHTesp::computeHeatIndex(float temperature, float percentHumidity, bool isFahrenheit) {
  // Using both Rothfusz and Steadman's equations
  // http://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml
  float hi;
  if (!isFahrenheit) {
    temperature = toFahrenheit(temperature);
  }
  hi = 0.5 * (temperature + 61.0 + ((temperature - 68.0) * 1.2) + (percentHumidity * 0.094));
  if (hi > 79) {
    hi = -42.379 +
             2.04901523 * temperature +
            10.14333127 * percentHumidity +
            -0.22475541 * temperature*percentHumidity +
            -0.00683783 * pow(temperature, 2) +
            -0.05481717 * pow(percentHumidity, 2) +
             0.00122874 * pow(temperature, 2) * percentHumidity +
             0.00085282 * temperature*pow(percentHumidity, 2) +
            -0.00000199 * pow(temperature, 2) * pow(percentHumidity, 2);
    if((percentHumidity < 13) && (temperature >= 80.0) && (temperature <= 112.0))
      hi -= ((13.0 - percentHumidity) * 0.25) * sqrt((17.0 - abs(temperature - 95.0)) * 0.05882);
```

```
    else if((percentHumidity > 85.0) && (temperature >= 80.0) && (temperature <= 87.0))
      hi += ((percentHumidity - 85.0) * 0.1) * ((87.0 - temperature) * 0.2);
  }

  return isFahrenheit ? hi : toCelsius(hi);
}
//boolean isFahrenheit: True == Fahrenheit; False == Celcius
float DHTesp::computeDewPoint(float temperature, float percentHumidity, bool isFahrenheit) {
  // reference: http://wahiduddin.net/calc/density_algorithms.htm
  if (isFahrenheit) {
    temperature = toCelsius(temperature);
  }
  double A0 = 373.15 / (273.15 + (double) temperature);
  double SUM = -7.90298 * (A0 - 1);
  SUM += 5.02808 * log10(A0);
  SUM += -1.3816e-7 * (pow(10, (11.344 * (1 - 1 / A0))) - 1) ;
  SUM += 8.1328e-3 * (pow(10, (-3.49149 * (A0 - 1))) - 1) ;
  SUM += log10(1013.246);
  double VP = pow(10, SUM - 3) * (double) percentHumidity;
  double Td = log(VP / 0.61078); // temp var
  Td = (241.88 * Td) / (17.558 - Td);
  return isFahrenheit ? toFahrenheit(Td) : Td;
}
//boolean isFahrenheit: True == Fahrenheit; False == Celcius
byte DHTesp::computePerception(float temperature, float percentHumidity, bool isFahrenheit) {
  // Computing human perception from dew point
  // reference: https://en.wikipedia.org/wiki/Dew_point ==> Relationship to human comfort
  // reference: Horstmeyer, Steve (2006-08-15). "Relative Humidity....Relative to What? The Dew
Point Temperature...a better approach". Steve Horstmeyer, Meteorologist, WKRC TV, Cincinnati, Ohio,
USA. Retrieved 2009-08-20.
  if (isFahrenheit) {
    temperature = toCelsius(temperature);
  }
  float dewPoint = computeDewPoint(temperature, percentHumidity);
  if (dewPoint < 10.0f) {
    return Perception_Dry;
```

```
  } else if (dewPoint < 13.0f) {
    return Perception_VeryComfy;
  } else if (dewPoint < 16.0f) {
    return Perception_Comfy;
  } else if (dewPoint < 18.0f) {
    return Perception_Ok;
  } else if (dewPoint < 21.0f) {
    return Perception_UnComfy;
  } else if (dewPoint < 24.0f) {
    return Perception_QuiteUnComfy;
  } else if (dewPoint < 26.0f) {
    return Perception_VeryUnComfy;
  }
  // else dew >= 26.0
  return Perception_SevereUncomfy;
}
//boolean isFahrenheit: True == Fahrenheit; False == Celcius
float DHTesp::getComfortRatio(ComfortState& destComfortStatus, float temperature, float percentHumidity, bool isFahrenheit) {
  float ratio = 100; //100%
  float distance = 0;
  float kTempFactor = 3; //take into account the slope of the lines
  float kHumidFactor = 0.1; //take into account the slope of the lines
  uint8_t tempComfort = 0;
  if (isFahrenheit) {
    temperature = toCelsius(temperature);
  }
  destComfortStatus = Comfort_OK;
  distance = m_comfort.distanceTooHot(temperature, percentHumidity);
  if(distance > 0)
  {
    //update the comfort descriptor
    tempComfort += (uint8_t)Comfort_TooHot;
    //decrease the comfot ratio taking the distance into account
    ratio -= distance * kTempFactor;
  }
```

```
  distance = m_comfort.distanceTooHumid(temperature, percentHumidity);
  if(distance > 0)
  {
   //update the comfort descriptor
   tempComfort += (uint8_t)Comfort_TooHumid;
   //decrease the comfot ratio taking the distance into account
   ratio -= distance * kHumidFactor;
  }
  distance = m_comfort.distanceTooCold(temperature, percentHumidity);
  if(distance > 0)
  {
   //update the comfort descriptor
   tempComfort += (uint8_t)Comfort_TooCold;
   //decrease the comfot ratio taking the distance into account
   ratio -= distance * kTempFactor;
  }
  distance = m_comfort.distanceTooDry(temperature, percentHumidity);
  if(distance > 0)
  {
   //update the comfort descriptor
   tempComfort += (uint8_t)Comfort_TooDry;
   //decrease the comfot ratio taking the distance into account
   ratio -= distance * kHumidFactor;
  }
  destComfortStatus = (ComfortState)tempComfort;
  if(ratio < 0)
   ratio = 0;
  return ratio;
 }
 float DHTesp::computeAbsoluteHumidity(float temperature, float percentHumidity, bool
isFahrenheit) {
  // Calculate the absolute humidity in g/mBi
  //   https://carnotcycle.wordpress.com/2012/08/04/how-to-convert-relative-humidity-to-absolute-
humidity/
  if (isFahrenheit) {
     temperature = toCelsius(temperature);
```

```
  }
  float absHumidity;
  float absTemperature;
  absTemperature = temperature + 273.15;

  absHumidity = 6.112;
  absHumidity *= exp((17.67 * temperature) / (243.5 + temperature));
  absHumidity *= percentHumidity;
  absHumidity *= 2.1674;
  absHumidity /= absTemperature;
  return absHumidity;
}
  DHT Temperature & Humidity Sensor library for Arduino & ESP32.
#ifndef dhtesp_h
#define dhtesp_h
#if ARDUINO < 100
  #include <WProgram.h>
#else
  #include <Arduino.h>
#endif
// Reference: http://epb.apogee.net/res/refcomf.asp (References invalid)
enum ComfortState {
  Comfort_OK = 0,
  Comfort_TooHot = 1,
  Comfort_TooCold = 2,
  Comfort_TooDry = 4,
  Comfort_TooHumid = 8,
  Comfort_HotAndHumid = 9,
  Comfort_HotAndDry = 5,
  Comfort_ColdAndHumid = 10,
  Comfort_ColdAndDry = 6
};
// References https://en.wikipedia.org/wiki/Dew_point ==> Relationship to human comfort
enum PerceptionState {
  Perception_Dry = 0,
  Perception_VeryComfy = 1,
```

```
Perception_Comfy = 2,
 Perception_Ok = 3,
 Perception_UnComfy = 4,
 Perception_QuiteUnComfy = 5,
 Perception_VeryUnComfy = 6,
 Perception_SevereUncomfy = 7
};

struct TempAndHumidity {
 float temperature;
 float humidity;
};
struct ComfortProfile
{
 //Represent the 4 line equations:
 //dry, humid, hot, cold, using the y = mx + b formula
 float m_tooHot_m, m_tooHot_b;
 float m_tooCold_m, m_tooHCold_b;
 float m_tooDry_m, m_tooDry_b;
 float m_tooHumid_m, m_tooHumid_b;
 inline bool isTooHot(float temp, float humidity) {return (temp > (humidity * m_tooHot_m + m_tooHot_b));}
 inline bool isTooHumid(float temp, float humidity) {return (temp > (humidity * m_tooHumid_m + m_tooHumid_b));}
 inline bool isTooCold(float temp, float humidity) {return (temp < (humidity * m_tooCold_m + m_tooHCold_b));}
 inline bool isTooDry(float temp, float humidity) {return (temp < (humidity * m_tooDry_m + m_tooDry_b));}
 inline float distanceTooHot(float temp, float humidity) {return temp - (humidity * m_tooHot_m + m_tooHot_b);}
 inline float distanceTooHumid(float temp, float humidity) {return temp - (humidity * m_tooHumid_m + m_tooHumid_b);}
 inline float distanceTooCold(float temp, float humidity) {return (humidity * m_tooCold_m + m_tooHCold_b) - temp;}
 inline float distanceTooDry(float temp, float humidity) {return (humidity * m_tooDry_m + m_tooDry_b) - temp;}
```

```
};
class DHTesp
{
public:
  typedef enum {
    AUTO_DETECT,
    DHT11,
    DHT22,
    AM2302,  // Packaged DHT22
    RHT03    // Equivalent to DHT22
  }
  DHT_MODEL_t;
  typedef enum {
    ERROR_NONE = 0,
    ERROR_TIMEOUT,
    ERROR_CHECKSUM
  }
  DHT_ERROR_t;

  TempAndHumidity values;
  // setup(dhtPin) is deprecated, auto detection is not working well on ESP32. Use setup(dhtPin, DHTesp::DHT11) instead!
  void setup(uint8_t dhtPin) __attribute__((deprecated));
  void setup(uint8_t pin, DHT_MODEL_t model=AUTO_DETECT);
  void resetTimer();
  float getTemperature();
  float getHumidity();
  TempAndHumidity getTempAndHumidity();
  DHT_ERROR_t getStatus() { return error; };
  const char* getStatusString();
  DHT_MODEL_t getModel() { return model; }
  int getMinimumSamplingPeriod() { return model == DHT11 ? 1000 : 2000; }
  int8_t getNumberOfDecimalsTemperature() { return model == DHT11 ? 0 : 1; };
  int8_t getLowerBoundTemperature() { return model == DHT11 ? 0 : -40; };
  int8_t getUpperBoundTemperature() { return model == DHT11 ? 50 : 125; };
  int8_t getNumberOfDecimalsHumidity() { return 0; };
```

```
  int8_t getLowerBoundHumidity() { return model == DHT11 ? 20 : 0; };
  int8_t getUpperBoundHumidity() { return model == DHT11 ? 90 : 100; };
  static float toFahrenheit(float fromCelcius) { return 1.8 * fromCelcius + 32.0; };
  static float toCelsius(float fromFahrenheit) { return (fromFahrenheit - 32.0) / 1.8; };
  float computeHeatIndex(float temperature, float percentHumidity, bool isFahrenheit=false);
  float computeDewPoint(float temperature, float percentHumidity, bool isFahrenheit=false);
  float getComfortRatio(ComfortState& destComfStatus, float temperature, float
percentHumidity, bool isFahrenheit=false);
  ComfortProfile getComfortProfile() {return m_comfort;}
  void setComfortProfile(ComfortProfile& c) {m_comfort = c;}
  inline bool isTooHot(float temp, float humidity) {return m_comfort.isTooHot(temp, humidity);}
  inline bool isTooHumid(float temp, float humidity) {return m_comfort.isTooHumid(temp,
humidity);}
  inline bool isTooCold(float temp, float humidity) {return m_comfort.isTooCold(temp,
humidity);}
  inline bool isTooDry(float temp, float humidity) {return m_comfort.isTooDry(temp, humidity);}
  byte computePerception(float temperature, float percentHumidity, bool isFahrenheit=false);
  float computeAbsoluteHumidity(float temperature, float percentHumidity, bool
isFahrenheit=false);
protected:
  void readSensor();
  float temperature;
  float humidity;
  uint8_t pin;
private:
  DHT_MODEL_t model;
  DHT_ERROR_t error;
  unsigned long lastReadTime;
  ComfortProfile m_comfort;
};
#include <GraphMenu.h>
#include <M5Stack.h>
#include <Preferences.h>
#include <Global.h>
#include <WebSetting.h>
#include <time.h>
```

```
#include <WiFi.h>
// Use these when printing or drawing text in GLCD and high rendering speed fonts
#define GFXFF 1
#define GLCD 0
#define FONT2 2
#define FONT4 4
#define FONT6 6
#define FONT7 7
#define FONT8 8

#define GM_MENU_POINTER_VERTICAL (220)
#define GM_MENU_POINTER_STEP (22)
#define GM_MENU_POINTER_RADIUS (5)
NTPClient *GraphMenu::_timeClient;
MHZ19_Modbus *GraphMenu::_MHZ19;
DHTesp *GraphMenu::_DHT22;
static bool GM_DrawEntry_WasChange = true;
static uint32_t GM_DrawEntry_LastDraw = millis();
static int8_t GM_MenuCurrent = -1;
static int8_t GM_MenuCount = 0;
static bool GM_NeedToStartWebSetting = false;
uint16_t color565(uint8_t r, uint8_t g, uint8_t b)
{
    return ((r & 0xF8) << 8) | ((g & 0xFC) << 3) | (b >> 3);
}

void GraphMenu::Start()
{
    _currentDrawEntry = NULL;
    _currentDrawEntryIterator = _drawFunctionList.begin();
    _timeClient = NULL;
    _MHZ19 = NULL;
    _DHT22 = NULL;
    srand(time(NULL));
}
void GraphMenu::Process()
```

```
{
  if (M5.BtnA.wasPressed())
  { // turn to left if posible
    if ((!_drawFunctionList.empty()) && (_currentDrawEntryIterator != _drawFunctionList.begin()))
    {
      _currentDrawEntryIterator = std::prev(_currentDrawEntryIterator);
      --GM_MenuCurrent;
      GM_DrawEntry_WasChange = true;
    }
  }

  if (M5.BtnB.pressedFor(2000))
  { // make ok if posible
    if (GM_MenuCurrent == (GM_MenuCount - 1))
      GM_NeedToStartWebSetting = true;
  }
  if (M5.BtnC.wasPressed())
  { // turn to right if posible
    if ((!_drawFunctionList.empty()) && (_currentDrawEntryIterator != _drawFunctionList.end()))
    {
      _currentDrawEntryIterator = std::next(_currentDrawEntryIterator);
      if (_currentDrawEntryIterator == _drawFunctionList.end())
      {
        --_currentDrawEntryIterator;
      }
      else
      {
        ++GM_MenuCurrent;
        GM_DrawEntry_WasChange = true;
      }
    }
  }
  if (!_drawFunctionList.empty())
  {
```

```
            DrawEntry_t drawEntry = (DrawEntry_t)(*_currentDrawEntryIterator);
            if (drawEntry != NULL)
            {
               drawEntry();
            }
         }
         else
         {
            M5.Lcd.printf("Menu empty\n");
         }
      }

      int8_t GraphMenu::AddEntry(DrawEntry_t entry, int8_t position)
      {
         // check for already have
         DrawFunctionListIterator_t it = std::find(_drawFunctionList.begin(), _drawFunctionList.end(),
entry);
         if (it == _drawFunctionList.end())
         { // if was not found we can add
            DrawFunctionListIterator_t it = _drawFunctionList.begin();
            for (int8_t i = 0; i < position; i++)
            {
               ++it;
            }
            if (position == -1)
            {
               it = _drawFunctionList.end();
            }
            else
            {
               if (GM_MenuCurrent >= position)
                  ++GM_MenuCurrent;
            }
            _drawFunctionList.insert(it, (void *)entry);
            ++GM_MenuCount;
            GM_DrawEntry_WasChange = true;
```

```
        if (_drawFunctionList.size() == 1)
        {
          GM_MenuCurrent = 0;
          _currentDrawEntryIterator = _drawFunctionList.begin();
        }
        return 0;
      }
      return -1;
    }
    void GraphMenu::SetNTPClient(NTPClient *timeClient)
    {
      _timeClient = timeClient;
    }
    void GraphMenu::SetMHZ19_Modbus(MHZ19_Modbus *MHZ19)
    {
      _MHZ19 = MHZ19;
    }
    void GraphMenu::SetDHTesp(DHTesp *DHT22)
    {
      _DHT22 = DHT22;
    }
    void GraphMenu::DrawEntryPointer()
    {
      if (GM_MenuCount == 0)
        return;
      for (int16_t i = 0; i < GM_MenuCount; ++i)
      {
        uint16_t color = TFT_DARKGREY;
        if (GM_MenuCurrent == i)
          color = TFT_LIGHTGREY;
        M5.Lcd.fillCircle((M5.Lcd.width() / 2) + (i - GM_MenuCount / 2) *
GM_MENU_POINTER_STEP + (GM_MENU_POINTER_STEP / 2) * (1 - GM_MenuCount % 2),
                          GM_MENU_POINTER_VERTICAL, GM_MENU_POINTER_RADIUS,
color);
      }
    }
```

```
void GraphMenu::DrawEntry_Time()
{

  bool needToDraw = GM_DrawEntry_WasChange || ((millis() - GM_DrawEntry_LastDraw) > 1000);
  if (GM_DrawEntry_WasChange == true)
  {
    M5.Lcd.setTextColor(TFT_BLACK, color565(139, 198, 247));
    M5.Lcd.fillScreen(color565(139, 198, 247)); // Clear screen
    GM_DrawEntry_WasChange = false;
    M5.Lcd.setFreeFont(&FreeMonoBold24pt7b); // Select the font

    // Set text datum to middle centre
    M5.Lcd.setTextDatum(MC_DATUM);
    DrawEntryPointer();
  }
  if ((_timeClient != NULL) && (needToDraw))
  {
    char buffer[30] = {0};
    time_t epochTime = (time_t)(_timeClient->getEpochTime());
    struct tm *tm = localtime(&epochTime);
    //  M5.Lcd.printf("%02i:%02i:%02i  %02i.%02i.%4i", tm->tm_hour, tm->tm_min, tm->tm_sec,
    //  tm->tm_mday, tm->tm_mon + 1, tm->tm_year + 1900);
    snprintf(buffer, 30, "%02i:%02i:%02i", tm->tm_hour, tm->tm_min, tm->tm_sec);
    M5.Lcd.drawString(buffer, 160, 80, GFXFF); // Print the string
    snprintf(buffer, 30, "%02i.%02i.%4i", tm->tm_mday, tm->tm_mon + 1, tm->tm_year + 1900);
    M5.Lcd.drawString(buffer, 160, 160, GFXFF); // Print the string
    GM_DrawEntry_LastDraw = millis();
  }
}
void GraphMenu::DrawEntry_TempAndHum()
{
  bool needToDraw = GM_DrawEntry_WasChange || ((millis() - GM_DrawEntry_LastDraw) > 1000);
```

```
if (GM_DrawEntry_WasChange == true)
 {
   M5.Lcd.setTextColor(TFT_BLACK, color565(232, 128, 58));
   M5.Lcd.fillScreen(color565(232, 128, 58)); // Clear screen
   GM_DrawEntry_WasChange = false;
   M5.Lcd.setFreeFont(&FreeMonoBold24pt7b); // Select the font
   // Set text datum to middle centre
   M5.Lcd.setTextDatum(MC_DATUM);
   DrawEntryPointer();
 }
 if ((_DHT22 != NULL) && (needToDraw))
 {
   char buffer[30] = {0};
   TempAndHumidity lastValues;
   lastValues = _DHT22->getTempAndHumidity();
   if (_DHT22->getStatus() == DHTesp::ERROR_NONE)
   {
     // snprintf(buffer, 30, "%2.1f C", ((float)(rand() % 150)) / 10 + 15);
     snprintf(buffer, 30, "%2.1f C", lastValues.temperature);
     M5.Lcd.drawString(buffer, 160, 80, GFXFF); // Print the string
     // degree sign draw
     M5.Lcd.drawCircle(205, 75, 6, TFT_BLACK);
     M5.Lcd.drawCircle(205, 75, 5, TFT_BLACK);
     M5.Lcd.drawCircle(205, 75, 4, TFT_BLACK);
     // snprintf(buffer, 30, "%2.1f%%", ((float)(rand() % 500)) / 10 + 10);
     snprintf(buffer, 30, "%2.1f%%", lastValues.humidity);
     M5.Lcd.drawString(buffer, 160, 160, GFXFF); // Print the string
   }
   else
   {
     snprintf(buffer, 30, "--.- C");
     M5.Lcd.drawString(buffer, 160, 80, GFXFF); // Print the string
     // degree sign draw
     M5.Lcd.drawCircle(205, 75, 6, TFT_BLACK);
     M5.Lcd.drawCircle(205, 75, 5, TFT_BLACK);
     M5.Lcd.drawCircle(205, 75, 4, TFT_BLACK);
```

```
            // snprintf(buffer, 30, "%2.1f%%", ((float)(rand() % 500)) / 10 + 10);
             snprintf(buffer, 30, "--.-%%", lastValues.humidity);
             M5.Lcd.drawString(buffer, 160, 160, GFXFF); // Print the string
          }
          GM_DrawEntry_LastDraw = millis();
        }
      }
      void GraphMenu::DrawEntry_CO2()
      {
        bool needToDraw = GM_DrawEntry_WasChange || ((millis() - GM_DrawEntry_LastDraw) >
1000);
        if (GM_DrawEntry_WasChange == true)
        {
          M5.Lcd.setTextColor(TFT_BLACK, color565(133, 242, 169));
          M5.Lcd.fillScreen(color565(133, 242, 169)); // Clear screen

          GM_DrawEntry_WasChange = false;
          M5.Lcd.setFreeFont(&FreeMonoBold24pt7b); // Select the font
          // Set text datum to middle centre
          M5.Lcd.setTextDatum(MC_DATUM);
          DrawEntryPointer();
        }
        if ((_MHZ19 != NULL) && (needToDraw))
        {
          char buffer[30] = {0};
          snprintf(buffer, 30, "CO");
          M5.Lcd.drawString(buffer, 160, 80, GFXFF); // Print the string
          M5.Lcd.setFreeFont(&FreeMonoBold18pt7b); // Select the font
          snprintf(buffer, 30, "2");
          M5.Lcd.drawString(buffer, 200, 90, GFXFF); // Print the string
          M5.Lcd.setFreeFont(&FreeMonoBold24pt7b);   // Select the font
          int16_t CO2Value = _MHZ19->getCO2();
          if (_MHZ19->_errorCode == MHZ19_M_EC_RESULT_OK)
          {
            snprintf(buffer, 30, "%4i ppm", CO2Value);
          }
```

```
        else
        {
            snprintf(buffer, 30, "  -- ppm");
        }
        M5.Lcd.drawString(buffer, 160, 160, GFXFF); // Print the string

        // CO2Value = rand() % 1100 + 400;

        GM_DrawEntry_LastDraw = millis();
      }
    }
    void GraphMenu::DrawEntry_State()
    {
      bool needToDraw = GM_DrawEntry_WasChange || ((millis() - GM_DrawEntry_LastDraw) >
1000);

      if (GM_DrawEntry_WasChange == true)
      {
        M5.Lcd.setTextColor(TFT_BLACK, TFT_WHITE);
        M5.Lcd.fillScreen(TFT_WHITE); // Clear screen
        GM_DrawEntry_WasChange = false;
        M5.Lcd.setFreeFont(&FreeMonoBold9pt7b); // Select the font
        // Set text datum to top left
        M5.Lcd.setTextDatum(TL_DATUM);
        DrawEntryPointer();
      }
      if (GM_NeedToStartWebSetting)
      {
        M5.Lcd.setTextColor(TFT_BLACK, TFT_WHITE);
        M5.Lcd.fillScreen(TFT_WHITE); // Clear screen
        GM_DrawEntry_WasChange = false;
        M5.Lcd.setFreeFont(&FreeMonoBold24pt7b); // Select the font
        // Set text datum to Middle center left
        M5.Lcd.setTextDatum(TC_DATUM);
        char buffer[30] = {0};
        snprintf(buffer, 30, "Web Setting");
```

```
    M5.Lcd.drawString(buffer, 160, 20, GFXFF); // Print the string
     // Web serserver for setting up Wi-Fi and Blynk credentials
     WebSetting *webSetting = new WebSetting();
     webSetting->setup();
     M5.Lcd.setFreeFont(&FreeMonoBold12pt7b); // Select the font
     snprintf(buffer, 30, "Wi-Fi SSID: %s", WIFI_AP_SSID);
     M5.Lcd.drawString(buffer, 160, 70, GFXFF); // Print the string
     snprintf(buffer, 30, "Web site: %s", WIFI_AP_IP);
     M5.Lcd.drawString(buffer, 160, 95, GFXFF); // Print the string
     M5.Lcd.qrcode(WIFI_AP_SITE, 100, 120, 120, 2);
     // waiting for user action then will be a reboot
     while (true)
     {
        webSetting->loop();
     }
  }
  else
  {

     if ((_DHT22 != NULL) && (needToDraw))
     {
        Preferences preferences;
        preferences.begin(PREF_NAME);
        char buffer[30] = {0};
        if (preferences.getBool(PREF_WIFI_IS_CONFIGURED))
        {
           char ssid[WIFI_MAX_SSID_LENGTH] = {0};
           preferences.getString(PREF_WIFI_SSID, ssid, WIFI_MAX_SSID_LENGTH);
           snprintf(buffer, 30, "Wi-Fi SSID: %s", ssid);
           // snprintf(buffer, 30, "Wi-Fi SSID: %s", WiFi.SSID().c_str());
           M5.Lcd.drawString(buffer, 10, 10, GFXFF); // Print the string
           if (WiFi.status() == WL_CONNECTED)
              snprintf(buffer, 30, "Wi-Fi ip: %s", WiFi.localIP().toString().c_str());
           else
              snprintf(buffer, 30, "Wi-Fi ip: not connected");
           M5.Lcd.drawString(buffer, 10, 30, GFXFF); // Print the string
```

```
// snprintf(buffer, 30, "Wi-Fi key: %s", WiFi.psk().c_str());
            // M5.Lcd.drawString(buffer, 10, 40, GFXFF); // Print the string
            if (preferences.getBool(PREF_BLYNK_IS_CONFIGURED))
            {
               snprintf(buffer, 30, "Blynk: configured");
               M5.Lcd.drawString(buffer, 10, 50, GFXFF); // Print the string
            }
            else
            {
               snprintf(buffer, 30, "Blynk: not configured");
               M5.Lcd.drawString(buffer, 10, 50, GFXFF); // Print the string
            }
         }
         else
         {
            snprintf(buffer, 30, "Wi-Fi: not configured");
            M5.Lcd.drawString(buffer, 10, 10, GFXFF); // Print the string
            snprintf(buffer, 30, "Blynk: not configured");
            M5.Lcd.drawString(buffer, 10, 30, GFXFF); // Print the string
         }
         preferences.end();
         GM_DrawEntry_LastDraw = millis();
      }
   }
}
#ifndef GRAPHMENU_H_
#define GRAPHMENU_H_
#include <stdint.h>
#include <stddef.h>
#include <list>
#include <NTPClient.h>
#include <MHZ19_Modbus.h>
#include <DHTesp.h>
typedef void (*DrawEntry_t)();
typedef std::list<void*> DrawFunctionList_t;
typedef DrawFunctionList_t::iterator DrawFunctionListIterator_t;
```

```
// only one member of this class
class GraphMenu
{
private:
    DrawFunctionList_t _drawFunctionList;
    DrawEntry_t _currentDrawEntry;
    DrawFunctionListIterator_t _currentDrawEntryIterator;
    static NTPClient *_timeClient;
    static MHZ19_Modbus *_MHZ19;
    static DHTesp *_DHT22;
public:
    void Start();
    void Process();
    int8_t AddEntry(DrawEntry_t entry, int8_t position = -1);
    void SetNTPClient(NTPClient *timeClient);
    void SetMHZ19_Modbus(MHZ19_Modbus *MHZ19);
    void SetDHTesp(DHTesp *DHT22);
    static void DrawEntryPointer();

    static void DrawEntry_Time();
    static void DrawEntry_TempAndHum();
    static void DrawEntry_CO2();
    static void DrawEntry_State();
};
#include <MHZ19_Modbus.h>
#define MODBUS_DeviceAddress_Length (1)
#define MODBUS_CommandType_Length (1)
#define MODBUS_DataSize_Length (1)
#define MODBUS_CRC_Length (2)
void MHZ19_Modbus::begin(HardwareSerial &serial, uint32_t config, int8_t rxPin, int8_t txPin)
{
    _MHZ19_Serial = &serial;
    _MHZ19_Serial->begin(9600, config, rxPin, txPin);
    _errorCode = MHZ19_M_EC_RESULT_ERR_NULL;
    _MHZ19_Modbus_Start_TS = millis();
    MHZ19_Modbus_SetupEnded = false;
```

```
}
int16_t MHZ19_Modbus::getCO2()
{
  if ((millis() - _MHZ19_Modbus_Start_TS) < MHZ19_Modbus_SETUP_PERIOD)
  {
    _errorCode = MHZ19_M_EC_RESULT_FAILED;
    return 0;
  }
  //construct command
  constructReadHoldingRegistersCommand(MHZ19_Modbus_Address,
MHZ19_Modbus_CO2_Register_Address, 1);
  emptyInputBuffer();
  //write to serial
  write(_constructedCommand);
  //return response
  uint8_t result = receiveReadHoldingRegistersResponse(_receivedData);
  if (result == 2)
    _lastCO2Value = (_receivedCommand[3] << 8) + _receivedCommand[4];
  else
  {
    // M5.Lcd.printf("Error code:%i\n", _errorCode);
    _errorCode = MHZ19_M_EC_RESULT_FAILED;
    return -1;
  }
  return _lastCO2Value;
}
void MHZ19_Modbus::constructReadHoldingRegistersCommand(uint8_t
MHZ19ModbusAddress, uint16_t StartAddress, uint16_t QuantityOfRegisters)
{
  //address of MHZ19
  _constructedCommand[0] = MHZ19ModbusAddress;
  //operation type
  _constructedCommand[1] = (uint8_t)MHZ19_M_CT_ReadHoldingRegisters;
  //starting address
  _constructedCommand[2] = (uint8_t)((StartAddress >> 8) & 0x00FF);
  _constructedCommand[3] = (uint8_t)(StartAddress & 0x00FF);
```

```
    //quantity of registers
    _constructedCommand[4] = (uint8_t)((QuantityOfRegisters >> 8) & 0x00FF);
    _constructedCommand[5] = (uint8_t)(QuantityOfRegisters & 0x00FF);
    //CRC16
    uint16_t CRC16 = checkSum_CRC16_Modbus(_constructedCommand, 6);
    _constructedCommand[6] = (uint8_t)(CRC16 & 0x00FF);
    _constructedCommand[7] = (uint8_t)((CRC16 & 0xFF00) >> 8);
    _constructedCommandLength = 8;
}
void MHZ19_Modbus::write(uint8_t *DataToSend)
{
    _MHZ19_Serial->write(_constructedCommand, _constructedCommandLength);
    _MHZ19_Serial->flush();
}
void MHZ19_Modbus::emptyInputBuffer()
{
    while (_MHZ19_Serial->available())
    {
        _MHZ19_Serial->read();
    }
}

int8_t MHZ19_Modbus::receiveReadHoldingRegistersResponse(uint8_t *data)
{
    uint8_t addressPointer = 0;
    uint8_t commandTypePointer = addressPointer + MODBUS_DeviceAddress_Length;
    uint8_t dataSizePointer = commandTypePointer + MODBUS_CommandType_Length;
    uint8_t dataPointer = dataSizePointer + MODBUS_DataSize_Length;
    uint8_t CRCPointer = 0;
    //wait for data in serial
    uint32_t timeStamp = millis();
    while (_MHZ19_Serial->available() < (MODBUS_DeviceAddress_Length +
MODBUS_CommandType_Length + MODBUS_DataSize_Length))
    {
        if (millis() - timeStamp >= MHZ19_Modbus_TIMEOUT_PERIOD)
        {
```

```
            _errorCode = MHZ19_M_EC_RESULT_ERR_TIMEOUT;
              return -1;
            }
          }
          uint8_t result = Serial2.readBytes(_receivedCommand, (MODBUS_DeviceAddress_Length +
MODBUS_CommandType_Length + MODBUS_DataSize_Length));
          // M5.Lcd.printf("%X %X %X\n", _receivedCommand[0], _receivedCommand[1],
_receivedCommand[2]);
          //check how much data was received
          if (result != (MODBUS_DeviceAddress_Length + MODBUS_CommandType_Length +
MODBUS_DataSize_Length))
          {
            _errorCode = MHZ19_M_EC_RESULT_ERR_FRAME;
            _errorCode = (MHZ19_Modbus_ERRORCODE)7;
            return -1;
          }
          if (_receivedCommand[addressPointer] != MHZ19_Modbus_Address)
          {
            _errorCode = MHZ19_M_EC_RESULT_ERR_FRAME;
            _errorCode = (MHZ19_Modbus_ERRORCODE)8;
            return -1;
          }

          if (_receivedCommand[commandTypePointer] != MHZ19_M_CT_ReadHoldingRegisters)
          {
            _errorCode = MHZ19_M_EC_RESULT_ERR_FRAME;
            _errorCode = (MHZ19_Modbus_ERRORCODE)9;
            return -1;
          }
          //check too big data
          if (_receivedCommand[dataSizePointer] > (32))
          {
            _errorCode = MHZ19_M_EC_RESULT_ERR_FRAME;
            _errorCode = (MHZ19_Modbus_ERRORCODE)10;
            return -1;
          }
```

```
//wait for data and CRC in serial
  while (_MHZ19_Serial->available() < (_receivedCommand[dataSizePointer] + MODBUS_CRC_Length))
  {
    if (millis() - timeStamp >= MHZ19_Modbus_TIMEOUT_PERIOD)
    {
      _errorCode = MHZ19_M_EC_RESULT_ERR_TIMEOUT;
      return -1;
    }
  }
  result = _MHZ19_Serial->readBytes(_receivedCommand + dataPointer, (_receivedCommand[dataSizePointer] + MODBUS_CRC_Length));

  if (result != (_receivedCommand[dataSizePointer] + MODBUS_CRC_Length))
  {
    _errorCode = MHZ19_M_EC_RESULT_ERR_FRAME;
    _errorCode = (MHZ19_Modbus_ERRORCODE)11;
    return -1;
  }
  CRCPointer = dataPointer + _receivedCommand[dataSizePointer];
  //check CRC16
  // M5.Lcd.printf("m:%X %X %X %X %X %X %X\n", _receivedCommand[0], _receivedCommand[1], _receivedCommand[2], _receivedCommand[3], _receivedCommand[4], _receivedCommand[5],
  // _receivedCommand[6]);
  uint16_t *messageCRC = (uint16_t *)(_receivedCommand + CRCPointer);
  uint16_t CRC = checkSum_CRC16_Modbus(_receivedCommand,
                (MODBUS_DeviceAddress_Length + MODBUS_CommandType_Length + MODBUS_DataSize_Length + _receivedCommand[dataSizePointer]));
  // M5.Lcd.printf("1:%X 2:%X\n", *messageCRC, CRC);
  if (CRC != *messageCRC)
  {
    _errorCode = MHZ19_M_EC_RESULT_ERR_CRC;
    _errorCode = (MHZ19_Modbus_ERRORCODE)12;
    return -1;
```

```
    }
    for (uint8_t i = 0; i < _receivedCommand[dataSizePointer]; i++)
    {
      _receivedData[i] = _receivedCommand[dataPointer + i];
    }
   _errorCode = MHZ19_M_EC_RESULT_OK;
    return _receivedCommand[dataSizePointer];
}
uint16_t MHZ19_Modbus::checkSum_CRC16_Modbus(uint8_t *message, uint8_t length)
{
    uint16_t crc = 0xFFFF;
    for (uint8_t pos = 0; pos < length; pos++)
    {
      crc ^= (uint16_t)message[pos]; // XOR byte into least sig. byte of crc

      for (int i = 8; i != 0; i--)
      { // Loop over each bit
        if ((crc & 0x0001) != 0)
        {             // If the LSB is set
          crc >>= 1; // Shift right and XOR 0xA001
          crc ^= 0xA001;
        }
        else          // Else LSB is not set
          crc >>= 1; // Just shift right
      }
    }
    return crc;
}
#ifndef MHZ19_MODBUS_H_
#define MHZ19_MODBUS_H_
#include <stdint.h>
#include <Arduino.h>
#ifdef ESP32
#include "esp32-hal-log.h"
#endif
#define MHZ19_Modbus_Address              (0x01)
```

```
#define MHZ19_Modbus_CO2_Register_Address   (0x0105)
#define MHZ19_Modbus_TIMEOUT_PERIOD        (50) // (ms) time out period for response
#define MHZ19_Modbus_SETUP_PERIOD          (20000) // (ms)
//enum alias for error code defintions
enum MHZ19_Modbus_ERRORCODE
{
        MHZ19_M_EC_RESULT_ERR_NULL = 0,
        MHZ19_M_EC_RESULT_OK,          //= 1,
        MHZ19_M_EC_RESULT_ERR_TIMEOUT,  // = 2,
        MHZ19_M_EC_RESULT_ERR_FRAME,    // = 3,
        MHZ19_M_EC_RESULT_ERR_CRC,      // = 4,
        MHZ19_M_EC_RESULT_FILTER,       // = 5,
        MHZ19_M_EC_RESULT_FAILED        // = 6
};
//alias for command types
typedef enum MHZ19_Modbus_COMMAND_TYPE
{
   MHZ19_M_CT_ReadHoldingRegisters = 0x03,
   MHZ19_M_CT_ReadInputRegisters = 0x04,
   MHZ19_M_CT_WriteSingleRegister = 0x06,
   MHZ19_M_CT_ReadDeviceIdentification = 0x2b
} MHZ19_Modbus_Command_Type;

class MHZ19_Modbus
{
public:
   //essential begin
   void begin(HardwareSerial &stream, uint32_t config, int8_t rxPin, int8_t txPin);
   //Holds last recieved errorcode from recieveResponse()
   MHZ19_Modbus_ERRORCODE _errorCode;
   //request CO2 values
   int16_t getCO2();
private:
   //pointer for Stream class to accept reference for hardware and software ports
   HardwareSerial *_MHZ19_Serial;
   uint32_t _MHZ19_Modbus_Start_TS;
```

```
    bool MHZ19_Modbus_SetupEnded;
      uint16_t _lastCO2Value;
    //Flag set by setFilter() to signify is "filter mode" was made active
    bool filterMode = false;
    //holder for new commands which are to be sent
    uint8_t _constructedCommand[32];
    uint8_t _constructedCommandLength;
    uint8_t _receivedCommand[32];
    //Constructs commands using command array and entered values
    void constructReadHoldingRegistersCommand(uint8_t MHZ19ModbusAddress, uint16_t StartAddress, uint16_t QuantityOfRegisters = 1);
    //Sends commands to the sensor
    void write(uint8_t *DataToSend);
    void emptyInputBuffer();
    //Assigns response to the correct communcation arrays//
    //void handleResponse(Command_Type commandtype);
    //Call retrieveData to retrieve values from the sensor and check return code
    int8_t receiveReadHoldingRegistersResponse(uint8_t* data);
    uint8_t _receivedData[32];
    //generates a checksum for sending and verifying incoming data
    uint16_t checkSum_CRC16_Modbus(uint8_t *message, uint8_t length);
};
#endif

#include <StateControl.h>
#include <Global.h>
// StateControl::StateControl(MHZ19_Modbus *MHZ19, DHTesp *DHT22, NTPClient *timeClient)
StateControl::StateControl()
{
  // _MHZ19 = MHZ19;
  // _DHT22 = DHT22;
  // _timeClient = timeClient;
  _currentState = SC_WiFiClosed;
  _ssid = new char(WIFI_MAX_SSID_LENGTH);
  _pass = new char(WIFI_MAX_PASSWORD_LENGTH);
```

```
}
StateControl::~StateControl()
{
    delete _pass;
    delete _ssid;
}
void StateControl::Process()
{
    switch (_currentState)
    {
    case SC_WiFiClosed:
        break;
    case SC_WiFiClosedButWantToStartAsClient:
        WiFi.mode(WIFI_STA);
        WiFi.begin(_ssid, _pass);
        _currentState = SC_WiFiConnectingAsClient;
        break;
    case SC_WiFiConnectingAsClient:
        if (WiFi.status() == WL_CONNECTED)
            _currentState = SC_WiFiReadyAsClient;
        break;
    case SC_WiFiReadyAsClient:
        break;
    default:
        break;
    }
}
StateControl::State StateControl::GetCurrentState()
{
    return _currentState;
}
bool StateControl::Command_ConnectWiFiAsClient(char *ssid, char *pass)
{
    if (_currentState == SC_WiFiClosed)
    {
        if ((ssid != NULL) & (pass != NULL) & (*ssid != 0))
```

```
        {
            strncpy(_ssid, ssid, WIFI_MAX_SSID_LENGTH);
            strncpy(_pass, pass, WIFI_MAX_PASSWORD_LENGTH);
            _currentState = SC_WiFiClosedButWantToStartAsClient;
            return true;
        }
    }
    return false;
}
bool StateControl::Command_DisconnectWiFi()
{
    return false;
}

#ifndef STATECONTROL_H_
#define STATECONTROL_H_
#include <WiFi.h>
#include <WiFiClient.h>
class StateControl
{
public:
    enum State
    {
        SC_WiFiClosed = 0,
        SC_WiFiClosedButWantToStartAsClient,
        SC_WiFiConnectingAsClient,
        SC_WiFiReadyAsClient
    };
private:
    // NTPClient *_timeClient;
    // MHZ19_Modbus *_MHZ19;
    // DHTesp *_DHT22;
    State _currentState;
    char *_ssid, *_pass;
public:
    StateControl();
```

```
    ~StateControl();
    void Process(void);
    State GetCurrentState();
    bool Command_ConnectWiFiAsClient(char *ssid, char *pass);
    bool Command_DisconnectWiFi();
};
#endif
#include <WebSetting.h>
#include <Global.h>
WebSetting::WebSetting() : apIP(192, 168, 0, 1), apSSID(WIFI_AP_SSID), webServer(80)
{
}
bool WebSetting::setup()
{
    preferences.begin(PREF_NAME);
    setupMode();
    return true;
}
bool WebSetting::loop()
{
    webServer.handleClient();
    return true;
}
void WebSetting::close()
{
    WiFi.mode(WIFI_MODE_STA);
    preferences.end();
}
void WebSetting::startWebServer()
{
    String strAPIP = WiFi.softAPIP().toString();
      webServer.onNotFound([this]() {
      String s = "<h1>CO2 Meter Wi-Fi and Blynk Settings</h1>"
            "<p>Please enter your Wi-Fi password by selecting the SSID.</p>"
            "<form method=\"get\" action=\"setap\">"
            "<label>Wi-Fi SSID: </label>"
```

```
                "<select name=\"ssid\">" +
                ssidList +
                "</select>"
                "<br>Password: <input name=\"pass\" length=64 type=\"password\">"
                "<p>Please enter your Blynk key.</p>"
                "<br>Blynk Project Auth Token: <input name=\"token\"length=64 type=\"password\">"
                "<input type=\"submit\" value=\"Save\"></form>"
                "<h2>Reset device to factory settings</h2>"
                "<form method=\"get\" action=\"factoryreset\">"
                "<input type=\"submit\" value=\"Reset\"></form>";
            webServer.send(200, "text/html", makePage("CO2 Settings", s));
        });

        webServer.on("/setap", [this]() {
            String ssid = urlDecode(webServer.arg("ssid"));
            String pass = urlDecode(webServer.arg("pass"));
            String token = urlDecode(webServer.arg("token"));

            // Store wifi config
            // M5.Lcd.println("Writing Password to nvr...");
            if (ssid.length())
            {
                preferences.putBool(PREF_WIFI_IS_CONFIGURED, true);
                preferences.putString(PREF_WIFI_SSID, ssid);
                preferences.putString(PREF_WIFI_PASSWD, pass);
            }
            if (token.length())
            {
                preferences.putBool(PREF_BLYNK_IS_CONFIGURED, true);
                preferences.putString(PREF_BLYNK_TOKEN, token);
            }
            // M5.Lcd.println("Write nvr done!");
            String s = "<h1>Setup complete.</h1><p>Device will be connected to \"" + ssid + "\" after the restart.</p>";
            webServer.send(200, "text/html", makePage("CO2 Settings", s));
```

```
        delay(3000);

        close();
        ESP.restart();
    });
    webServer.on("/factoryreset", [this]() {
        // Store wifi config
        preferences.putBool(PREF_WIFI_IS_CONFIGURED, false);
        preferences.putBool(PREF_BLYNK_IS_CONFIGURED, false);

        String s = "<h1>Factory reset complete.</h1><p>Device will restart.</p>";
        webServer.send(200, "text/html", makePage("CO2 Settings", s));
        delay(3000);

        close();
        ESP.restart();
    });
    webServer.begin();
}
void WebSetting::setupMode()
{
    WiFi.mode(WIFI_MODE_STA);
    WiFi.disconnect();
    delay(100);
    int n = WiFi.scanNetworks();
    delay(100);
    M5.Lcd.println("");
    for (int i = 0; i < n; ++i)
    {
        ssidList += "<option value=\"";
        ssidList += WiFi.SSID(i);
        ssidList += "\">";
        ssidList += WiFi.SSID(i);
        ssidList += "</option>";
    }
    // create Wi-Fi AP for WebServer
```

```
    WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
    WiFi.softAP(apSSID);
    WiFi.mode(WIFI_MODE_AP);
    startWebServer();
}

String WebSetting::makePage(String title, String contents)
{
    String s = "<!DOCTYPE html><html><head>"
            "<meta name=\"viewport\" content=\"width=device-width,user-scalable=0\">"
            "<title>" +
            title +
            "</title></head><body>" + contents +
            "</body></html>";
    return s;
}
String WebSetting::urlDecode(String input)
{
    String s = input;
    s.replace("%20", " ");
    s.replace("+", " ");
    s.replace("%21", "!");
    s.replace("%22", "\"");
    s.replace("%23", "#");
    s.replace("%24", "$");
    s.replace("%25", "%");
    s.replace("%26", "&");
    s.replace("%27", "\'");
    s.replace("%28", "(");
    s.replace("%29", ")");
    s.replace("%30", "*");
    s.replace("%31", "+");
    s.replace("%2C", ",");
    s.replace("%2E", ".");
    s.replace("%2F", "/");
    s.replace("%2C", ",");
```

```
    s.replace("%3A", ":");
    s.replace("%3A", ";");
    s.replace("%3C", "<");
    s.replace("%3D", "=");
    s.replace("%3E", ">");
    s.replace("%3F", "?");
    s.replace("%40", "@");
    s.replace("%5B", "[");
    s.replace("%5C", "\\");
    s.replace("%5D", "]");
    s.replace("%5E", "^");
    s.replace("%5F", "-");
    s.replace("%60", "`");
    return s;
}
#ifndef _WEBSETTING_H_
#define _WEBSETTING_H_
#include <M5Stack.h>
#include <ESPmDNS.h>
#include <Preferences.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
class WebSetting
{
public:
  bool setup();
  bool loop();
  void close();
  WebSetting();
private:
  long counter = 0;
  const IPAddress apIP;
  const char* apSSID;
  String ssidList;
  String wifi_ssid;
```

```
  String wifi_password;
  WebServer webServer;
  // wifi config store
  Preferences preferences;
  void startWebServer();
  void setupMode();
  String makePage(String title, String contents);
  String urlDecode(String input);
};
#endif
```