Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« _____ » _____ 20___ г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Система сбора метеоданных
тема

Руководитель	_____ подпись, дата	доцент, канд. физ.-мат. наук должность, ученая степень	К. В. Коршун инициалы, фамилия
Выпускник	_____ подпись, дата		Д. В. Шафигуллин инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, канд. физ.-мат. наук должность, ученая степень	К. В. Коршун инициалы, фамилия

Красноярск 2023

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О.В. Непомнящий
подпись инициалы, фамилия
« _____ » _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Красноярск 2023

Студенту Шафигуллину Дмитрию Вадимовичу
фамилия, имя, отчество

Группа КИ19-08Б Направление (специальность) 09.03.01
Номер Код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы Система сбора метеоданных

Утверждена приказом по университету № _____ от _____

Руководитель ВКР К.В. Коршун, канд. физ.-мат. наук, доцент научно-учебной лаборатории телекоммуникационных систем ИКИТ СФУ
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР Задание на ВКР

Перечень разделов ВКР Анализ задания и определение требований, проектирование системы, реализация системы.

Перечень графического материала Презентация в формате Microsoft PowerPoint.

Руководитель ВКР _____ К.В. Коршун
подпись инициалы, фамилия

Задание принял к исполнению _____ Д.В. Шафигуллин
подпись инициалы, фамилия

« __ » _____ 20___ г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Система сбора метеоданных» содержит 57 страниц текстового документа, 14 иллюстраций, 5 таблиц, 11 использованных источников и 4 приложения.

МЕТЕОДАННЫЕ, СБОР, БАЗА ДАННЫХ, ГРАФИКИ, ARDUINO, МИКРОКОНТРОЛЛЕР, ДАТЧИКИ, ВЕБ-СЕРВЕР, СХЕМА ПОДКЛЮЧЕНИЯ.

Актуальность – сбор метеоданных имеет огромное значение в современном мире. Они необходимы в различных областях жизни, таких как сельское хозяйство и прогнозирование погоды. Метеоданные помогают определить наилучшее время для посева и сбора урожая, позволяют планировать использование энергетических ресурсов и эффективно управлять водными ресурсами. Эти данные влияют на принятие важных решений, которые напрямую влияют на качество жизни людей.

Объект исследования – система сбора метеоданных.

Предмет исследования – сбор, хранение и отображения актуальной информации в доступном формате.

Цель работы – разработать и реализовать систему, способную собирать, хранить и обрабатывать метеоданные, предоставлять пользователям актуальную, точную и достоверную информацию о погодных условиях на определенной местности.

Главные задачи системы:

– сбор метеоданных;

– хранение метеоданных;

– обработка метеоданных;

– отображение данных на графиках через веб-сервер.

Практическая значимость проекта заключается в возможности получения актуальной информации о погодных условиях в удобном формате.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современной метеорологии сбор, анализ и обработка данных играют ключевую роль в изучении погодных условий и климатических изменений. Непрерывный мониторинг и анализ данных являются неотъемлемой частью процесса прогнозирования погоды, понимания климатических тенденций и принятия соответствующих метеорологических мер.

Одной из наиболее значимых задач в метеорологии является наблюдение и запись показателей, таких как температура, влажность и давление. Эти данные являются фундаментальными для анализа и прогнозирования погодных явлений, а также для изучения климатических изменений на глобальном и региональном уровнях.

Современные метеорологические станции, оснащенные передовыми датчиками и системами сбора данных, способны непрерывно и автоматически регистрировать показатели и передавать их для дальнейшей обработки. Это позволяет метеорологам получать актуальные данные в реальном времени и анализировать их для прогнозирования погоды и изучения климатических закономерностей.

Сбор, анализ и обработка данных в метеорологической среде имеют огромное значение. Они позволяют метеорологам выявлять паттерны, связи и тенденции в погодных явлениях, определять риски и предупреждать о неблагоприятных условиях. Это помогает гражданам, государственным организациям и промышленным предприятиям принимать соответствующие меры для обеспечения безопасности и эффективности своих операций.

1 Анализ задания и определение требований

В данной главе проведен анализ задания на выпускную квалификационную работу: раскрыта его цель, актуальность и содержание. Проведен анализ предметной области, в котором раскрыто понятие метеостанции и ее классификации, приведена сравнительная характеристика существующих решений, на основании которой был сделан вывод о преимуществах и недостатках каждой из готовых систем. Были обозначены возможные подходы к созданию системы и сделан выбор в пользу одного из них. Сформулированы требования к разрабатываемой системе сбора метеоданных, затрагивающие как аппаратную, так и программную части проекта, их функциональные особенности. Определено: какие параметры погоды следует собирать, какие источники данных использовать, как организовать хранение и обработку данных. В конце главы сделан вывод.

1.1 Классификации систем сбора метеоданных

В рамках данной выпускной квалификационной работы используются формулировки – система или устройство сбора метеоданных, которые равноправно могут быть заменены словом – метеостанция. Метеостанция – это комплексное измерительное устройство, которое позволяет измерять и регистрировать различные метеорологические параметры в окружающей среде. Метеостанции могут быть классифицированы по различным критериям, например:

– по функциональности: базовые и продвинутые. Базовые метеостанции обычно предназначены для измерения основных параметров погоды, таких как температура, давление, влажность, скорость и направление ветра. Продвинутые метеостанции могут также включать дополнительные датчики, такие как датчик осадков, UV-индекса, ультразвуковой дальномер и другие;

– по принципу измерения: аналоговые и цифровые. Аналоговые метеостанции используют механические и электромеханические датчики, такие как биметаллические термометры и анероидные барометры. Цифровые метеостанции используют электронные датчики, которые обеспечивают более точные измерения и позволяют автоматически записывать и передавать данные;

– по области применения: метеостанции для дома, сада, сельского хозяйства, морской и авиационной навигации, метеостанции для научных исследований;

– по типу питания: метеостанции, работающие от солнечных батарей, батареек, USB-порта, электрической сети;

– по точности измерения: метеостанции для обычных пользователей, метеостанции для профессионалов, научные метеостанции.

В выпускной квалификационной работе рассматриваются этапы создания бытовой домашней метеостанции.

1.2 Сравнение существующих решений

На рынке присутствует достаточное количество готовых систем, ориентированных на выполнение задач сбора метеоданных. Однако, в некоторых случаях, эти системы не могут полностью удовлетворить требованиям конкретных пользователей. В таких ситуациях возможным решением является создание собственной системы сбора метеоданных, которая будет адаптирована к индивидуальным потребностям и требованиям. В данной работе будет рассмотрено проектирование и разработка такой системы. Для этого будут проанализированы существующие решения, определены требования к системе и спроектирована ее архитектура.

Датчики в метеорологии, измеряющие температуру называются – термометры, для измерения атмосферного давления – барометры, для измерения влажности воздуха – гигрометры.

В таблице 1 содержатся сведения о домашних метеостанциях в пределах стоимости до 10 тысяч рублей.

Таблица 1 – Примеры домашних метеостанций

Модель	**Цена, руб**	**Диапазон темп., °C**	**Диапазон влажн. (rH), %**	**Возможность измерения атм. давления**	**Тип питания**	**Габариты (ВхШхГ), мм**	**Вес, кг**
La Crosse Technology S84107 (Китай)	9999	-40...+60	10...99	Да	Сеть + батарейки	165x105x29	0.2
TFA Nexus (Германия)	8999	-20...+60	1...99	Да	Сеть + батарейки	180x110x35	0.4
ADE-WS (Германия)	7990	-20...+60	20...95	Да	Сеть + батарейки	140x80x25	0.3
ADE WS1703 (Германия)	7990	-10...+60	20...99	Да	Сеть + батарейки	170x140x35	0.6
TFA Nexus Plus (Германия)	6999	-20...+60	1...99	Да	Сеть + батарейки	193x133x45	0.5
MeteoHelix (Словакия)	6575	-40...+85	0...100	Да	Сеть + солнечная батарея	155x80x50	0.5
Technoline WS 6860 (Германия)	5850	-10...+60	20...95	Да	Сеть + батарейки	170x120x30	0.3
Oregon Scientific BAR268HG (Китай)	4825	-5...+50	1...99	Да	Сеть + батарейки	128x110x38	0.3
TFA Cosy (Германия)	3450	0...+50	20...95	Нет	Сеть + батарейки	106x47x96	0.2
Dostmann Electronic TFA 35.1089 (Германия)	3245	-10...+60	10...99	Да	Сеть + батарейки	118x62x19	0.1

Так же существуют и профессиональные метеостанции. Основное отличие профессиональных метеостанций от домашних заключается в их более высокой точности и функциональности. Профессиональные метеостанции имеют более широкий диапазон измеряемых параметров и

более точные датчики, что позволяет получать более точные и надежные данные.

Кроме того, профессиональные метеостанции имеют более продвинутые функции обработки и анализа данных, такие как возможность автоматической записи и обработки данных, анализ трендов, создание графиков и диаграмм, а также передачу данных на компьютер или интернет.

Профессиональные метеостанции также имеют более прочную конструкцию и могут работать в более экстремальных условиях, таких как высокая влажность, сильный ветер, высокие или низкие температуры.

Однако, профессиональные метеостанции обычно стоят значительно дороже, чем домашние, и требуют более сложной установки и настройки. Они также могут быть более громоздкими и тяжелыми, что может затруднить их перемещение и установку в разных местах.

В таблице 2 представлены примеры профессиональных метеостанций.

Таблица 2 – Примеры профессиональных метеостанций

Модель	**Цена, руб**	**Диапазон темп., °С**	**Диапазон rH, %**	**Диапазон измеряемого атм. давления, мм рт. ст.**	**Точность измерения, °С**	**Габариты (ВхШхГ), мм**	**Вес, кг**
Davis Vantage Pro2 (США)	59 990	-40...65	0...100	510...800	+/- 0.3	318x508 x44	4.1
Ambient Weather WS-2000	22 490	-40...60	10...99	300...1100	+/- 0.5	178x203 x76	1.36
Netatmo Weather Station	17 990	-40...65	0...100	–	+/- 0.5	45x45x1 55	0.7
La Crosse Technology C83100 (Китай)	12 290	-40...60	10...99	300...1100	+/- 0.5	292x330 x44	2.2
Oregon Scientific WMR86	9 990	-30...60	10...99	700...1100	+/- 1	152x203 x29	0.35

1.2.1.1 Результат сравнения существующих решений

Проанализировав таблицы и сравнив виды метеостанций, был сделан вывод, что каждый вид метеостанций имеет свои преимущества и недостатки, которые зависят от требований и целей использования. Например, профессиональные метеостанции обладают большей точностью и функциональностью, но также являются более дорогостоящими и требуют профессиональной настройки и обслуживания. В то же время, домашние метеостанции предоставляют базовый набор функций и обычно более доступны в цене, но могут иметь ограничения в точности измерений и функциональности.

В конечном итоге, разрабатываемая система может быть наделена некоторыми из следующих функций: измерение температуры воздуха внутри и снаружи помещения, измерение относительной влажности воздуха внутри и снаружи помещения. измерение атмосферного давления, измерение скорости и направления ветра, предоставление прогноза погоды на основе собранных данных, предоставление графического отображения данных на дисплеи, возможность управления дополнительными устройствами, такими как обогреватели или кондиционеры, на основе собранных данных, связь с мобильным приложением для удаленного мониторинга данных, измерение качества воздуха внутри помещения.

1.3 Возможные подходы к реализации проекта

На основании имеющихся систем и принципах их работы, определены следующие возможные подходы к реализации проекта:

а) через телеграм бота - этот подход может быть полезен, если нужно получать уведомления о текущих значениях метеоданных, а также устанавливать определенные пороговые значения, при достижении которых

бот будет высылать оповещения. Но при этом, использование бота может ограничить функционал и гибкость системы;

б) через подключения платы к интернету и отправки данных на сервер - этот подход даст большую гибкость в обработке данных и отображении их на графиках, но потребует наличия постоянного интернет-соединения и наличия сервера для обработки данных;

в) реализация веб-сервера непосредственно на самой плате - этот подход может быть полезен, если нужно быстро получить данные и отобразить их на графиках, и нет необходимости использовать сервер. Но при этом, плата может иметь ограниченные ресурсы для обработки данных и может быть сложно настроить веб-сервер на плате;

г) реализация приложения на смартфоне, которое будет получать данные от метеостанции через Bluetooth или Wi-Fi и отображать их на экране смартфона;

д) реализация метеостанции в виде автономного устройства с возможностью записи данных на SD-карту. Данные можно будет потом перенести на компьютер для анализа и построения графиков.

1.4 Выбор варианта реализации. Обоснование

Выбор был сделан в пользу третьего варианта – «реализация веб-сервера на самой плате».

Преимущества:

а) более быстрый доступ к данным. Поскольку данные хранятся локально на плате, доступ к ним может быть более быстрым, чем в случае, если данные хранятся на удаленном сервере;

б) уменьшение затрат на обслуживание. Поскольку все данные хранятся и обрабатываются на одной плате, это может уменьшить затраты на обслуживание и поддержку системы. Не придется платить за аренду сервера;

в) автономность системы. Реализация проекта таким образом позволит создать максимально независимую от сторонних решений систему.

Недостатки:

а) ограниченность ресурсов платы. Платы могут иметь ограниченные ресурсы, такие как объем памяти и процессорная мощность, что может ограничить возможности системы;

б) ограниченность масштабируемости. Поскольку все данные хранятся локально на плате, масштабирование системы может оказаться затруднительным;

в) более сложная настройка. Настройка веб-сервера непосредственно на плате может быть более сложной, чем настройка сервера на удаленном компьютере.

Однако существенным недостатком является лишь сложность реализации, поскольку в рамках выпускной квалификационной работы не ставится цель работы с большим объёмом данных или возможная масштабируемость. Если такая цель возникнет, то всегда возможно увеличить встроенный объём запоминающего устройства путем установки SD карты, в специальный слот.

1.5 Требования к создаваемой системе

Главные задачи системы:

а) сбор метеоданных. Для этого нужно понять, какие устройства возможно использовать для измерения различных показателей погоды, таких как температура воздуха, влажность, давление, скорость и направление ветра или другие;

б) хранение метеоданных. Для этого следует использовать базу данных, которая сможет хранить все собранные данные. База данных должна быть масштабируемой, чтобы можно было добавлять новые данные по мере их поступления;

в) обработка метеоданных. Для этого нужно разработать алгоритмы, которые будут анализировать данные и генерировать отчеты и графики, показывающие изменения погодных условий во времени;

г) предоставление доступа к метеоданным. Для этого нужно разработать веб-интерфейс, который позволит пользователям просматривать отчеты и графики.

1.5.1 Требования к аппаратной части

К аппаратной части в разрабатываемой системе относятся: источник питания, датчики, устройство управления, устройства вывода и индикации, модуль Wi-Fi. На рисунке 1 представлен прототип структурной схемы.

Рисунок 1 – Прототип структурной схемы системы

– источник питания. Необходим для питания всей системы. Минимальное напряжение на выходе должно быть не меньше напряжения питания устройства управления. Точный ток потребления зависит от

конкретных характеристик подключенных датчиков, поэтому рекомендуется провести расчеты и выбрать источник питания, который сможет обеспечить достаточный ток для всех компонентов системы. Размеры предъявляемые к источнику питания должны не сильно отличаться от размеров самого устройства управления, для возможности помещения всей системы в закрытый короб. Большая независимость в работе системы возможна, если дополнить источник питания модулем зарядки;

– датчики. Датчики – это физические устройства, которые преобразуют физические величины в измеряемый сигнал, это ключевой элемент всей системы. Необходимо и достаточно, чтобы датчики делали срез по таким основным метеопараметрам как – температура, влажность, атмосферное давление, содержание CO2 в воздухе, а также имели поддерживаемый платой интерфейс для подключения;

– устройство управления. Это главный компонент системы. Под устройством управления подразумевается контроллер, способный выполнять функции сбора и передачи данных, управление подключением к сети, осуществлять взаимодействием с другими устройствами. Требования к устройству следующие: дешевизна, простота использования, достаточное количество литературы и библиотек для работы, вычислительная мощность и память должны покрывать необходимые потребности для работы станции.

– устройства вывода и индикации. Элемент системы, необходимый для упрощения пользования продуктом. Требования следующие: возможность отображать информацию с устройства управления в реальном времени, иметь небольшое энергопотребление, иметь обширную документацию и библиотеки для работы, иметь поддерживаемый платой интерфейс для подключения;

– модуль Wi-Fi. Модуль необходим для возможности работы с сетью. Отправки и получения пакетов. Основными требования являются: работа на частоте 2.4 или 5 ГГц, низкое энергопотребление, наличие библиотек для настройки и работы.

Основными требованиями, предъявляемыми ко всей аппаратной части, являются: компактность, дешевизна и простота в настройке.

1.5.2 Требования к программной части

К программной части проекта относятся: язык программирования, среда разработки, способы хранения и обработки данных, отображение информации и настройка веб-сервера:

– язык программирования. Требуется найти язык, с которым имеется опыт работы, который будет высокоэффективен в условиях ограниченных ресурсов платы, где нужно максимально эффективно использовать каждый байт памяти и такт процессора, который имеет поддержку сообщества разработчиков, готовые примеры, библиотеки, фреймворки;

– среда разработки. От среды разработки требуется поддержка выбранного языка, удобный и понятный интерфейс, для написания кода, поддержка выбранной платы, возможность управления выбранными библиотеками, наличие инструментов анализа и отладки ошибок;

– способы хранения и обработки данных. Согласно выбранному способу реализации, хранение будет осуществляться во встроенной памяти микроконтроллера. Но при использовании этого способа нужно учитывать ограниченный объем памяти устройства. Обработкой данных называется процесс перехода сырых данных, в готовые данные для отображение на веб-сервере;

– отображение информации. Отображение данных на дисплее – пользователь может увидеть текущие данные о температуре и влажности, который является частью метеостанции. Отображение данных на дисплее позволяет пользователям просматривать информацию без необходимости подключения к интернету; веб-сервер может отображать текущие показания всех датчиков в виде графиков;

– веб-сервер. Главная задача веб-сервера это отображение веб-страницы и обновление данных на графиках.

1.5.3 Функциональные возможности системы

Функциональные возможности разрабатываемой системы включают:

– сбор и обработка данных с датчиков температуры, влажности, давления и уровня углекислого газа;

– отображение текущих значений температуры, влажности и давления на дисплее;

– отображение истории измерений на странице веб-сервера в виде 3-ех графиков и хранение истории измерений на плате в памяти;

– индикация о содержании уровня углекислого газа в воздухе путем смены цвета RGB-светодиода;

– работа в автономном режиме от собственного источника питания;

– возможность подзарядки источника питания.

1.6 Вывод по первой главе

В первой главе был проведен тщательный анализ поставленной задачи, изучены существующие аналоги и прототипы разрабатываемой системы. Были выявлены и проанализированы их недостатки и преимущества, на основе которых была обоснована актуальность и возможность реализации проекта. Кроме того, были оценены различные подходы к реализации системы, и сделан выбор в пользу одного из них.

Далее были составлены необходимые требования к программной и аппаратной частям системы, создан прототип структурной схемы, отражающий ключевые компоненты системы сбора метеоданных.

2 Проектирование системы

Во второй главе, на основе составленных требований к системе и ее функциональных особенностей, полученных на предыдущем этапе, был осуществлен выбор компонентов аппаратной части: микроконтроллера, датчиков, дисплея, источника питания. Рассчитана максимально возможная потребляемая мощность системы. Определен язык программирования со средой разработки и выбран метод работы с данными на веб-сервере. Письменные изложения дополнены графическими схемами и таблицами. Перечислены необходимые знания для успешной реализации работы, рассчитана стоимость всех компонентов, составлен текст-описание внешнего вида системы и описан принцип работы всего проекта В конце главы составлен план реализации всего проекта и сделан вывод по второй главе.

2.1 Выбор компонентов аппаратной части

При выборе компонентов были учтены требования к разрабатываемой системе. К аппаратной части относятся – микроконтроллер, датчики, дисплей, светодиод, Wi-Fi модуль и система питания, представленная источником питания и зарядным модулем.

2.1.1 Выбор микроконтроллера

Выбор платы, как устройства управления, основан на таблице 3.

Таблица 3 – Таблица сравнения некоторых популярных плат

Плата	CPU	Частота процессора	ОЗУ	WiFi	Bluetooth	Ethernet	USB	Цена
Arduino Uno	ATmega328P	16 МГц	2 КБ	Нет	Нет	Нет	1	$20
Arduino Mega	ATmega2560	16 МГц	8 КБ	Нет	Нет	Нет	4	$40

Окончание таблицы 3

Плата	CPU	Частота процессора	ОЗУ	WiFi	Bluetooth	Ethernet	USB	Цена
Raspberry Pi 3	ARM Cortex-A53	1.2 ГГц	1 ГБ	Да	Да	Да	4	$35
STM32 Nucleo	ARM Cortex-M4	168 МГц	1 МБ	Нет	Нет	Нет	1	$20-$30
ESP32	Dual-core Tensilica LX6	240 МГц	520 КБ	Да	Да	Нет	1	$5-$10

Raspberry Pi 3 имеет более мощный процессор и большой объем оперативной памяти, что может быть полезно для обработки большого объёма данных. Arduino Mega и STM32 Nucleo имеют множество портов USB, что может быть полезно для подключения дополнительных устройств. Выбор платы был сделан в сторону ESP32.

2.1.2 Характеристика микроконтроллера ESP32

ESP32 – это небольшой, высокопроизводительный, двухъядерный микроконтроллер с поддержкой Wi-Fi и Bluetooth, производимый компанией Espressif Systems. Он был выпущен в 2016 году и быстро завоевал популярность среди разработчиков благодаря своей высокой производительности, низкому энергопотреблению и богатой функциональности. Основные характеристики ESP32:

– модуль: ESP32-WROOM с чипом ESP32-D0WDQ6;

– частота беспроводной передачи: 2,4 ГГц;

– стандарт Wi-Fi: 802.11b/g/n;

– стандарт Bluetooth: BLE v4.2 BR/EDR;

– тактовая частота: до 240 МГц;

– flash-память: 448 КБ;

– внешняя Flash-память: 4 МБ;

– SRAM-память: 520 КБ;

– пинов общего назначения: 25 ввода-вывода (GPIO) и 4 ввода (GPI);

– контактов с АЦП: 15;

– разрядность АЦП: 12 бит;

– контактов с ЦАП: 2;

– разрядность ЦАП: 8 бит;

– контактов с ШИМ: 21 (16 каналов);

– разрядность ШИМ: 16 бит;

– контактов с ёмкостным сенсором: 8;

– пинов с прерываниями: 25;

– аппаратные интерфейсы: 3×SPI, 3×UART, 2×I2C и 2×I2S;

– напряжение логических уровней: 3,3 В;

– максимальный ток с пина или на пин: 12 мА;

– максимальный выходной ток пина 3V3: 1 А;

– входное напряжение через пин Vin: 5–14 В;

– поддержка многопоточности и аппаратного ускорения для выполнения сложных вычислений.

Чип ESP32-D0WDQ6 — выполнен по технологии SoC (англ. System-on-a-Chip — система на кристалле), в которую входит 2-ядерный 32-битный процессор Tensilica Xtensa LX6 с блоками памяти ROM на 448 КБ и SRAM на 520 КБ. В кристалле также расположены беспроводные технологии Wi-Fi/Bluetooth, радио-модуль, датчик Холла и сенсор температуры.

USB-UART преобразователь. Преобразователь USB-UART на микросхеме CP2102 обеспечивает связь модуля ESP32-WROOM с USB-портом компьютера. При подключении к ПК — платформа ESP32 DevKit определяется как виртуальный COM-порт.

Разъём micro-USB. Предназначен для прошивки и питания платформы ESP32 DevKit с помощью компьютера.

Кнопка EN. Кнопка предназначена для ручного сброса программы — аналог кнопки RESET обычного компьютера.

Кнопка BOOT. Кнопка служит для ручного перевода модуля в режим прошивки.

Регулятор напряжения. Линейный понижающий регулятор напряжение AMS1117-3.3 обеспечивает питание микроконтроллера. Выходное напряжение 3,3 вольта с максимальным током 1 А.

2.1.3 Выбор датчиков

Датчики на рынке представлены небольшим количеством, некоторые из них отображены в таблице 4.

Таблица 4 – Сравнение датчиков

Датчик	Диапазон измерений	Точность измерений	Интерфейс связи	Цена (примерно)	Преимущества
BME280	-40°C...+85°C	±1°C	I2C, SPI	$3-$5	Высокая точность измерений температуры, влажности и давления. Компактный размер и низкое энергопотребление.
	0...100% RH	±3% RH			Высокая точность измерений температуры и влажности. Низкая стоимость. Низкая точность измерений влажности.
	300...1100 hPa	±1 hPa			
DHT22	-40°C...+80°C	±0.5°C	OneWire	$3-$5	
	0...100% RH	±2-5% RH			Высокая точность измерений температуры и влажности. Высокая стоимость.
SHT3x	-40°C...+125°C	±0.2°C	I2C	$8-$15	
	0...100% RH	±1.5% RH			Высокая точность измерений температуры и давления. Низкое энергопотребление. Не измеряет влажность.
BMP280	-40°C...+85°C	±1°C	I2C, SPI	$2-$4	

При выборе датчиков необходимо учитывать их интерфейс подключения к плате ESP32 и наличие готовых библиотек для работы с

ними. В нашем случае высокая точность измерений не является необходимым условием, поэтому можно обратить внимание на универсальный и недорогой датчик BMP280, и простой датчик-детектор превышения уровня CO2 в воздухе – MQ135.

2.1.4 Выбор дисплея

Существует несколько видов дисплеев, которые могут быть использованы в системе сбора данных:

ЖК-дисплеи, или жидкокристаллические дисплеи, являются наиболее распространенным типом дисплеев. Они используют жидкие кристаллы, контролируемые электрическим током, для создания изображения.

OLED-дисплеи обладают высокой контрастностью, яркими цветами и низким энергопотреблением без подсветки. Однако они могут быть дороже и иметь ограниченную продолжительность службы. LED-дисплеи с светодиодами обеспечивают отличную яркость, контрастность и энергоэффективность.

LED-дисплеи яркие и читаемые на солнце, но могут быть дорогими, потреблять много энергии и иметь ограниченный угол обзора. Выбор зависит от предпочтений пользователя и требуемых характеристик.

При соблюдении общих требований, где требования к дисплею следующие: известный интерфейс для подключения, наличие библиотек, небольшое энергопотребление, небольшое разрешение экрана, необходимое и достаточное для отображения трех метеопараметров, дешевизна, был выбран OLED-дисплей, с размерами диагонали 0,91 дюйма.

2.1.5 Выбор источника питания

Сначала необходимо рассчитать потребляемую мощность всей системы. Для расчета потребляемой мощности необходимо знать значения потребления каждого компонента в системе.

Для ESP32 в режиме работы с Wi-Fi модулем потребляемая мощность может колебаться в диапазоне от 70 мА до 200 мА в зависимости от используемого режима передачи данных и мощности передачи.

Датчик MQ135 потребляет около 150 мкА в режиме ожидания и до 50 мА при работе в активном режиме.

OLED-дисплей на базе SSD1306 потребляет до 25 мА в активном режиме, при этом энергопотребление зависит от яркости и количества отображаемой информации.

Датчик BME280 потребляет до 1 мА в режиме ожидания и до 3 мА при работе в активном режиме.

Светодиод обычно потребляет от 5 мА до 20 мА в зависимости от яркости и цвета.

Суммируя все компоненты, получим оценку максимальной потребляемой мощности нашей системы:

200 мА (ESP32 в режиме работы с Wi-Fi модулем) + 50 мА (MQ135) + 25 мА (OLED-дисплей) + 3 мА (BME280) + 20 мА (светодиод) = 298 мА

Таким образом, максимальная потребляемая мощность нашей системы составляет около 300 мА. Однако, эта лишь приблизительная оценка максимально потребляемой мощности.

Питание платы и подключенных к ней компонентов мощно осуществить при помощи:

– батареи – простой, доступный вариант;

– адаптера питания – тогда придется пожертвовать главным условием всей системы – автономностью работы, а это невозможно;

– солнечная батарея – трудности при настройке, дороговизна, ограниченность в использовании.

Мной выбран US18650VTC6 – это литий-ионный аккумулятор производства Sony. Его номинальное напряжение составляет 3,6 В, емкость - 3120 мАч, максимальный ток разряда - 30 А.

Время работы (в часах) = емкость аккумулятора (в мАч) / потребляемый ток (в мА).

Примерное время работы системы сбора метеоданных = 3120 мАч/ 300 мА = около 10 часов до полной разрядки аккумулятора.

Представляется возможным дополнить систему функцией подзарядки, путем установки модуля зарядного устройства литиевой батареи TP4056.

2.2 Выбор компонентов программной части

В этом разделе был сделан выбор: языка программирования, среды разработки, веб-сервера и способа хранения информации.

2.2.1 Выбор языка программирования

Данный пункт зависит от личных предпочтений и имеющегося опыта работы. Мой выбор пал на IDE от фирмы Arduino и язык программирования C++.

На ESP32 можно программировать на нескольких языках, включая Python, Lua и JavaScript. Однако, наиболее распространенным является C++, поскольку он обеспечивает более низкий уровень абстракции и более прямой доступ к аппаратным ресурсам, позволяет работать с железом напрямую,

имеет широкую поддержку и обширную библиотеку для работы с датчиками и другими периферийными устройствами.

2.2.2 Выбор среды разработки

Arduino IDE, так как она проста в использовании, имеет понятный интерфейс и хорошую поддержку для ESP32. Кроме того, в Arduino IDE доступны множество библиотек и примеров кода для работы с различными устройствами.

Среди альтернативных сред разработки для ESP32 можно выделить следующие:

– «Visual Studio Code»: популярный текстовый редактор с поддержкой различных языков программирования, в том числе C++. Существуют дополнительные плагины для работы с платами ESP32.

– «Eclipse»: расширяемая среда разработки с открытым исходным кодом, подходящая для разработки на C++.

– «PlatformIO»: платформа для разработки, которая позволяет работать с различными микроконтроллерами, включая ESP32. Она предоставляет удобный интерфейс для управления библиотеками, загрузки кода на плату и отладки программы.

2.2.3 Выбор веб-сервера и способа хранения информации

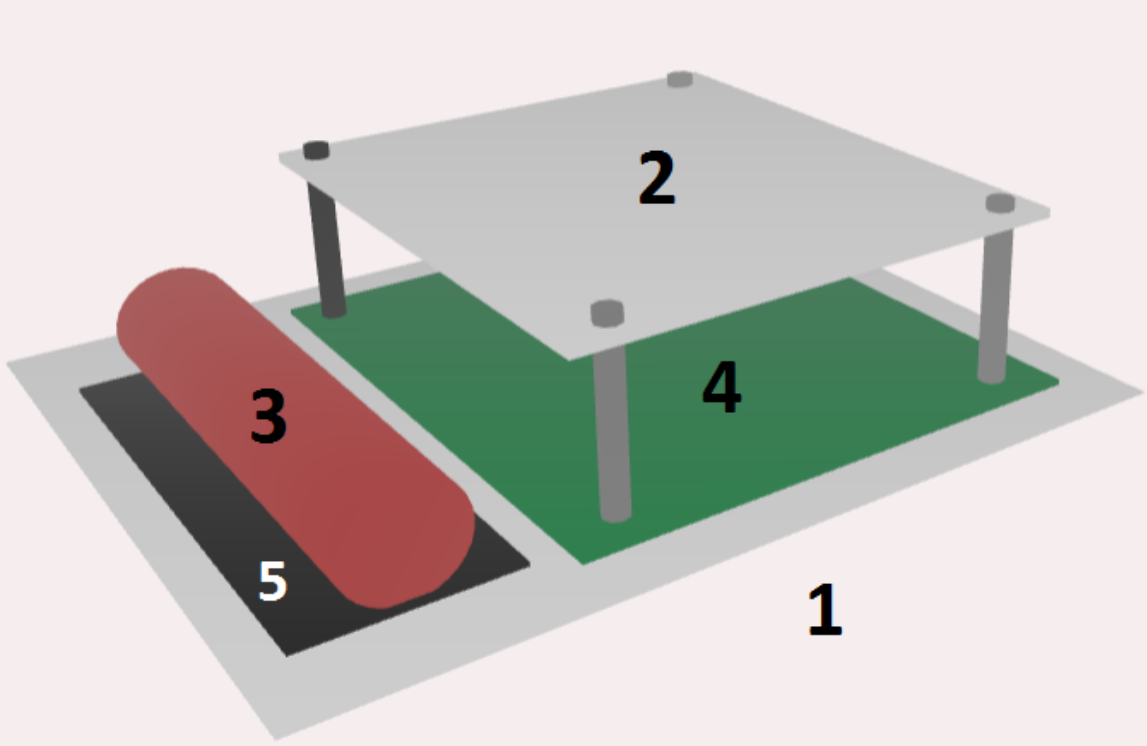
Выбор контроллера ESP32 позволяет развернуть веб-сервер на самой плате. Настроить его работу в асинхронном режиме при помощи библиотеки – «ESPAsyncWebServer». Такой режим работы позволяет обрабатывать множество запросов одновременно без блокировки сервера.

Способ хранения информации в постоянной памяти микроконтроллера, без использования дополнительный SD-карт. Хранение данных в постоянной памяти облегчает их манипулирование и обработку. Также, система сбора

метеоданных не требует хранения больших объемов информации в длительной перспективе, поэтому использование постоянной памяти эффективное и экономное решением. Это позволяет избежать лишних расходов и научиться реализовывать хранения при помощи базы данных.

2.3 Описание внешнего вида системы

Кейс должен обеспечивать доступ к портам и разъемам на плате, быть достаточно прочным. Размер и форма коробки должны соответствовать размерам компоновке устройства. Плата и ее компоненты должны располагаться внутри плотно, удобно, аккуратно. На рисунке 2 изображена 3D-модель системы сбора метеоданных.



1 – нижняя пластина; 2 – верхняя пластина; 3 – аккумуляторная батарея; 4 – макетная плата; 5 – отсек для батареи

Рисунок 2 – 3D-модель системы сбора метеоданных

Внешний вид системы представляет из себя кейс, состоящих из двух прозрачных, акриловых пластин, шириной 3 мм: первая нижняя, основная – размеры 11х13 см, вторая верхняя, дополнительная – размеры 8х11 см. В каждой пластине есть по 4 отверстия, размерами М3, размещаемых в углах, с отступом в 1 см от края с каждой стороны.

Сбор акрилового кейса осуществляется следующим образом – на первую, основную пластину при помощи 4 болтов М3х30 мм, крепится макетная плата с установленными в штыревыми разъемы компонентами, в оставшееся на пластине свободное место приклеивается отсек для аккумулятора. Следом, при помощи 4-ех гаек, фиксируется вторая, верхняя пластина кейса. Проверяется надежность конструкции.

Макетная плата устанавливается внутрь с распаянными под интерфейс I2C дорожками, линиями питания и припаянными штыревыми разъемами.

2.4 План реализации проекта

– выбор необходимого оборудования: выбрать и приобрести плату, датчики, OLED-дисплей, а также другие компоненты, которые могут понадобиться для проекта.

– подготовка среды разработки: установить Arduino IDE и необходимые библиотеки для работы с оборудованием;

– написание кода: написать код для работы с датчиками и дисплеем, настройки Wi-Fi-соединения, взаимодействия с веб-сервером и другими функциями, необходимыми для проекта;

– сборка и тестирование: подключить все компоненты к ESP32 и проверить их работу. Если все работает корректно, то продолжить сборку, если нет, то устранить причину;

– загрузка прошивки: загрузить написанный код в плату;

– тестирование и отладка: протестировать проект и убедиться в его корректной работе. Если есть ошибки или проблемы, исправить их;

– деплой: развернуть веб-сервер и загрузить на него веб-страницы. Протестировать корректность работы сервера, правильность построения и отображения графиков, валидность данных;

– размещение в корпусе: поместить все компоненты в акриловый кейс;

– использование: использовать проект по его назначению, получать информацию о погоде и качестве воздуха в реальном времени;

– обслуживание: следить за состоянием оборудования и заменять изношенные компоненты при необходимости.

2.5 Вывод по второй главе

Во второй главе были выбраны компоненты аппаратной и программной частей:

– используемая плата – ESP32;

– датчик влажности, давления и температуры – BME280;

– датчик-детектор CO2 – MQ135.

– дисплей – OLED-дисплей на базе SSD1306;

– язык программирования – C++;

– среда разработки – Arduino IDE;

– библиотека для работы с базой данных – SQLite3;

– источник питания – аккумуляторная Li-Ion батарея (US18650VTC6);

– модуль зарядного устройства – TP4056;

– wi-fi модуль – встроенный в плату модуль.

Внешний вид: корпус из двух акриловых пластин, скрепленных 4-мя стальными болтами; макетная плата, с припаянными модулями; аккумуляторная батарея с отсеком.

3 Реализация системы

Третья глава – это практический этап реализации, включающий в себя установку необходимых программных компонентов, чтение документации, подключение компонентов, написание кода, настройку и тестирование системы.

3.1 Подключение устройств. Аппаратная часть

Перед подключением к плате датчиков, дисплея и другой периферии, необходимо ознакомиться с выходами на плате ESP32 DevKit v1 и их описанием. Это можно посмотреть на рисунке 3.

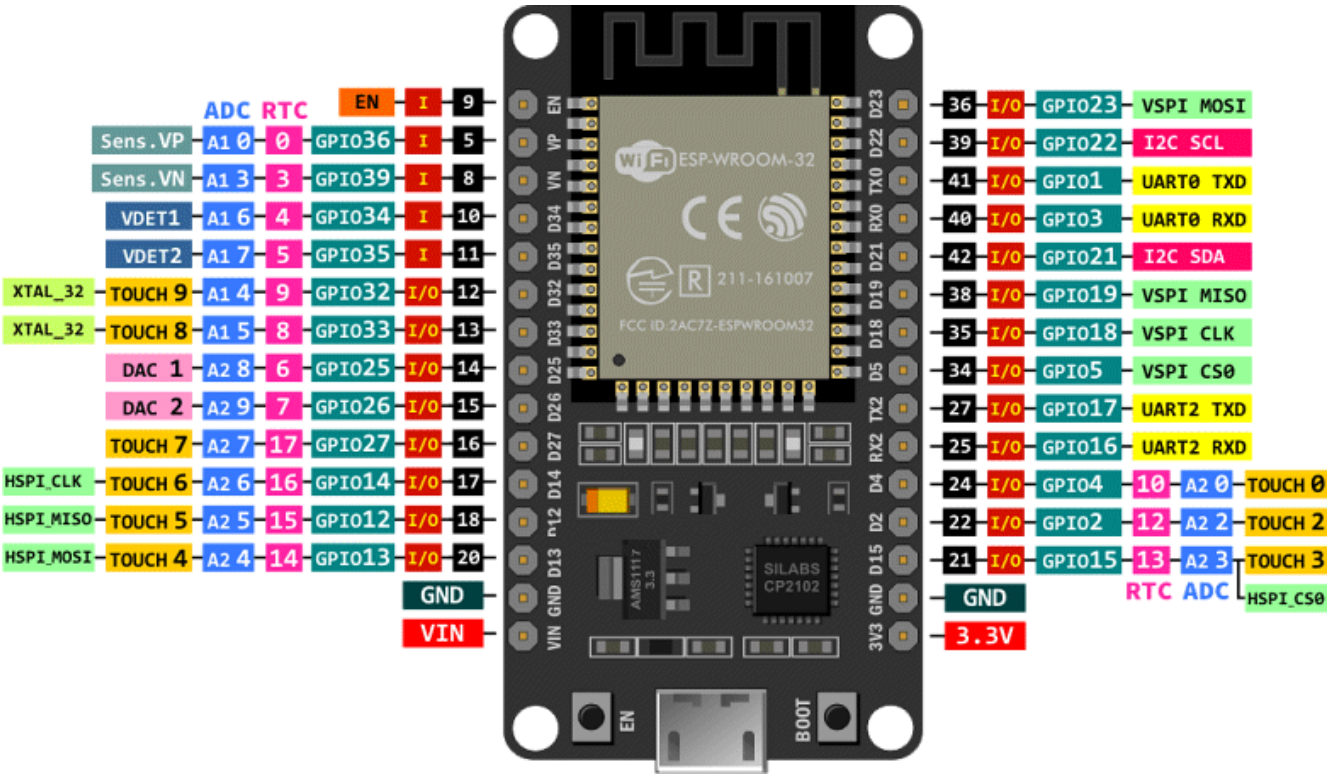


Рисунок 3 – Схема выходов платы ESP32 DevKit v1

GND — общий провод, "земля".

5 .. 42 — номер пина на чипе ESP32

VIN — контакт для подключения внешнего источника питания 5V (стабилизатор AMS1117-3.3 позволяет подавать питание в широком

диапазоне от 5 до 10 V). Хотя стабилизатор допускает подачу более высокого напряжения (до 15 V), но без дополнительного охлаждения возможен перегрев микросхемы.

3.3V — контакт, на который подается выходное напряжение внутрисхемного стабилизатора. Может быть использован для питания подключаемых к плате датчиков.

I/O. I — контакты могут быть использованы только как входы. I/O — контакты могут быть использованы как входы и выходы.

GPIO (General Purpose Interput Output) — контакты ввода/вывода общего назначения. Могут быть сконфигурированы как входы или выходы и программно назначены на различные функции.

EN (Chip Enable) — контакт включения ESP32 в рабочий режим, одновременно может быть задействован для перезапуска контроллера (Reset).

ADC. A1, A2 — выводы встроенного аналого-цифрового преобразователя (АЦП). Входные каналы АЦП имеют разрешение 12 бит. Преобразованные значения лежат в интервале 0 — 4095. Входной диапазон напряжений составляет от 0 до 3,3 В. Есть возможность установить разрешение каналов АЦП в коде, а также диапазон АЦП.

DAC — цифро-аналоговый преобразователь (ЦАП). На ESP32 имеются два 8-битных канала ЦАП для преобразования цифровых сигналов в аналоговые выходные сигналы напряжения.

UART (Universal Asynchronous Receiver-Transmitter) — асинхронный последовательный интерфейс устанавливает связь с другими устройствами по шине UART. Каждая линия может быть переназначена пользователем на любой GPIO.

SPI (Serial Peripheral Interface) — последовательный периферийный интерфейс. ESP32 имеет два SPI (VSPI и HSPI) в ведущем и подчиненном режимах.

TOUCH — контакты ёмкостных сенсорных датчиков. Реагируют не изменение ёмкости в электрической цепи вывода, вызванное прикосновением пальца к соответствующему контакту.

I2C. Интерфейс I2C — последовательная асимметричная шина. I2C используется для подключения датчиков и периферийных устройств.

RTC — ядро низкого энергопотребления.ESP32 имеет сопроцессор с ультранизким энергопотреблением (Ultra Low Power — ULP). Выводы RTC GPIO, перенаправленные в подсистему с низким энергопотреблением, могут использоваться для выхода ESP32 из глубокого сна при работе сопроцессора (ULP). Требуют предварительной программной подготовки.

VDET — аналоговые контакты ядра низкого энергопотребления (RTC). По аналогии с цифровыми контактами, предназначены для вывода процессора ESP32 из режимов энергосбережения. Требуют предварительной программной подготовки.

XTAL_32 — контакты внешнего кварцевого генератора с частотой 32.768 КГц.

Датчик Холла. Sens_VP (positive), Sens_VN (negative) — контакты встроенного датчика Холла, который обнаруживает изменения в магнитном поле в его окружении.

3.1.1 Модуль зарядки TP4056 и Li-Ion батарея

TP4056 - это интегральная схема зарядного устройства для литий-ионных аккумуляторов с одной ячейкой. Она может использоваться для зарядки аккумуляторов различной ёмкости, от 300 мАч до 4000 мАч, и поддерживает зарядку через микро-USB разъем, 5V, 1A. TP4056 имеет встроенный токовый контроллер и термическую защиту. TP4056 автоматически завершает цикл зарядки при достижении напряжения на нем 4.2В и снижении тока заряда до 1/10 от запрограммированной величины. роме контроллера зарядки TP4056 на плате добавлены два чипа DW01 (схема

защиты) и ML8205A (сдвоенный ключ MOSFET), служат для зашиты аккумулятора от переразряда, перезаряда, перегрузки и короткого замыкания.

На рисунке 4 можно увидеть устройство модуля зарядки.

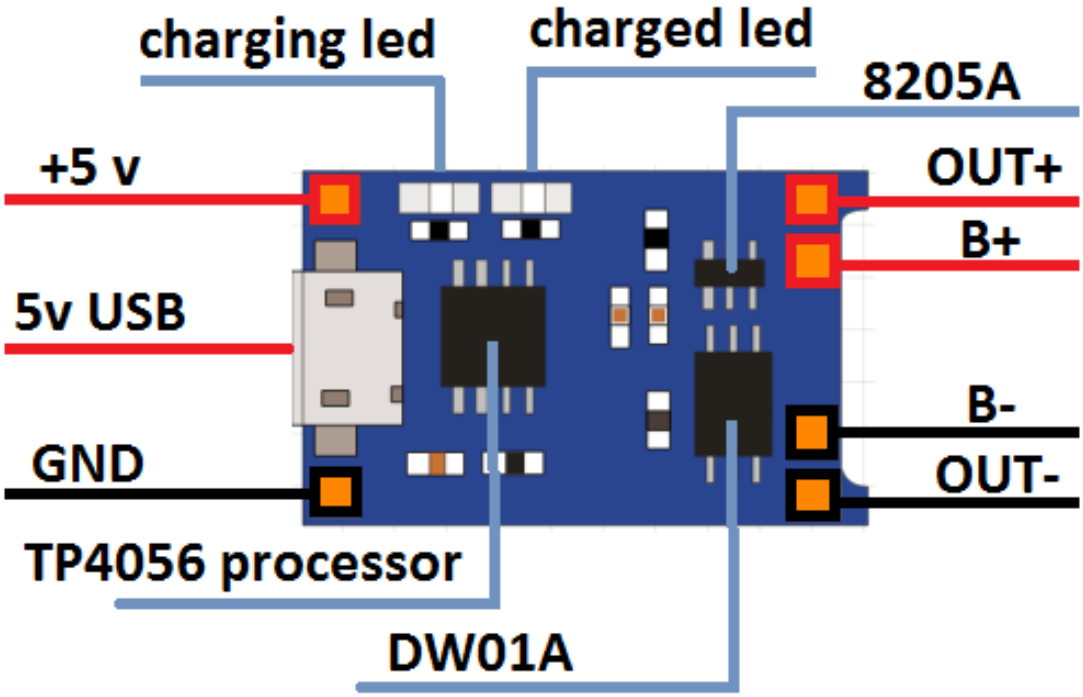


Рисунок 4 – Устройство модуля зарядки TP4056

OUT+ – нагрузка «+», OUT- – нагрузка «-», B+ – к «+» аккумулятора, B- – к «-» аккумулятора, «charging led» – индикация зарядки, «charged led» – индикация заряда. «+5v» и «GND» - для подключения внешнего источника питания. На рисунке 5 показана схема подключения системы энергопитания.

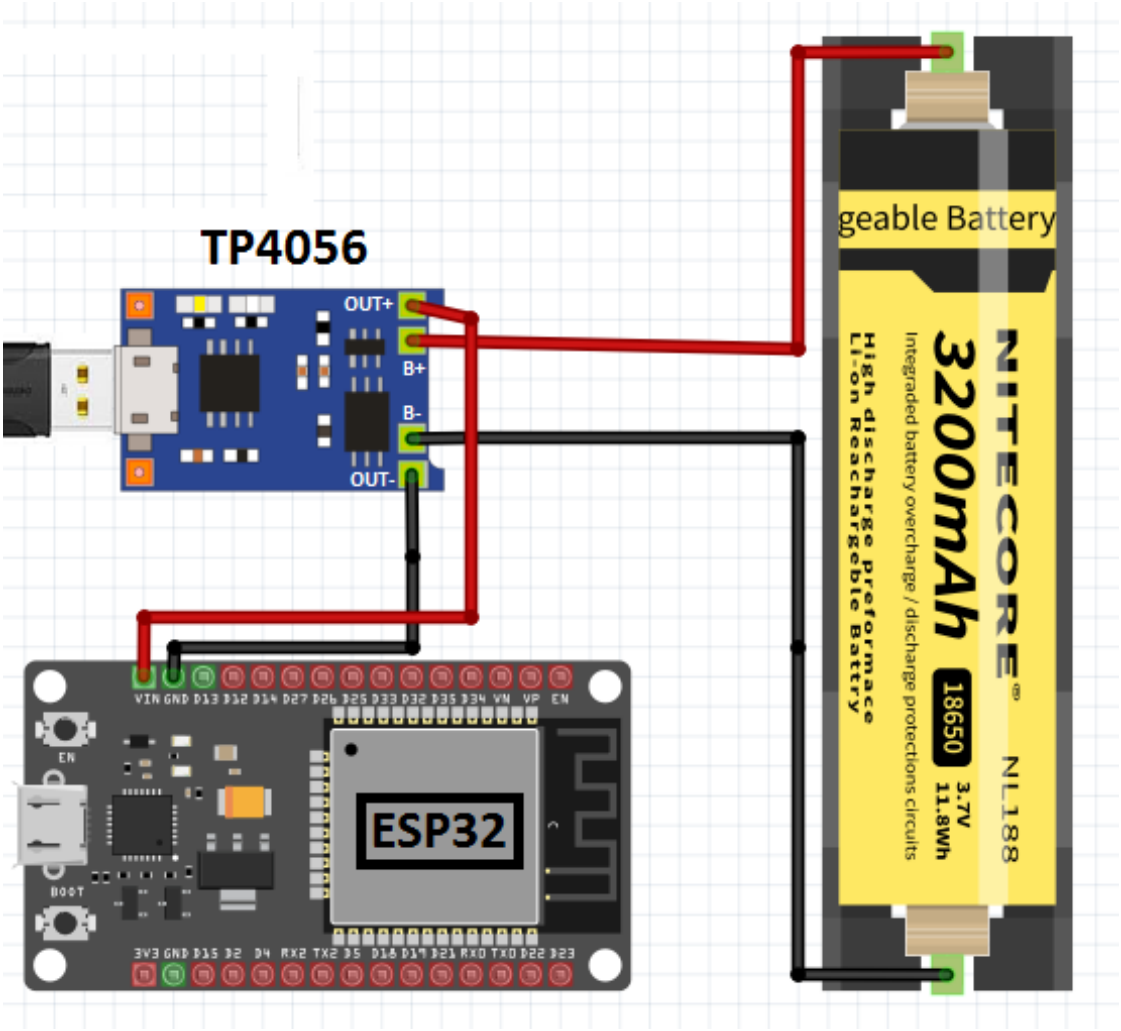


Рисунок 5 – Схема подключения системы питания к плате ESP32

В пины «VIN» и «GND» платы ESP32 подключаются выходы «OUT+» и «OUT-» соответственно.

3.1.2 Датчик BME280 и OLED-дисплей

Датчик представляет собой модуль, являющийся печатной платой небольшого размера с расположенным на ней датчиком BME280 и четырьмя выходами: питание, земля, SCL, SDA. Представлен на рисунке 6.

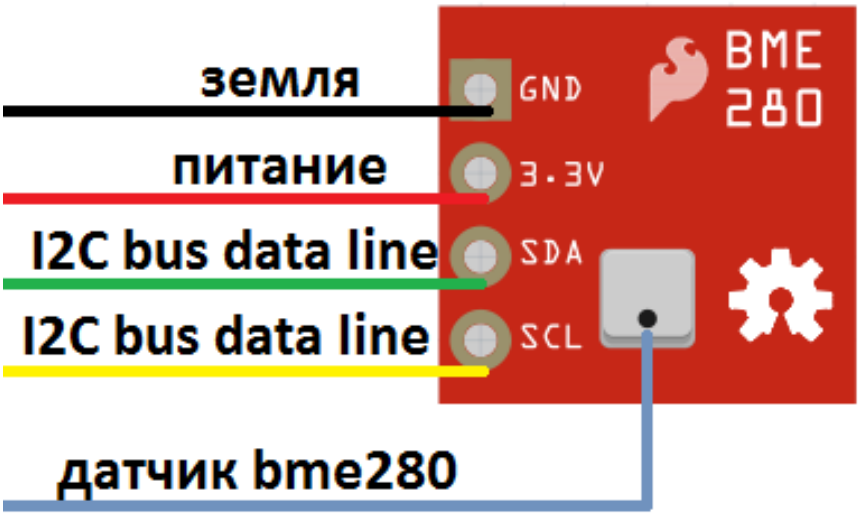


Рисунок 6 – Устройство модуля с I2C-интерфейсом и датчиком BME280

Технические характеристики модуля: интерфейс: SPI, I2C; напряжение питания: 3.3В; диапазон измерений давления: 300-1100hPa; диапазон измерений температуры: -40 - +85 °C; диапазон измерений влажности: 0 - 100 %; энергопотребление: режим измерений - 2.74 нА; в спящий режим: - 0.1 нА. Точность измерений: давление - 0.01 hPa; температура - 0.01° C; влажность – 3%.

OLED-дисплей с голубым цветом текста, размерами 0,91”, разрешением 128x32 пикселя, подключаемого по шине I2C, выполненного на базе однокристального драйвера матричных индикаторов SSD1306. На рисунке 7 представлено устройство модуля с OLED-дисплеем.

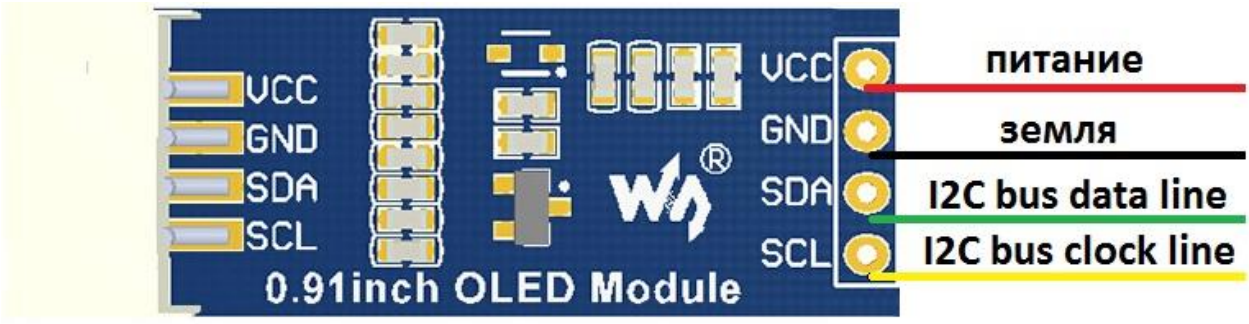


Рисунок 7 – Устройство OLED-дисплея

Необходимо показать способ подключения OLED-дисплея к плате ESP32. Это подключение отображено на рисунке 8.

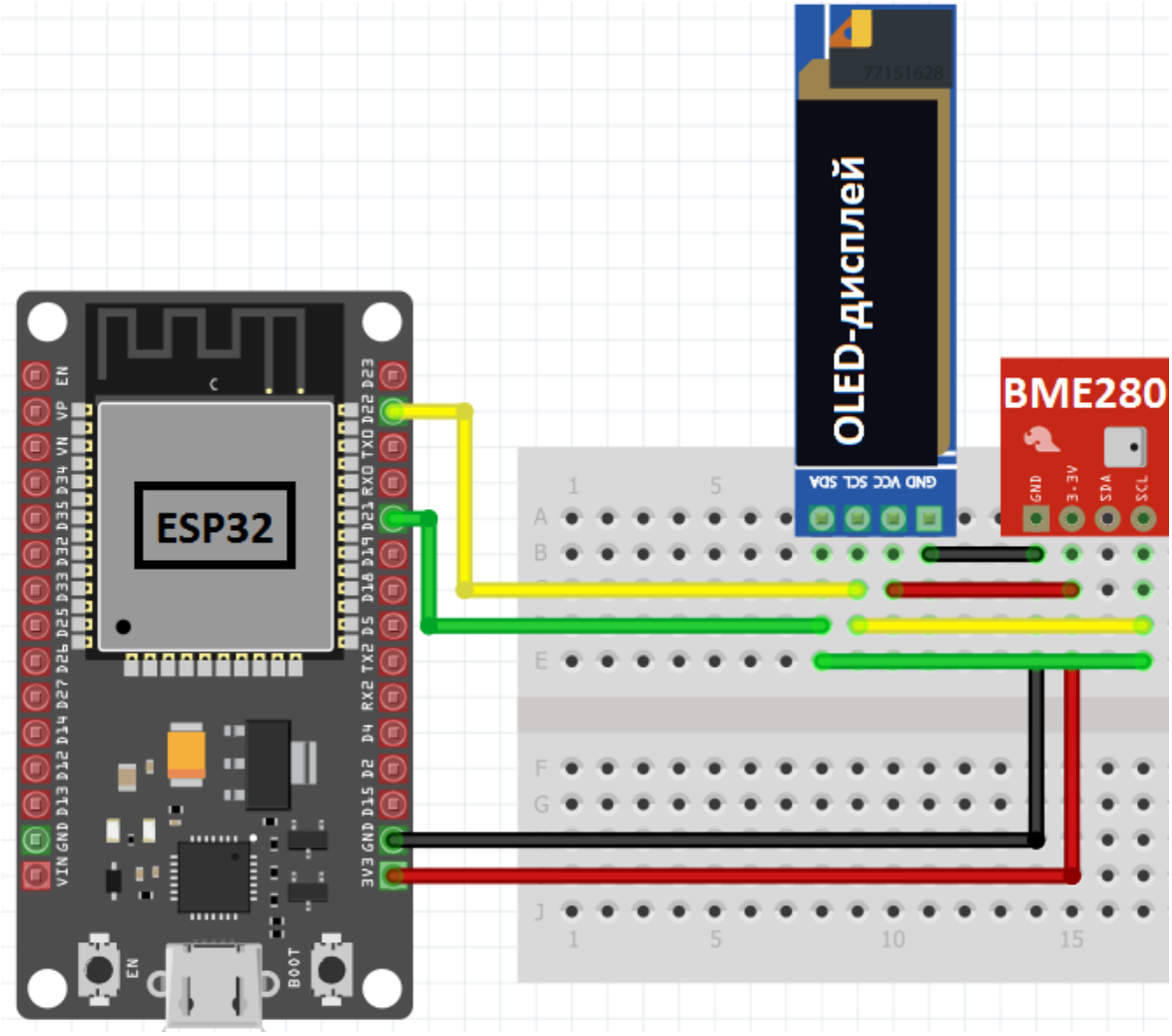


Рисунок 8 – Схема подключения датчика BME280 и OLED-дисплея к плате ESP32

На схеме изображена плата ESP32 и датчик BME280. Они соединены 4-мя проводами: красным – питание (3.3 В), черным – земля, зеленым – SDA (данные, D21), желтым – SCL (тактовые импульсы, D22). D21 и D22 это выходы с платы, которые согласно схеме выходов работают по интерфейсу I2C.

3.1.3 Датчик MQ135 и RGB-светодиод

Датчик представляет собой модуль, с расположенным на нем газовым анализатором MQ135, 4 выходами: питание, земля, DO (digital output – выход с компаратора, в виде нуля или единицы), AO (analog output – в виде числового значения), компаратором LM393, потенциометром, LED-индикацией питания (LED power) и выходом компаратора (LED DO). (Рисунок 9).

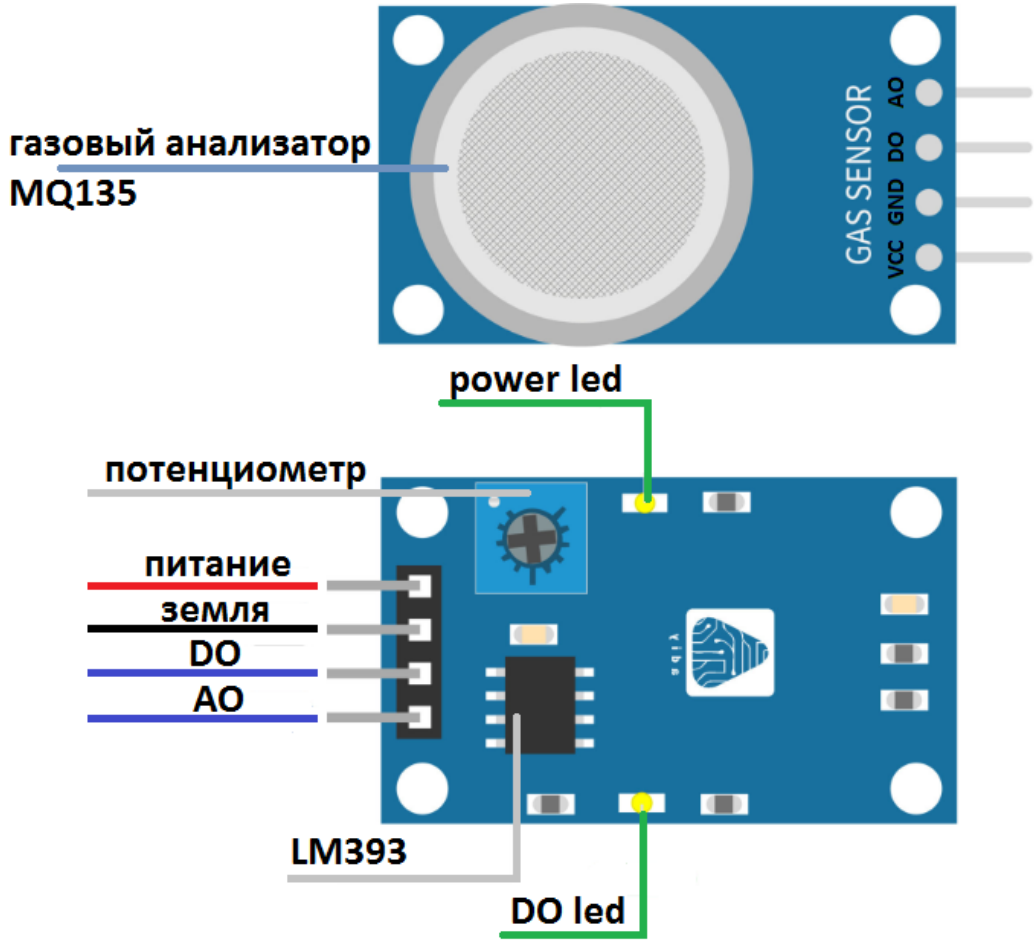


Рисунок 9 – Устройство модуля с газовым анализатором MQ135

Технические характеристики модуля: напряжение питания нагревателя: 5 В, напряжение питания датчика: 3,3–5 В; время прогрева при включении: 1 мин; потребляемый ток: 150мА; диапазон измерений: аммиак: 10—300 ppm; бензин: 10—1000 ppm; алкоголь: 10—300 ppm. Схема подключения на рисунке 10.

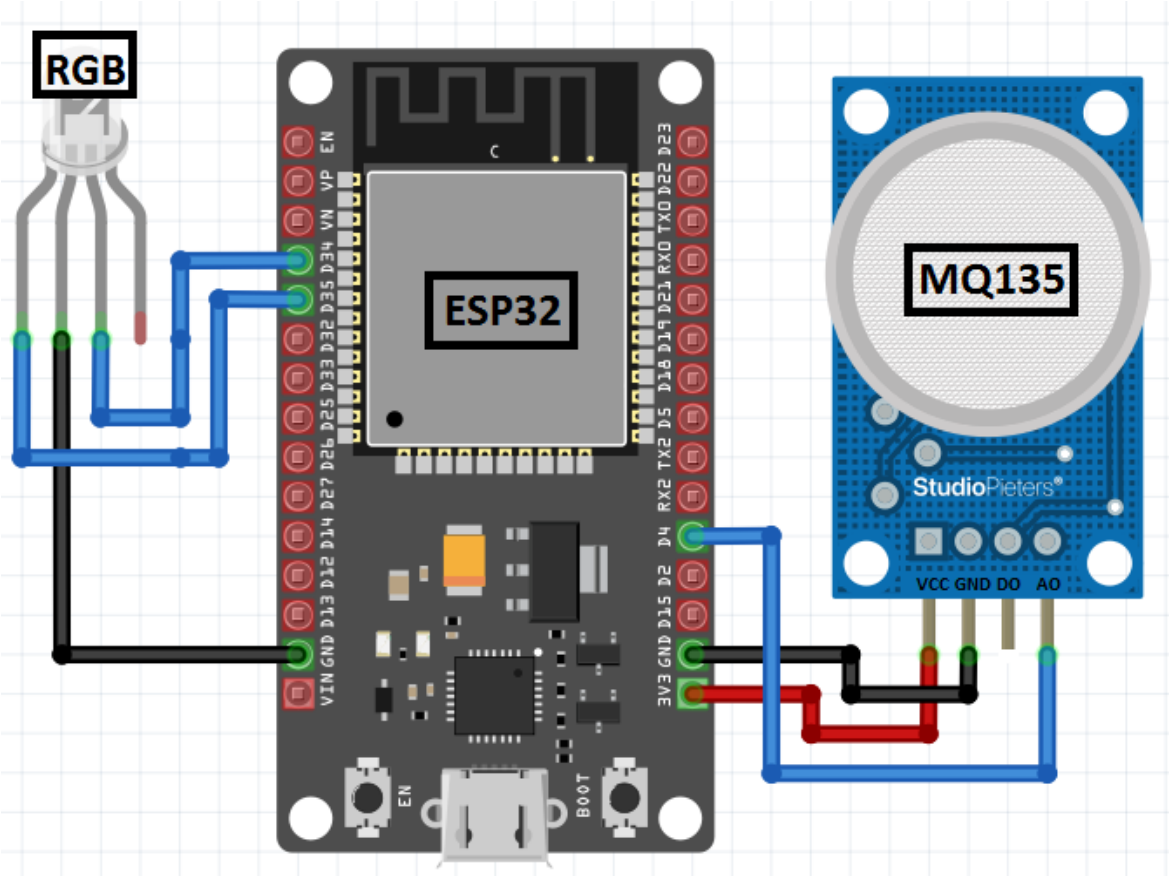


Рисунок 10 – Схема подключения датчика MQ135 и RGB-светодиода к плате ESP32

RGB-светодиод с общим катодом подключается к земле и к пинам D34 и D35. С этих пинов происходит управление красной и зеленой составляющей светодиода. Нога, отвечающая за синий цвет, не подключается, поскольку в работе этот цвет не участвует. Датчик MQ135 питается выходами с платы: красный провод – питание 3,3V; черный – земля. Аналоговый выход с датчика подключен к пину D4 с АЦП.

3.1.4 Общая схема подключения

На рисунке 11 представлена общая схема подключения.

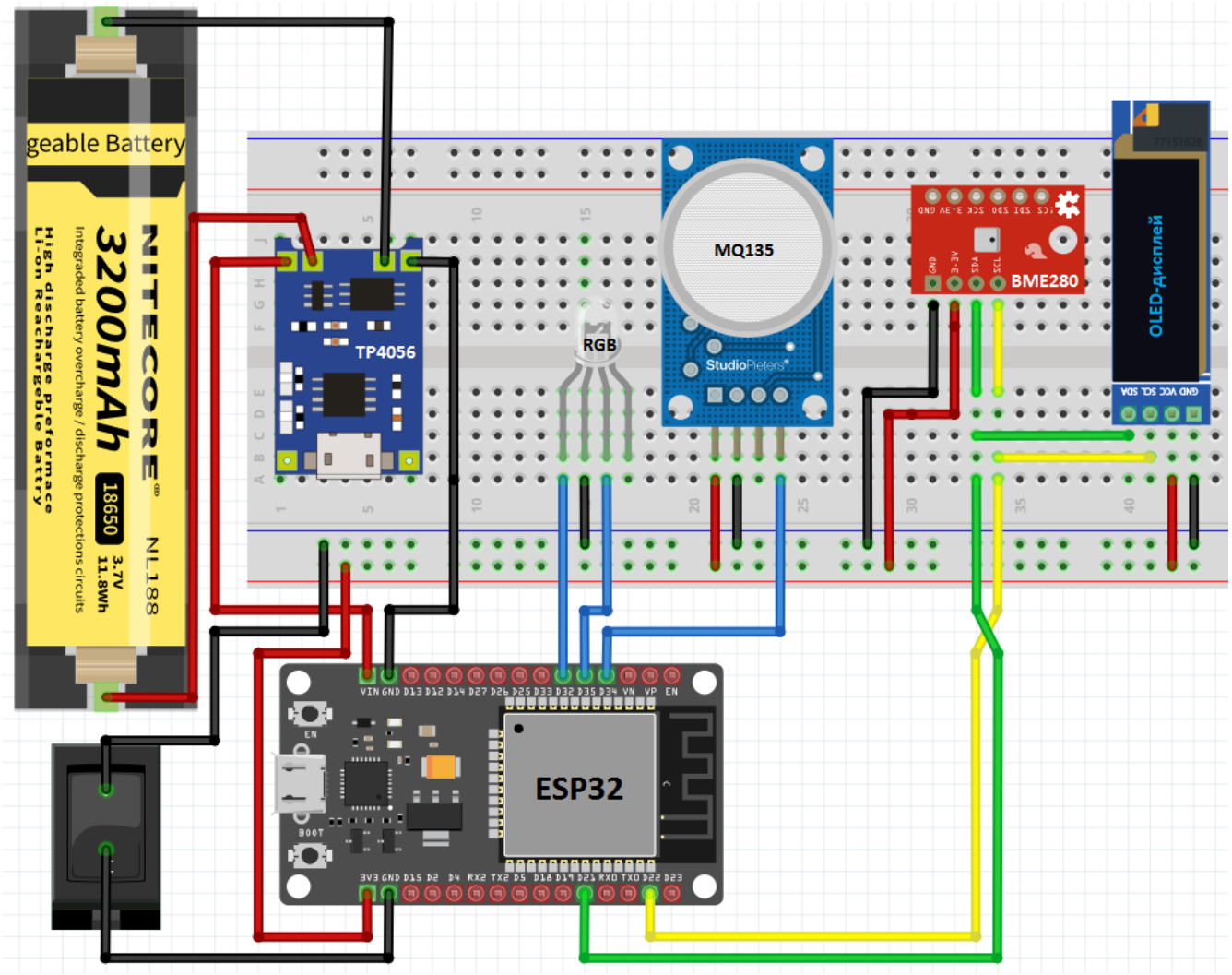


Рисунок 11 – Общая схема подключения компонентов системы сбора метеоданных

Все компоненты соединены проводами разных цветов: красный – питание; черный – земля; желтый – SCL; зеленый – SDA, синий – для остальных случаев. Все элементы располагаются на макетной плате, называемой breadboard. Кнопка переключает режимы питания – от батареи или от кабеля через разъем micro-USB. На плату загружается код-прошивка, что позволяет управлять всеми компонентами: для дисплея и BME280 – это интерфейс I2C, для работы которого необходимо подключить устройства к

выводам D22 -SCL и D21 – SDA. RGB-светодиод подключен также, как рассматривалось выше. Датчик MQ135 опрашивается с пина D34, имеющем аналого-цифровой преобразователь.

3.1.5 Перенос схемы на печатную плату

Общая схема подключения была адаптирована под необходимые условия, а именно, перенесена на печатную плату, размерами 7х9 см. Изменилось только расположение компонентов – оно стало компактнее (Рисунок 12).

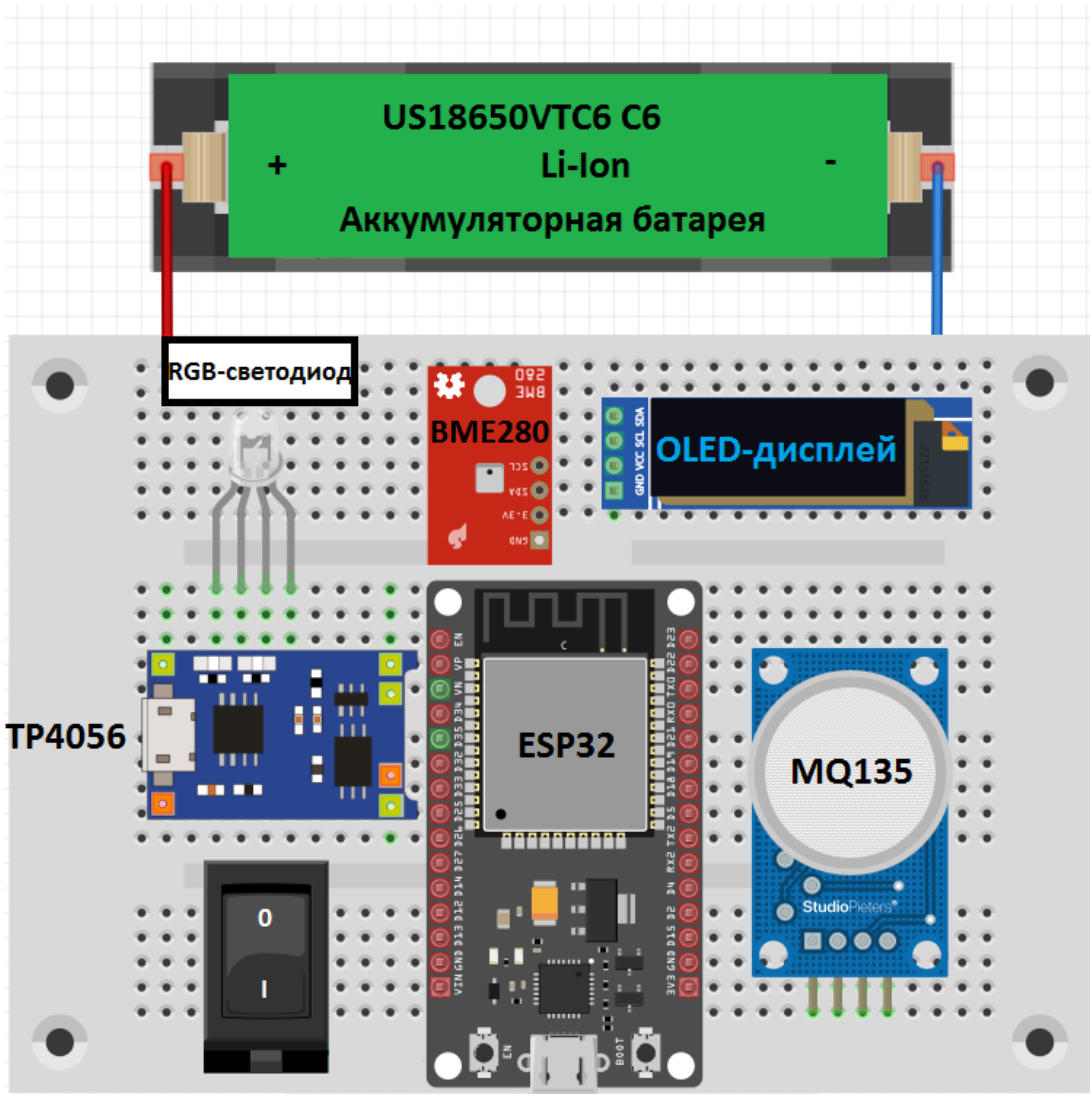


Рисунок 12 – Расположение элементов на печатной плате

3.2 Написание кода. Программная часть

В данной части проекта написан, проверен, протестирован код, добавлены комментарии, спроектирована, показана и объяснена файловая структура проекта. Рассмотрено содержимое отдельных файлов проекта, нуждающихся в дополнительном разъяснении. На рисунке 13 показана файловая структура проекта.

3.2.1 Файловая структура проекта

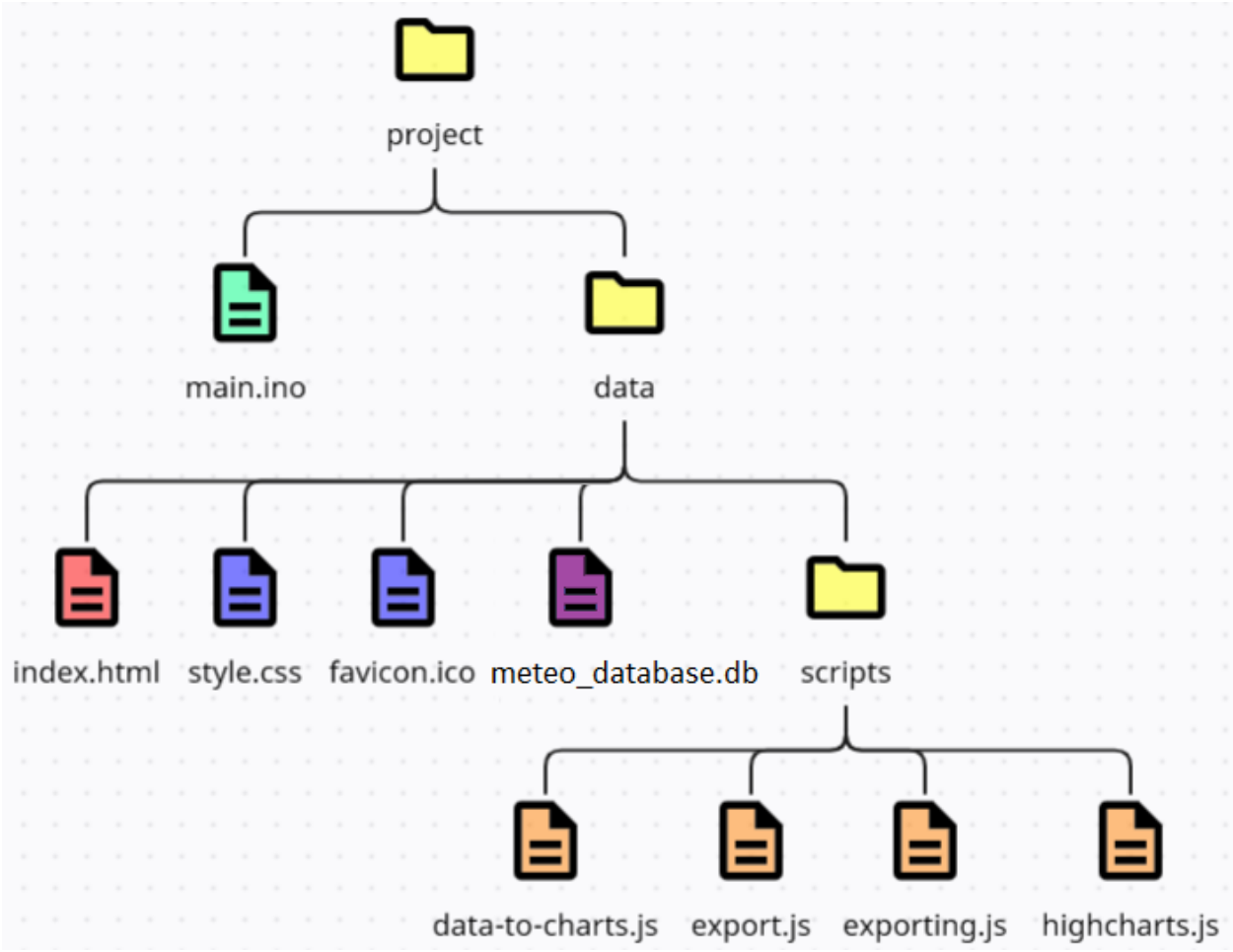


Рисунок 13 – Файловая структура проекта

3.2.1.1 Директория «projects»

Эта директория является основной директорией всего проекта, в ней находятся: главный файл проекта – «main.ino» – скетч-прошивка, загружаемая в саму плату. Написана в Arduino IDE, на языке C++; директория «data» – корневой каталог файловой системы SPIFFS. В этой директории находятся файлы, которые при загрузке скетча попадают во FLASH-память устройства и хранятся там. Эти же файлы использует веб-сервер.

3.2.1.2 Директория «data»

В области этой директории находятся файлы, которые использует веб-сервер при своей работе, а именно: «index.html» – файл разметки страницы; «style.css» – файл со всеми стилями; «favicon.ico» – это небольшой значок для показа на вкладке; «scripts» – каталог, содержащий файлы с расширением .js; «meteo_database.db» – файл базы данных, хранящий таблицу с данными.

3.2.1.3 Директория «scripts»

Включает в себя 4 файла: «data-to-charts.js» – скрипт, строит графики на странице; «highcharts.js» – библиотека, позволяет строить графики; «export.js» и «exporting.js» – дополнения к библиотеке для экспорта данных.

3.2.2 Содержимое файлов проекта

Полное содержимое рассматриваемых файлов представлено в приложениях к выпускной квалификационной работы, за исключением файлов – «highcharts.js», «export.js», «exporting.js», поскольку это файлы библиотек, они являются общедоступными и имеют документацию.

В этом разделе рассматриваются определенные моменты в коде, которые нуждаются в дополнительном объяснении.

3.2.2.1 Содержимое файла «main.ino»

Функция – «void setup(){…}» (Приложение А Листинг кода «main.ino», строка 46) выполняется один раз при запуске Arduino и используется для настройки пинов, подключения к Wi-Fi, настройки сервера и обработчиков запросов.

Строки номер 80-82 (Приложение А Листинг кода «main.ino», строки 80-82) устанавливают обработчик запроса для корневого пути ("/") при HTTP-методе GET. При получении такого запроса, сервер отправляет файл "index.html" из файловой системы SPIFFS. Аналогичным образом, остальные строки «server.on()» устанавливают обработчики запросов для различных файлов, таких как стили CSS и скрипты JavaScript, которые будут использоваться веб-интерфейсом.

Функция – «void loop» (Приложение А Листинг кода «main.ino», строка 115), выполняется в бесконечном цикле после выполнения функции setup(). Внутри цикла происходит считывание значений с датчиков, вывод значений на дисплей и управление светодиодами в зависимости от показаний датчика MQ135.

Внутри этой функции содержится следующий фрагмент, который стоит объяснить (Приложение А Листинг кода «main.ino», строки 129-154). Этот блок кода отвечает за управление светодиодами в зависимости от показаний датчика MQ135. Сначала считывается значение ppm (частиц в миллионе) с датчика MQ135. Затем проверяется, находится ли значение ppm выше порога RED_THRESHOLD. Если да, то красный светодиод устанавливается в максимальную яркость (255), а зеленый светодиод выключается (0). Если значение ppm ниже порога GREEN_THRESHOLD, то красный светодиод выключается (0), а зеленый светодиод устанавливается в максимальную

яркость (255). Если значение ppm находится между GREEN_THRESHOLD и RED_THRESHOLD, то происходит промежуточное управление яркостью светодиодов в зависимости от значения ppm с использованием функции map(). Функция map() используется для пересчёта значения переменной из одного диапазона в другой, позволяет линейно масштабировать значения переменных.

3.2.2.2 Содержимое файла «data-to-charts.js»

Файл "data-to-charts.js" отвечает за обновление данных в графиках на веб-странице. Он содержит код, который периодически отправляет HTTP-запросы на сервер, получает новые данные о температуре, влажности и давлении, а затем обновляет соответствующие графики на странице с помощью библиотеки «Highcharts». В общем, файл «data-to-charts.js» играет роль посредника между сервером, который предоставляет данные о метеопараметрах, и веб-страницей, отображающей эти данные в виде графиков. Он обеспечивает автоматическое обновление графиков с новыми данными без необходимости перезагрузки страницы.

Рассмотрим пример кода из приложения Г Листинг файла «data-to-charts.js», строки 55-59. Данный код выполняет периодический AJAX (Asynchronous JavaScript and XML) запрос на сервер для получения данных о температуре с URL “/temperature” и обновляет график с использованием полученных данных.

Что касается строчек 46-49 (Приложение Г Листинг файла «data-to-charts.js», строки 46-49) В них происходит добавление новой точки данных [x, y] на график chartT, перерисовывается график, удаляется самая старая точка при необходимости и добавляется анимацию при добавлении точки. Первое «true» – это параметр «redraw», который указывает, нужно ли перерисовывать график после добавления новой точки. Второе «true» – это параметр «shift», который определяет, нужно ли удалять самую старую точку

данных на графике, если количество точек превышает определенный предел. Третье «true» – это параметр «animation», который определяет, должна ли быть анимация при добавлении новой точки на графике.

Аналогичные блоки кода используются для графиков влажности и давления (chartH и chartP), с небольшими отличиями в идентификаторах элементов, тексте заголовков и цветах линий графиков. Эти блоки кода выполняются с интервалом в 1 секунду и отправляют GET-запросы на соответствующие адреса ("/humidity" и "/pressure"). Полученные значения добавляются в соответствующие серии данных графиков.

3.2.2.3 Назначение файла «meteo_database.db»

Файл необходим для хранения данных, получаемых с датчиков. Данные записываются в таблицу «meteo_table».

База данных реализована на sqlite. Информация с датчиков, подключенных к плате, поступает на плату в формате float. В базе данных хранятся: id-значения и сами значения по трем столбцам - значения температуры, влажности и давления (Рисунок 14).

meteo_table		
KEY	**NAME**	**TYPE**
PK	id	INT PRIMARY KEY AUTO_INCREMENT
	temperature_value	REAL
	humidity_value	REAL
	pressure_value	REAL

Рисунок 14 – Логическая модель базы данных

3.3 Вывод по третьей главе

В третьей главе была реализована цель выпускной квалификационной работы, создана динамическая система отображения данных с датчиков на графиках в реальном времени, использующая библиотеку «Highcharts».

Это позволяет наглядно отслеживать и анализировать данные с датчиков и обеспечивает более полное понимание происходящих процессов и тенденций. Также в ходе работы была произведена сборка проекта и загрузка программного обеспечения в микроконтроллер. Далее созданный проект прошел стадию тестирования, в ходе которой было выявлено, что созданная система полностью выполняет возложенные на нее задачи.

ЗАКЛЮЧЕНИЕ

В заключении стоит отметить навыки, потребовавшиеся при выполнении работы, а именно - какими умениями, знаниями и технологиями необходимо владеть, для успешной реализации проекта:

а) по аппаратной части:

– базовые знания по электротехнике, электронике и схемотехнике;

– опыт в пайке электронных устройств – для установки всех модулей системы на макетной плате;

– знание протокола коммуникации – I2C, для связи с датчиками и другими устройствами;

– знание особенностей платы ESP32 и ее функциональных возможностей;

– умение работать с макетной платой и электронными компонентами, включая подключение модулей и датчиков к ESP32;

– понимание основных принципов работы датчиков.

б) по программной части:

– умение настроить окружение для работы и выбрать подходящий язык;

– основы программирования на языке C++ – переменные, функции, условные операторы и циклы;

– навыки работы со средой для программирования – Arduino IDE;

– знание и умение работать с требуемыми библиотеками: <WiFi.h>, <ESPAsyncWebServer.h>, <SPIFFS.h>, <Adafruit_BME280.h>, <Adafruit_SSD1306.h>, <MQ135.h>, <SQLite3.h>;

– основы веб-разработки: HTML для разметки страницы, CSS для стилизации и JavaScript для интерактивности и отображения данных;

– ознакомиться с использованием файловой системы SPIFFS;

– основы сетевых технологий – понимание работы сетей и сетевых протоколов;

– основы сетевого программирования, такие как работа с HTTP-запросами и ответами;

– знания пожарной безопасности – в процессе пайки.

Помимо этого, для успешной реализации проекта может потребоваться умение анализировать и отлаживать программный код, умение проводить тестирование и отладку системы, а также способность и желание изучать новые технологии и решать возникающие проблемы.

Расчет стоимости компонентов.

Стоимость всех компонентов, а также цена их доставки указы в таблице 5.

Таблица 5 – Расчет стоимости компонентов проекта

Наименование	**Количество**	**Цена, руб**	**Сумма, руб**
Модуль зарядки TP4056, LiIo/LiPo, с защитой, microUSB	1	36	36
Разъем штыревой, 1x40 контактов, 2.54 мм, папа	1	13	13
Разъем штыревой, 1x40 контактов, 2.54 мм, мама	3	23	69
Печатная плата, 7x9 см, двусторонняя	1	85	85
Дисплей OLED, 0.91 дюйм, I2C, монохромный голубой	1	304	304
Датчик газа MQ135	1	199	199
Отсек для аккумуляторов 18650	1	46	46
Контроллер ESP32, с модулем CH9102	1	562	562
Модуль с RGB-светодиодом	1	33	33
Выключатель	1	17	17
Датчик BME280	1	500	500
Болт стальной, M3x30 мм, 10 шт	1	53	53
Гайка стальная, M3, 50 шт	1	61	61
Монтажный провод папа-мама, 10 см, 40 шт	1	99	99
Монтажный провод мама-мама, 10 см, 40 шт	1	99	92
Оргстекло прозрачное, 3 мм, 20x30 см	1	270	270
Организация доставки	2	310	620
			ИТОГО: 3059

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Википедия – свободная энциклопедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Метеостанция

2 cnx-soft: новости, инструкции, обзоры встраеваемых систем [Электронный ресурс] – Режим доступа: https://cnx-software.ru/2020/03/25/znajte-razlichiya-mezhdu-raspberry-pi-arduino-i-esp8266-esp32/

3 Генератор значков для сайта [Электронный ресурс] – Режим доступа: https://favicon.io/favicon-generator/

4 Справочник по HTML [Электронный ресурс] – Режим доступа: http://htmlbook.ru/

5 Вольтик.ру – интернет-магазин diy-электроники [Электронный ресурс] – Режим доступа: https://voltiq.ru/esp32-web-server-with-bme280-mini-weather-station/

6 Онлайн справочник по микроконтроллерам [Электронный ресурс] – Режим доступа: https://wikihandbk.com/wiki/ESP32:Примеры/

7 КМПУ – портал с теорией по модульному проектированию [Электронный ресурс] – Режим доступа: http://kmpu.ru/other/esp32_01_introduction/index.html

8 r2ino.ru – статьи и уроки по программированию [Электронный ресурс] – Режим доступа: https://r2ino.ru/blog/uroki-programmirovaniya/

9 Электронный портал, каталог проектов [Электронный ресурс] – Режим доступа: https://randomnerdtutorials.com/esp32-plot-readings-charts-multiple/

10 myrobot.ru [Электронный ресурс] – Режим доступа: https://myrobot.ru/wiki/index.php?n=Experiences.Esp32Pinout

11 amperka.ru [Электронный ресурс] – Режим доступа: http://wiki.amperka.ru/products:esp32-wroom-wifi-devkit-v1

ПРИЛОЖЕНИЕ А

Листинг кода «main.ino»

```
1  // Подключение библиотек
2  #include <Arduino.h>
3  #include <WiFi.h>
4  #include <ESPAsyncWebServer.h>
5  #include <SPIFFS.h>
6  #include <Adafruit_BME280.h>
7  #include <Adafruit_SSD1306.h>
8  #include <sqlite3.h>
9  #define SEALEVELPRESSURE_HPA (1017) //Давление над уровнем
   моря в hPa
10 // Номера портов светодиодов
11 const int RED_PIN = 32;
12 const int GREEN_PIN = 33;

13 // Установка свойств ШИМ-сигнала
14 const int frequency = 5000; // Частота
15 const int resolution = 8; // Разрешение = 256
16 const int redChannel = 0; // Номер красного канала
17 const int greenChannel = 1; // Номер зеленого канала

18 // Настройки точки доступа
19 const char* ssid = "meteo"; // Имя точки доступа

20 AsyncWebServer server(80); // Присваивание экземпляру
   класса имени и назначение порт
21 Adafruit_BME280 bme; // Присваивание экземпляру класса
   имени
22 Adafruit_SSD1306 display(128, 32); // Передача размеров
   дисплея экземпляру класса

23 IPAddress ip(192, 168, 0, 20); // Локальный статический IP-
   адрес
24 IPAddress gateway(192, 168, 0, 20); // IP-адрес шлюза
25 IPAddress subnet(255, 255, 255, 0); // Маска подсети

26 // Функция отображения значений температуры, влажности и
   давления на OLED-дисплее
27 void values_on_display(float temperature, float humidity,
   float pressure, float altitude){
28 display.setTextColor(WHITE);             // Установка цвета
   текста
29 display.clearDisplay();                  // Очистка дисплея
30 display.setCursor(0,0);                   // Установка
   позиции курсора
31 display.print("Temperature: ");                 // Вывод
   метки температуры
32 display.print(temperature);       // Вывод температуры
```

```
33 display.println(" C");                    // Добавление
   единиц измерения и переход на следующую строку
34 display.print("Himidity: ");                  // Вывод
   метки влажности
35 display.print(humidity);          // Вывод влажности
36 display.println(" %");                    // Добавление
   единиц измерения и переход на следующую строку
37 display.print("Pressure: ");                // Вывод метки
   давления
38 display.print(pressure / 100);                 // Вывод
   давления
39 display.println(" hPa");                  // Добавление
   единиц измерения и переход на следующую строку
40 display.print("Altitude: ");     // Вывод метки уровня
41 display.print(round(altitude));          // Вывод уровня
   над морем
42 display.println(" m");                    // Добавление
   единиц измерения и переход на следующую строку
43 display.display();                        // Вывод на
   дисплей данных из оперативной памяти
44 }
45 // Функция setup, которая выполняется один раз при запуске
46 void setup(){
47 // Настраиваем каналы
48 ledcSetup(redChannel, frequency, resolution);
49 ledcSetup(greenChannel, frequency, resolution);

50 // Привязываем порты к каналам
51 ledcAttachPin(RED_PIN, redChannel);
52 ledcAttachPin(GREEN_PIN, greenChannel);

53 bme.begin(0x76); // Инициализация датчика по его адресу
54 display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // Инициализация
   дисплея
55 delay(500); // Задержка
56 SPIFFS.begin(); // Инициализация файловой системы

57 // Открываем файл
58 File file  = SPIFFS.open("meteo_database.db");
59 // Получаем размер файла
60 size_t fileSize = file.size();
61 // Закрываем файл
62 file.close();

63 // Открытие соединения с базой данных
64 sqlite3* db;
65 sqlite3_open("/spiffs/meteo_database.db", &db);

66 // Удаляем все записи в таблице, если размер файла
   превышает 1 МБ (1048576 байт)
67 if (fileSize > 1048576) {
68 sqlite3_exec(db, "DELETE FROM meteo_table WHERE value_id >
   10;", 0, 0, 0);
```

```
69 }
70 delay(500); // Задержка

71 // Применение настроек Access Point
72 WiFi.softAP(ssid);
73 WiFi.softAPConfig(ip, gateway, subnet);
74 delay(500);

75 /*
76 Обработчик запроса для корневого пути,
77 который вызывает лямбда-функцию request и отправляет
   index.html
78 при обращении клиента к корневому каталогу файловой системы
79 */
80 server.on("/", HTTP_GET, [](AsyncWebServerRequest
   *request){
81 request->send(SPIFFS, "/index.html");
82 });
83 server.on("/favicon.ico", HTTP_GET,
   [](AsyncWebServerRequest *request){
84 request->send(SPIFFS, "/favicon.ico", "image/x-icon");
85 });
86 server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest
   *request){
87 request->send(SPIFFS, "/style.css", "text/css");
88 });
89 server.on("/scripts/data-to-charts.js", HTTP_GET,
   [](AsyncWebServerRequest *request){
90 request->send(SPIFFS, "/scripts/data-to-charts.js",
   "application/javascript");
91 });
92 server.on("/scripts/highcharts.js", HTTP_GET,
   [](AsyncWebServerRequest *request){
93 request->send(SPIFFS, "/scripts/highcharts.js",
   "application/javascript");
94 });
95 server.on("/scripts/exporting.js", HTTP_GET,
   [](AsyncWebServerRequest *request){
96 request->send(SPIFFS, "/scripts/exporting.js",
   "application/javascript");
97 });
98 server.on("/scripts/export-data.js", HTTP_GET,
   [](AsyncWebServerRequest *request){
99 request->send(SPIFFS, "/scripts/export-data.js",
   "application/javascript");
100     });
101     server.on("/temperature", HTTP_GET,
   [](AsyncWebServerRequest *request){
102     request->send_P(200, "text/plain",
   String(sqlite3_exec(db, "SELECT temperature_value FROM
   meteo_table ORDER BY value_id DESC LIMIT 1;", 0, 0,
   0)).c_str());
103     });
```

```
104     server.on("/humidity", HTTP_GET,
  [](AsyncWebServerRequest *request){
105     request->send_P(200, "text/plain",
  String(sqlite3_exec(db, "SELECT humidity_value FROM
  meteo_table ORDER BY value_id DESC LIMIT 1;", 0, 0,
  0)).c_str());
106     });
107     server.on("/pressure", HTTP_GET,
  [](AsyncWebServerRequest *request){
108     request->send_P(200, "text/plain",
  String(sqlite3_exec(db, "SELECT pressure_value FROM
  meteo_table ORDER BY value_id DESC LIMIT 1;", 0, 0,
  0)).c_str());
109     });
110     // Закрытие базы данных
111     sqlite3_close(db);
112     server.begin(); // Запуск сервера
113     delay(1000);
114     }

115     void loop(){ // Функция loop() выполняется в
  бесконечном цикле и содержит основную логику программы
116     float temperature = bme.readTemperature(); // Считываем
  температуру
117     float humidity = bme.readHumidity(); // Считываем
  влажность
118     float pressure = bme.readPressure(); // Считываем
  давление
119     float altitude =
  bme.readAltitude(SEALEVELPRESSURE_HPA); // Получаем высоту
  над уровнем моря

120     // Открытие соединения с базой данных
121     sqlite3* db;
122     sqlite3_open("/spiffs/meteo_database.db", &db);

123     // Формирование SQL-запроса для вставки данных в
  таблицу meteo_table
124     char insertQuery[200];
125     sprintf(insertQuery, "INSERT INTO
  meteo_table(temperature_value, humidity_value,
  pressure_value) VALUES (%f, %f, %f);", temperature,
  humidity, pressure);
126     sqlite3_exec(db, insertQuery, 0, 0, 0);

127     // Закрытие базы данных
128     sqlite3_close(db);

129     int sensorValue = analogRead(4); // Чтение аналогового
  значения с пина датчика
130     float voltage = sensorValue * (3.3 / 4096.0); //
  Преобразование аналогового значения в напряжение (3.3 -
  напряжение питания ESP32, 4096 - разрешение АЦП ESP32)
```

```
131     float ppm = (1.1 * voltage - 0.1) * 1000 / 3; //
  Преобразование напряжения в концентрацию CO2 в ppm
  (значения 1.1 и 0.1 могут варьироваться в зависимости от
  калибровки датчика)

132     // Смена цвета всетодиода в зависимости от значения ppm
133     if (ppm <= 400){ // зеленый цвет при значиниях <=
  400ppm
134     ledcWrite(redChannel,0);
135     ledcWrite(greenChannel, 255);
136     }
137     else if (ppm <= 800){
138     ledcWrite(redChannel,126);
139     ledcWrite(greenChannel, 255);
140     }
141     else if (ppm <= 1200){
142     ledcWrite(redChannel,126);
143     ledcWrite(greenChannel, 126);
144     }
145     else if (ppm <= 1600){
146     ledcWrite(redChannel,255);
147     ledcWrite(greenChannel, 126);
148     }
149     else if (ppm >= 1800){ // Красный цвет при значиниях
  больше >= 1800ppm
150     ledcWrite(redChannel,255);
151     ledcWrite(greenChannel, 0);
152     }
153     // Задержка перед следующей итерацией
154     delay(1000);
155     }
```

ПРИЛОЖЕНИЕ Б

Листинг кода «index.html»

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4  <!-- Устанавливаем заголовок страницы веб-сайта -->
5  <title>Метеостанция</title>
6  <!-- Определяем настройки просмотра страницы на мобильных
   устройствах -->
7  <meta name="viewport" content="width=device-width, initial-
   scale=1">
8  <!-- Устанавливаем иконку (фавикон) для веб-страницы -->
9  <link rel="icon" type="image/x-icon" href="favicon.ico">
10 <!-- Подключаем внешний файл стилей CSS для оформления
   страницы -->
11 <link rel="stylesheet" type="text/css" href="style.css">
12 <!-- Подключаем библиотеку Highcharts, которая используется
   для создания графиков -->
13 <script src="scripts/highcharts.js"></script>
14 <!-- Подключаем скрипт, который добавляет возможность
   экспорта данных графика -->
15 <script src="scripts/export-data.js"></script>
16 <!-- Подключаем скрипт, который добавляет кнопки для
   экспорта данных графика -->
17 <script src="scripts/exporting.js"></script>
18 </head>
19 <body>
20 <!-- Создаем блок с заголовком страницы -->
21 <div class="header">
22 <!-- Пишем заголовок страницы -->
     a. <h1>СИСТЕМА СБОРА МЕТЕОДАННЫХ</h1>
23 </div>
24 <!-- Создаем блок для отображения графиков -->
25 <div class="charts">
26       <!-- Создаем сетку для расположения графиков внутри
         блока. -->
27       <div class="card-grid">
28       <!-- Создает карту (карточку) для каждого графика -->
29       <div class="card">
30       <!-- Создаем контейнер с уникальным идентификатором
         "chart-temperature" для отображения графика
         температуры -->
31       <div id="chart-temperature">
32       </div>
33       </div>
34       </div>
35       <!-- График влажности -->
36       <div class="card-grid">
37       <div class="card">
```

```
38      <div id="chart-humidity">
39      </div>
40      </div>
41      </div>
42      <!-- График давления -->
43      <div class="card-grid">
44      <div class="card">
45      <div id="chart-pressure">
46      </div>
47      </div>
48      </div>
49 </div>
50 <!-- Создаем блок с подвалом страницы -->
51 <div class="footer">
52 <!-- Отображем информацию о создателе страницы -->
     a. <h1>Шафигуллин Дмитрий Вадимович, студент группы КИ19-
        08Б</h1>
53 </div>
54 <!-- Подключаем скрипт, который обрабатывает данные и
   строит графики на основе этих данных -->
55 <script src="scripts/data-to-charts.js"></script>
56 </body>
57 </html>
```

ПРИЛОЖЕНИЕ В

Листинг кода «style.css»

```
1  html { /* Стиль всей страницы */
2  font-family: Arial, Helvetica, sans-serif; /* Шрифт */
3  display: inline-block; /* Чтобы применять выравнивание и
   другие свойства блочного элемента */
4  text-align: center; /* Выравнивает текст внутри элемента по
   центру */
5  }
6  body {
7  margin: 0; /* У\Отступы для всего содержимого страницы */
8  }
9  .header { /* */
10 overflow: hidden; /* Скрывает любое содержимое, которое не
   помещается внутри блока */
11 background-color: #0b1c52; /* Цвет фона блока */
12 color: white; /*  Цвет текста внутри блока */
13 }
14 .card-grid {
15 padding: 5%; /* Отступы вокруг содержимого блока */
16 max-width: 1800px; /*  Максимальная ширину блока */
17 margin: 0 auto; /* Выравнивание блока по горизонтали по
   центру */
18 display: grid; /* Устанавливает блок в качестве контейнера
   сетки */
19 grid-gap: 2rem; /*  Промежуток между элементами сетки */
20 /*
21 Шаблон колонок сетки .auto-fit позволяет элементам
   автоматически растягиваться для заполнения доступного
   пространства,
22 а minmax(200px, 1fr) устанавливает минимальную ширину
   каждой колонки в 200 пикселей,
23 но также позволяет им растягиваться с помощью fr (доля от
   доступного пространства).
24 */
25 grid-template-columns: repeat(auto-fit, minmax(200px,
   1fr));
26 }
27 .card {
28 background-color: white; /* Цвет фона карточки */
29 box-shadow: 2px 2px 12px 1px rgba(140,140,140,.6); /*
   Тень*/
30 }
31 .footer {
32 overflow: hidden; /* Скрывает любое содержимое, которое не
   помещается внутри блока */
33 background-color: #0b1c52; /* Цвет фона */
34 color: white; /* Цвет текста внутри блока */
35 }
```

ПРИЛОЖЕНИЕ Г

Листинг кода «data-to-charts.js»

```
1  // Создается новый экземпляр графика Highcharts,
2  // связанный с элементом с идентификатором 'chart-
   temperature'
3  var chartT = new Highcharts.Chart({
4  chart:{ renderTo : 'chart-temperature' },
5  // Заголовок графика устанавливается на 'Температура'
6  title: { text: 'Температура' },
7  // Создается серия данных для графика с пустым массивом
   значений.
8  series: [{
9  showInLegend: false,
10 data: [],
11 name: 'bme280'
12 }],
13 // Опции для отображения линии графика
14 plotOptions: {
15 line: { animation: false},
16 series: { color: '#059e8a'}
17 },
18 // Ось X настроена для отображения даты и времени
19 xAxis: { type: 'datetime'},
20 // Ось Y настроена с заголовком 'Temperature, °C'
21 yAxis: { title: { text: 'Temperature, °C' }},
22 // Подпись графика в правом нижнем углу отключена
23 credits: { enabled: false },
24 // Установлено, что время будет использовать локальное
   время, а не время в формате UTC
25 time: { useUTC: false},
26 });
27 // Эта функция будет выполняться с определенным интервалом
   времени
28 setInterval(function ( ) {
29 // Создается новый объект XMLHttpRequest, который
   используется для отправки HTTP-запросов на сервер
30 var xhttp = new XMLHttpRequest();
31 // Здесь устанавливается обработчик события изменения
   состояния запроса
32 // Функция будет вызываться каждый раз, когда изменяется
   состояние XMLHttpRequest
33 xhttp.onreadystatechange = function() {
34 // Проверяется, что состояние запроса равно 4 (завершено),
   а статус ответа сервера равен 200 (Успешный ответ)
35 if (this.readyState == 4 && this.status == 200) {
36       // Создается переменная x, которая содержит текущее
         время в миллисекундах
37       // Она используется для определения временной метки на
         графике
```

```
38      var x = (new Date()).getTime(),
39      // Создается переменная y, которая содержит числовое
        значение, полученное из ответа сервера
40      // Она будет представлять значение температуры,
        полученное от датчика.
41      y = parseFloat(this.responseText);
42      // Проверяется длина массива данных на графике
43      // Если она превышает 30 элементов, то будет
        использоваться метод addPoint с параметрами [x, y],
        true, true, true
44      // Это означает, что новая точка будет добавлена в
        конец графика, старая точка будет удалена с начала, и
        график будет автоматически прокручиваться для
        отображения новых данных.
45      if(chartT.series[0].data.length > 30) {
46      chartT.series[0].addPoint([x, y], true, true, true);
        //  В этом случае новая точка будет добавлена в конец
         графика без удаления старых точек
47      }
48 else {
49        chartT.series[0].addPoint([x, y], true, false, true);
50        }
51 }
52 };
53 //  Здесь устанавливается метод и URL для отправки GET-
   запроса на сервер
54 // В данном случае, отправляется GET-запрос на URL
   "/temperature"
55 xhttp.open("GET", "/temperature", true);
56 // Отправляет HTTP-запрос на сервер
57 xhttp.send();
58 // Интервал в 1000 миллисекунд
59 }, 1000 ) ;

60 var chartH = new Highcharts.Chart({
61 chart:{ renderTo:'chart-humidity' },
62 title: { text: 'Влажность' },
63 series: [{
64 showInLegend: false,
65 data: [],
66 name: 'bme280'
67 }],
68 plotOptions: {
69 line: { animation: false}
70 },
71 xAxis: { type: 'datetime'},
72 yAxis: { title: { text: 'Humidity, %' }},
73 credits: { enabled: false },
74 time: { useUTC: false},
75 });
76 setInterval(function ( ) {
77 var xhttp = new XMLHttpRequest();
78 xhttp.onreadystatechange = function() {
```

```
79 if (this.readyState == 4 && this.status == 200) {
80    var x = (new Date()).getTime(),
81    y = parseFloat(this.responseText);
82    if(chartH.series[0].data.length > 30) {
83    chartH.series[0].addPoint([x, y], true, true, true);
84    } else {
85    chartH.series[0].addPoint([x, y], true, false, true);
86    }
87 }
88 };
89 xhttp.open("GET", "/humidity", true);
90 xhttp.send();
91 }, 1000 ) ;

92 var chartP = new Highcharts.Chart({
93 chart:{ renderTo:'chart-pressure' },
94 title: { text: 'Давление' },
95 series: [{
96 showInLegend: false,
97 data: [],
98 name: 'bme280'
99 }],
100     plotOptions: {
101     line: { animation: false},
102     series: { color: '#18009c'}
103     },
104     xAxis: { type: 'datetime'},
105     yAxis: { title: { text: 'Pressure, hPa' }},
106     credits: { enabled: false },
107     time: { useUTC: false},
108     });
109     setInterval(function ( ) {
110     var xhttp = new XMLHttpRequest();
111     xhttp.onreadystatechange = function() {
112     if (this.readyState == 4 && this.status == 200) {
113    var x = (new Date()).getTime(),
114    y = parseFloat(this.responseText);
115    if(chartP.series[0].data.length > 30) {
116    chartP.series[0].addPoint([x, y], true, true, true);
117    } else {
118    chartP.series[0].addPoint([x, y], true, false, true);
119    }
120     }
121     };
122     xhttp.open("GET", "/pressure", true);
123     xhttp.send();
124     }, 1000 ) ;
```

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« 20 » 06 2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Система сбора метеоданных
тема

Руководитель 20.06.2023 доцент, канд. физ.-мат. наук К. В. Коршун
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 20.06.2023 Д. В. Шафигуллин
подпись, дата инициалы, фамилия

Нормоконтролер 20.06.2023 доцент, канд. физ.-мат. наук К. В. Коршун
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2023