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

Институт машиностроения
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.04.04 Электроника и наноэлектроника
(код и наименование направления подготовки)

Электронные приборы и устройства
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Автономное переносное устройство логированных жилых помещений»

Студент Н.И. Мордвинов
(И.О. Фамилия) (личная подпись)

Научный руководитель к.т.н., доцент, М.В. Позднов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Содержание

Список обозначений и сокращений

Аббревиатура	Расшифровка	Расшифровка на русском
IoT	Interment of things	Интернет вещей
LCD	Liquid Crystal Display	Жидкокристаллический дисплей
ОС		Операционная Система
ПО		Программное Обеспечение
eCO2		эквивалент диоксида углерода
tVOC	Total Volatile Organic Compounds	Индекс летучих органических соединений
OLED	Organic-Light-Emitting-Diodes	Органический светодиод
MVP	Minimum Viable Product	Минимально жизнеспособный продукт
JSON	JavaScript Object Notation	Обозначение объекта JavaScript
SoC	System On Chip	Система на кристалле
BLE	Bluetooth Low Energy	Bluetooth с низким энергопотреблением
GPIO	General-Purpose Input/Output	Интерфейс ввода/вывода общего назначения
АЦП		Аналого-Цифровой Преобразователь
i2c	Inter-Integrated Circuit	Последовательная асимметричная шина
MOSFET	Metal–Oxide–Semiconductor Field-Effect Transistor	полевой транзистор с изолированным затвором
SDK	Software Development Kit	Набор средств разработки
IDE	Integrated development environment	Интегрированной среды разработки
СКВ		Система контроля версий
ФС		Файловая система
RTOS	Real-Time Operating System	Операционная Система Реального Времени
RTC	Real Time Clock	Часы реального времени
OTA	Over-The-Air	Обновление по воздуху
БД		База данных
СУБД		Система управления базами данных
MVC	Model-View-Controller	Модель-Представление-Контроллер
ЛУТ		Лазерно-Утюжного Технология
AWG	American Wire Gauge	Американский калибр проводов

Введение

Множество сфер человеческой деятельности упрощаются и совершенствуются благодаря применению в них современных информационных технологий, IoT (Interment of things с англ. Интернет вещей) [17], уже не является новостью для людей, умные устройства, управляемые со смартфона, уже начали входить в привычку, однако, можно заметить, что данный процесс происходит неравномерно и не все сферы человеческой деятельности одинаково получили внедрение информационных технологий.

Сущностью рассматриваемой проблемы состоит в том, при сдаче жилого фонда арендодатели ставят ряд условий, которые необходимо выполнять арендатору, что зачастую закрепляется в договоре, а также имеет штрафные санкции за несоблюдение данных условий.

К таким условиям можно отнести, например, запрет на курение в помещении, запрет на громкий шум в определённых интервалах времени. Эти условия назначаются не случайно и имеют достаточно весомые обоснования.

Уровень шума для отдельный строений это не является такой проблемой, как для многоквартирных домов, там зачастую это мешает соседям их обычной жизни. Возможны даже ситуации, когда соседи «стучатся» непосредственно в двери шумных арендаторов квартиры, звонят арендодателю, или же могут вызвать участкового полицейского.

Проблема курения в квартире — это не только самое курение, например обычных сигарет, но и так же, например кальянов или использование непредназначенных для этого источников открытого пламени. Сами эти действия могут мешать соседям, например при распространении дыма по вентиляции, запах так же въедается в предметы интерьера, а мелкодисперсные частоты из дыма оседают на поверхностях давая желтый сложно удаляемый налёт. Так же, курение в помещение может повредить

предметы интерьера и обивку, например если сигарету, тлеющий оставят на комоде, но что гораздо хуже это потенциальная пожароопасность таких действий при неосторожности.

Контролировать выполнение договора в связи со сложностью отслеживания наступления указанных выше ситуаций представляется затруднительным. Однако, поскольку некоторые условия могу быть сформулированы с технической точки зрения, то возможно использовать некоторое устройство, которое бы автоматически отслеживало их выполнение.

В устройстве для отслеживания параметров можно установить датчики дыма, шума, возможно несколько концевых выключателей и собственный источник питания, для обеспечения непрерывной работы.

Можно предположить, что устройство могло бы размещаться на стене или потолке, имело бы модульную систему датчиков, отсоединяемое основание, которое бы регистрировало бы что устройство кто-то трогал и снял со своего места крепления. Датчики были был заключены в корпуса одного стандартного размера и как «кубики лего» вставлялись бы в базовое устройство, что могло бы в дальнейшем позволить человеку выбрать какие датчики он хочет, например датчик движения, датчик освещённости. Так же возможно, что к устройству можно будет подключить датчик открытия двери/окна на концевом выключателе или герконе.

В итоге бы получалось много функциональное устройство, которое бы регистрировало бы курят гости ли, шумят ли, а после сдачи квартиры владельцу, недобросовестных арендаторов можно было бы оштрафовать за нарушение условий договора.

1.1 Состояние вопроса

В промышленности записи некоторых параметров, таких как содержание CO2, влажности воздуха и температуры уже разработаны такие устройства как логгеры. Логгер – это специализированное электронное устройство, оно применяется для регистрации и фиксации определённых показателей среды. Обычно данные устройства применяются в пищевой индустрии, например для отслеживания температуры в авторефрижераторах или же в холодильных камерах, однако спектр применения таких устройств гораздо шире и существует большое количество их разновидностей. Принцип работы таких устройств достаточно прост, они устанавливаются в месте, в котором необходимо проконтролировать определённый параметр, после происходит запись показаний с внутреннего датчика в память устройства – запись лога, после прошествии определённого времени устройство отключается и из него получают зафиксированные показания -лог.

Существует так же другой подход, это разнообразные датчики и мониторы микроклимата, уже для дома. Бывают как отдельные датчики температуры или, например освещённости в помещении, так и устройства, совмещающие в себе несколько функций, однако, чаще, данные устройства предоставляют возможность получения лишь текущих показаний с некоторого пользовательского интерфейса и не имеют встроенной памяти для ведения записи. Так же существует проблема того, что с некоторыми датчиками необходимо дополнительно покупать базовую станцию или хаб, который будет производит агрегацию показаний, снятых с датчиков и только после этого к ним можно будет получить доступ. Кроме этого, обычно пользовательские интерфейсы данных систем не приспособлены для того, чтобы получать данные из разных физических мест, как может быть в реальной ситуации, когда у арендодателя имеются несколько жилых помещений в разных домах, которые доступны для аренды.

1.2 Рассмотрение аналогов

Было решено рассмотреть имеющиеся на рынке аналоги для выявления их преимуществ и недостатков.

ClearGrass Air Monitor

Это система мониторинга качества воздуха, имеет встроенный сенсорный LCD дисплей, на котором можно выводит текущие показания, изображённый на рисунке 1, ниже, несколько датчиков газоанализаторов, для оценки качества воздуха и датчик температуры и влажности. Само устройство имеет аккумуляторную батарею, с помощью которой оно может некоторое непродолжительное время работать автономно.

В качестве преимуществ можно выделить датчики газоанализаторы профессионального уровня. В качестве недостатков необходимо подчеркнуть, что устройство ориентировано скорее не на анализ истории изменения показаний, а скорее на контроль текущего показателя. Историю можно посмотреть лишь с экрана самого устройства, в приложении можно наблюдать только текущие показания, так же в приложении Mi Home есть возможность добавлять только новые комнаты, но не квартиры в целом.

Рисунок 1 - Фотография ClearGrass Air Monitor

Характеристики

- Процессор 1,2 ГГц четырехъядерный A7 (Rockchip RK3128 PX3-SE)
- ОС Embedded Linux
- Вес 217 гр
- Размер экрана: 3,1 дюйма (разрешение экрана 720×720)
- Беспроводная сеть: Wi-Fi и Bluetooth
- Тип батареи: литий-ионный аккумулятор (2000 мАч / 3,7В)
- Интерфейс: USB-C (только зарядка)
- Питание: 5В, 1.5 А
- Основной датчик: Sensirion SCD30

Пределы измерений

- PM2,5: 0 ~ 999 мкг /м3 (датчик Sheng Sirui ZS02007)
- Температура: -10 ~ 50°C (датчик Sheng Sirui SHT30)
- Влажность: 0 ~ 100% (датчик Sheng Sirui SHT30)
- tVOC: 0,005 ~ 9,999 мг/м3 (датчик Sensirion MOXSens SGP30)
- CO_2 400 ~ 9999 ppm (датчик Sensirion SCD30) [21]

Цена: 11 000 руб

Honeywell HAQ

Система мониторинга качества воздуха. Так же имеет сенсорный OLED дисплей, представленный на рисунке 2, ниже для вывода текущих показаний и встроенный аккумулятор.

Из преимуществ можно выделить меньшие габариты и чуть большую ёмкость аккумулятора, недостатки схожи с предыдущим образцом.

Рисунок 2 - Фотография Honeywell HAQ

Характеристики устройства:

Количество датчиков: 5 шт, PM2.5, T/H, TVOC (ЛОС), CO2, HCHO (формальдегиды)

Дисплей: Сенсорный OLED

Материал корпуса: алюминий

Приложение: Да

Интерфейсы: Wi-Fi, MicroUSB

Встроенный аккумулятор: 2600 мАч

Входное напряжение: 5 В

Потребляемый ток: 1 А

Измерительный диапазон влажности: 20–90% RH

Диапазон измерений CO_2 : 0–3000 ppm

Диапазон рабочих температур: 0–50 °C

Габариты: 80 x 80 x 22 мм

Вес: 156 г [33]

Цена: 10 000 руб

Логгер iPlug TH температура и влажность (одноразовый)

Данное устройство предназначено для длительной автономной записи показаний температуры и влажности окружающей среды во внутреннюю память и по окончании срока вывода данных в виде скомпаннового файла по USB, рисунок 3.

Из положительных качеств можно выделить меньшую стоимость относительную простоту устройства и автономность. Недостатков же больше, это прежде всего его одноразовость и малая информативность.

Рисунок 3 - Фотография одноразового логгера iPlug TH

Характеристики устройства

Диапазон температур: -40°C до +65°C

Диапазон влажности: 0 – 100 % рс

Шкала температуры: °C & °F

Погрешность по влажности: 2%

Количество записей за сессию: по 15 000

Интервал между записями: от 30 секунд

Форма отчета: Программа

Связь с компьютером: USB порт

Размеры (мм): 73 x 15 x 33

Вес: 30гр.

Срок действия батареи/ тип 2 года / 3V Lithium CR 2450

Индикация температуры: устанавливается 4 уровня [30]

Цена: 3600 руб

Логгер данных температуры и влажности (Многоразовый) Testo 184 H1

Данное устройство предназначено для автономной записи показаний окружающей среды во внутреннюю память с последующим их получением через USB порт, рисунок 4.

Ключевой особенностью является перезаряжаемая аккумуляторная батарея. Недостатками же можно считать так же малую информативность показаний и крайне высокую стоимость.

Рисунок 4 - Фотография Testo 184 H1

Характеристики устройства:

Диапазон температур: -20°C до +70°C

Диапазон влажности: 0 – 100 %

Шкала температуры: °C

Батарея: Сменная батарея

Гарантия аккумулятора: 500 дней

Форма отчета: Программа

Связь с компьютером: USB порт

Размеры (мм): 44 x 12 x 47

Вес: 45гр. [32] [31]

Цена: 16 000 руб

Датчик открытия окна/двери FIBARO FGDW-002

Данный датчик является классическим примером части одной из экосистем умного дома, сам датчик автономен минималистичен и прост в работе, представлен на рисунке 5, ниже, однако требует наличие базовой станции, для агрегации данных, что является его явным недостатком. С другой же стороны его преимуществом является то, что кроме основной его функции, датчик так же может измерять температуру воздуха.

Рисунок 5 - Фотография FIBARO FGDW-002

Характеристики устройства

Источник питания: батарея 3,6 В постоянного тока

Тип батареи: ER14250 (1 / 2AA)

Срок службы батареи: около 2 лет (настройки по умолчанию)

Радио-протокол: Z-Wave (чип серии 500)

Радиочастота: 868,4 МГц ЕС; 908,4 МГц США; 921,4 МГц ANZ; 869,0 МГц RU;

Мощность радиопередачи: до 2 дБм (EIRP)

Диапазон: до 40 м в помещении

(в зависимости от местности и конструкции здания)

Предназначенная среда: только для использования в помещении

Рабочая температура: 0-40 ° C

Диапазон измерения температуры: 0-60 ° C

Точность измерения температуры: ± 0,5 ° C

Размеры (Д x Ш x В): 71 x 18 x 18 мм [6]

Цена: 5500 руб

1.3 Цель работы

Резюмируя материал изложенный выше, можно заключить, что некоторого конкретного и воплощённого в одном устройстве решения имеющейся проблемы нет, таким образом цель работы становится разработка прототипа такого устройства, который в можно было бы сравнить с MVP (minimum viable product с англ. минимально жизнеспособный продукт), на котором можно было бы протестировать работу системы в целом, который не обязательно включал бы в себя все желаемы части.

Само же устройство должно стать чем-то средним между всеми рассмотренными аналогами и вобрать в себя только необходимое: возможность длительной автономной работы, несколько датчиков, возможность отправки данных на удалённый сервер и просмотра этих данных в виде графика.

Решено было разработать платформу, которую в последующем можно будет доработать и добавить в неё все необходимые технические решения, которая может собирать показания CO2 температуры и влажности, так же было решено использовать некоторый Wi-FI радио модуль для связи устройства с сервером.

Для достижения поставленной цели работы необходимо решить следующие задачи:

- Разработка структурной схемы устройства
- Разработка алгоритма и написание программного кода
- Разработка принципиальной схемы устройства
- Разработка и изготовление печатной платы
- Сборка и тестирование устройства

Вывод

В результате был произведён анализ известных технических решений, были выявлены их преимущества и недостатки, было решено разработать прототип, были поставлены задачи работы.

2 Описание устройства

Проанализировав данные, полученные в ранее, было принято решение, собирать информацию о показаниях на одном удалённом веб сервере, с нескольких устройств. В таком случае сеть из устройств и сервера, будет построена по схеме звезда, где главенствующим будет веб сервер, а остальные узлы будут подключатся к нему, это изображено на рисунке 6

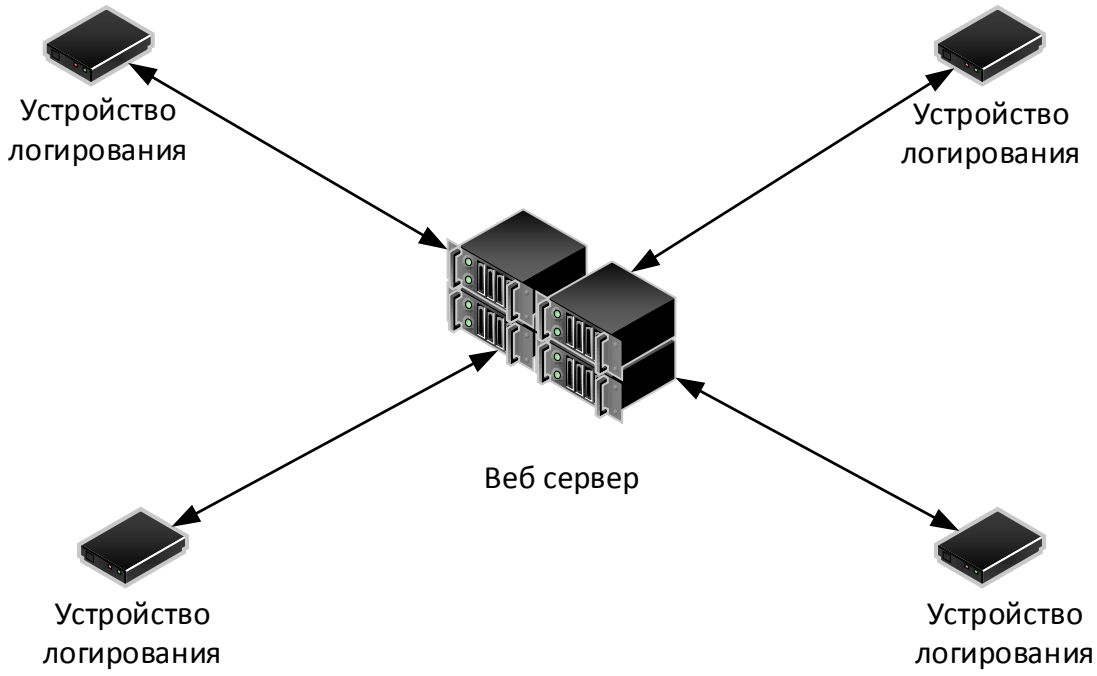


Рисунок 6 - Графическое представление топологи сети

Каждый отдельный экземпляр устройства, будет подключатся через Wi-Fi к беспроводному маршрутизатору, а далее получив доступ к интернету, отправлять данные по протоколу HTTP в виде JSON файла, на веб сервер, что представлено на рисунке 7

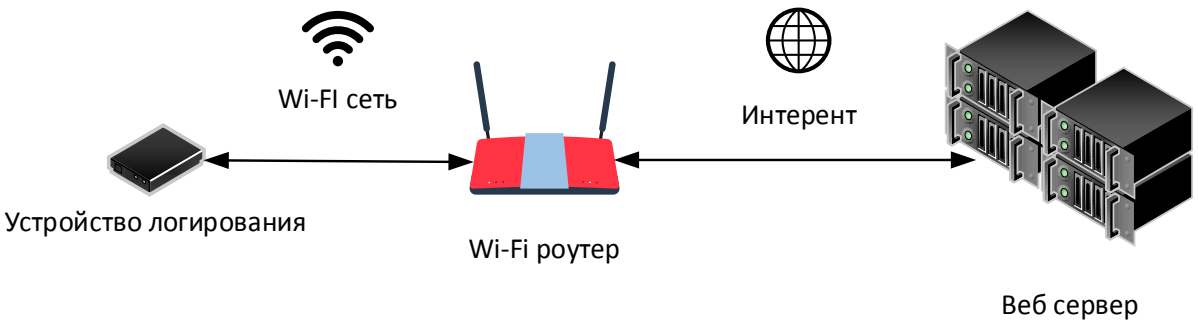


Рисунок 7 - Схема сети устройство-сервер

Так же, необходимо отметить, что веб сервер, который хранит и обрабатывает данные, может быть локальным, то есть, это не обязательно должен быть удалённый сервер на каком-либо хостинге, это может быть некоторый компьютер в локальной сети.

Саму же систему в целом, можно представить в виде первичной структурной схемы, представленной на рисунке 8

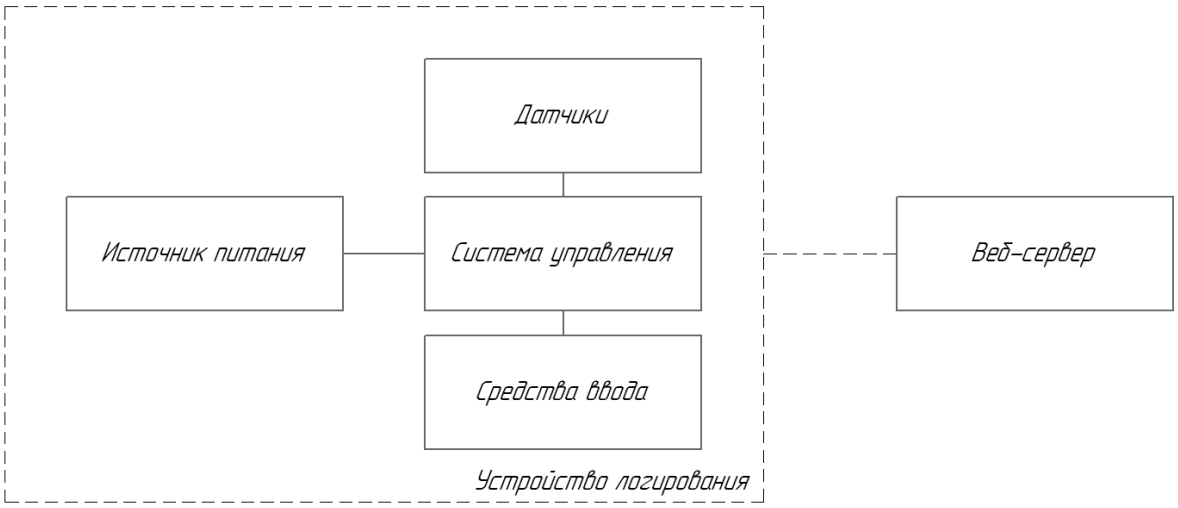


Рисунок 8 - Первичная структура схема системы

В дальнейшем каждый компонент данной системы будет подобран, в соответствии с требованиями, после чего будет составлена полная структура схема. Источник питания будет обеспечивать компоненты необходимы

током, система управления будет получать информацию с датчиков и устройств ввода, после чего анализировать, компоновать и отправлять на веб сервер.

Вывод

Таким образом была предложена структура, желаемого устройства, была составлена первичная структура схема, обозначены первичные и вторичные компоненты системы, была выявлена необходимость разработки как программного обеспечения для устройства, так и для серверной части.

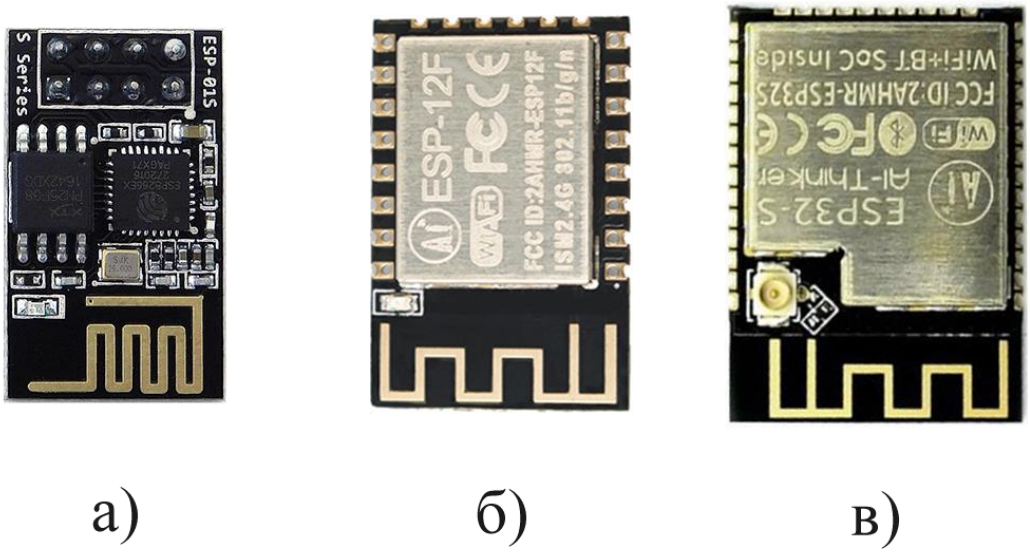
3 Подбор компонентой базы

Для разработки устройства требуется подобрать те схем-технические решения, которые будут оптимальны как со стороны стоимости, так и со стороны практичности использования. Для управления устройством было решено использовать микроконтроллер, который будет подобран далее,

3.1 Выбор микроконтроллера

В качестве системы управления было решено использовать микроконтроллеры компании Espressif Systems. Данная компания специализируется на производстве **SoC**, которые уже включают в себя модули беспроводного подключения таких технологий как BLE и Wi-Fi, что делает их не заменимым в проектах типа IoT. Выбор обосновывается так же и тем, что микроконтроллеры данной компании получили широкую популярность в интернет-сообществе, а значит, что при разработки технических решений, можно будет полагаться не только документацию производителя, но и на различные дополнительные интернет-ресурсы, такие как: онлайн репозитории кода, тематические форумы, сайты и др.

После чего необходимо было выбрать какой конкретно модуль использовать. Для сравнения были взяты три популярных модуля: ESP-01S, ESP-12F и ESP-32S, их фотографии представлены на рисунке 9



а) б) в)

Рисунок 9 - фотографии модулей микроконтроллеров: а) ESP-01S, б) ESP-12F, в) ESP-32S

Их ключевые характеристики [8],[9],[19] приведены ниже, в таблице 1

Таблица 1 - Сравнение характеристик микроконтроллеров

	ESP-01S	ESP-12F	ESP-32S
Чип	ESP8266EX	ESP8266EX	Xtensa Dual-Core LX6
Разрядность ядра	32 бит	32 бит	32 бит
Частота ядра	80 МГц	80 МГц	240 МГц
Объём Flash памяти	1 Мб	4 Мб	4 Мб
Объём оперативной памяти	80 Кб	80 Кб	520 Кб
Количество GPIO	4	17	39
Встроенная PCB-антенна	да	да	да
Рабочее напряжение	от 3 до 3,6 В	от 3 до 3,6 В	от 2,2 до 3,6 В
Рабочий ток	40-80 мА	40-80 мА	160-260 мА
Наличие Wi-Fi	да	да	да
Наличие BLE	нет	нет	да
Металлическое	нет	да	да

экранирование			
ESP-Touch	да	да	да
Внешний вывод АЦП	нет	да	да
Габариты	14.3x24.8x3 мм	16x24x3 мм	18x25x2.8 мм
Средняя рыночная стоимость	105 р	191 р	592 р

Из данной таблицы видно, что модули микроконтроллеров ESP-01S, ESP-12F имеют одинаковый чип, а соответственно ток потребления, объём оперативной памяти и рабочее напряжение, однако они различаются в объёме flash памяти и количестве GPIO, тут более выгодно вновь показывает ESP-12F. В габаритных размерах в целом они схожи. Так же преимуществом для ESP-12F является наличие металлического экрана по периметру области содержащий сам чип и его обвязку. Так же преимуществом для ESP-12F является наличие внешнего вывода АЦП, что в последующем может пригодится для измерения некоторый величины, представленной в аналоговом варианте.

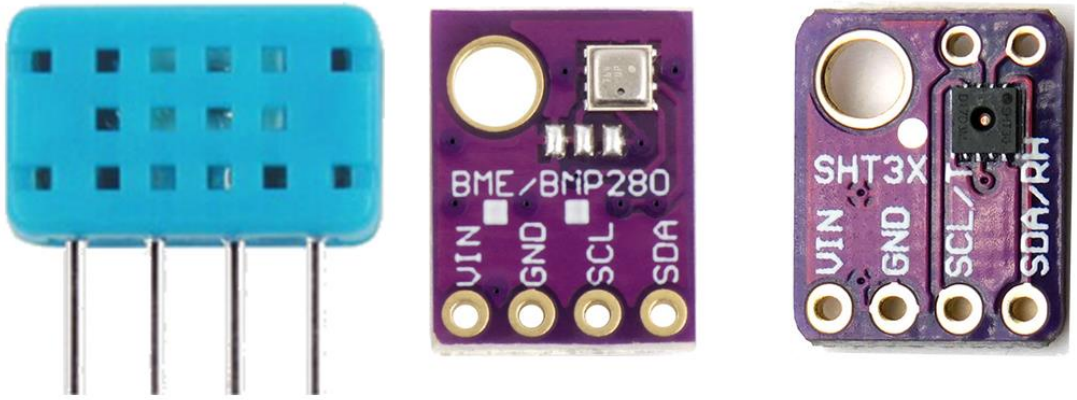
Если же сравнивать ESP-12F и ESP-32S, видно, что последний имеет большую вычислительную мощность из-за чипа, частоты ядра и объёма оперативной памяти. Так же видно различие в количество GPIO, ESP-32s имеет их наибольшее количество. Однако заметно и больший рабочий ток у ESP-32s, это связано с наличие Bluetooth модуля в чипе и другим ядром.

В качестве компромисса, было принято решение использовать ESP-12F, т.к. данный модуль микроконтроллера совмещает в себя малые габаритные размеры, большой объём flash памяти, меньший рабочий ток и среднюю дороговизну за один экземпляр модуля.

3.2 Выбор датчика температуры и влажности

При выборе датчика температуры и влажности основными критериями стали: компактность, малый потребляемый ток, дешевизна, интерфейс передачи данных.

Были рассмотрены следующие модули датчиков DHT12, BME280, SHT30, их фотографии представлены на рисунке 10



а) б) в)

Рисунок 10 - фотографии датчиков температуры и влажности: а) DHT12, б) BME280, в) SHT30

Так же была составлена сравнительная таблица 2 их основных характеристик [5],[3],[22]

Таблица 2 - Сравнение характеристик датчиков температуры и влажности

	DHT12	BME280	SHT30
Питание	2.7-5.5 В	1.7-3.6 В	2.4-5.5 В
Потребляемый ток	30-800 мкА	1.8-3.6 мкА	1.5 мкА
Измерение влажности	20-90 %	0-100%	0-100%
Измерение влажности,	+/- 5%	+/- 2%	+/- 3%

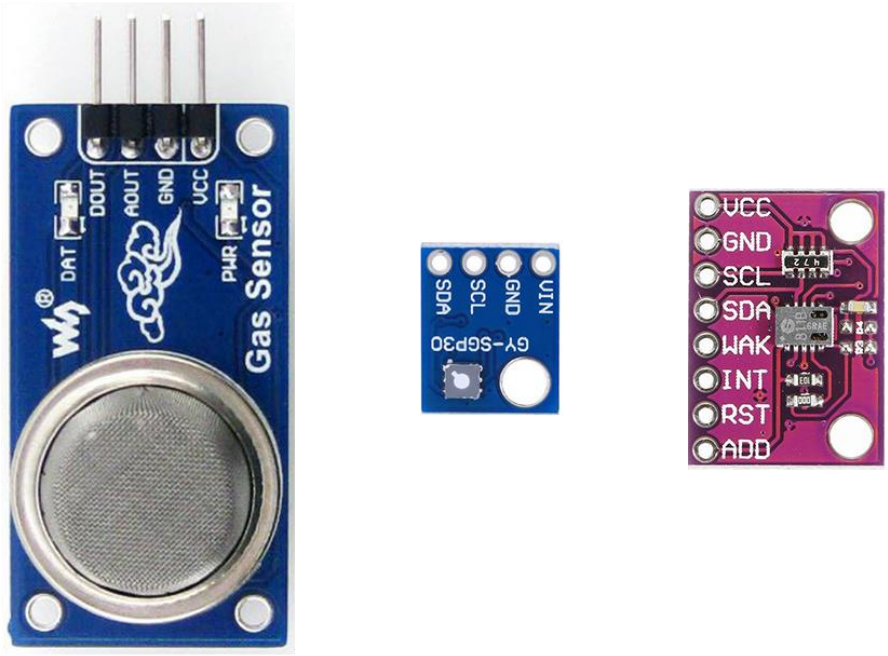
погрешность			
Измерение температуры	от -20 до + 60 °C	от -40 до 80 °C	от -40 до +125°C
Измерение температуры, погрешность	+/- 0.5 °C	+/- 0.5 °C	+/- 3%
Частота измерений	0.5 Гц	>10 Гц	>10 Гц
Средняя рыночная стоимость	296 р	250 р	377 р

Исходя из данной таблицы можно заметить, что наибольший диапазон питающих напряжений и снимаемых показаний по температуре имеет SHT30. DHT12 оказался менее точным своих конкурентов, и так же имеет наименьший диапазон измерений влажности, BME280 же является более выходным с точки зрения цены доступных диапазонов измерений. Приятным бонусом BME280 является так же то, что он позволит производит измерение атмосферного давления.

Исходя из вышеизложенной информации было принято решение в качестве датчика температуры и влажности выбрать BME280.

3.3 Выбор датчика C02

Выбор модулей датчиков co2 не так велик по сравнению с датчиками температуры и влажности, особенно если не учитывать датчики со стоимостью заранее слишком большой для проекта, например CO2-Z19, этот датчик имеет стоимость приблизительно в 30 раз превышающую стоимость микроконтроллера, который им будет управлять. Поэтому на рассмотрение были выдвинуты модули датчиков MQ-135, GY-SGP30 и CCS811, они изображены на рисунке 11



а) б) в)

Рисунок 11 - фотографии датчиков co2: а) MQ-135, б) GY-SGP30, в) CCS811

Далее, была составлена сравнительная таблица 3 основных характеристик модулей [27],[29],[28]:

Таблица 3 - Сравнение характеристик датчиков CO2

	MQ-135	GY-SGP30	CCS811
Напряжение питания	5 В	1.8-5 В	1.8-3.6 В
Потребляемый ток	160 мА	40 мА	0.019-26 мА
Получение показаний в цифровом варианте	нет	да	да
Диапазон измеряемой концентрации co2	10-1000 ppm	400-60000ppm	400-8192 ppm
Диапазон измеряемых величин tvoc	нет	0-60000ppb	0-1187 ppb

Период опроса	-	0.025 с	0.25 – 60 с
Средняя рыночная стоимость	280 р	900 р	750 р

Однако, необходимо заметить, что в данной таблице не отражено что модуль датчика MQ-135 является скорее аналоговым чем цифровым, то есть данный модуль имеет построечный резистор, который может регулировать его порог чувствительности и путём этого можно задавать некоторую пороговую величину концентрации CO2, после которой на цифровой выход модуля датчика будет подавятся логическая единица, однако фактическое снятие показаний возможно только с аналогового выхода датчика, а соответственно частота оцифровки сигнала и будет является частотой снятия показаний. Остальные же два датчика, могут передать сигнал по шине i2c, что упрощает с ними работу.

Было принято решение использовать датчик CO2 CCS811, т.к. он имеет среднюю стоимость и минимальный потребляемый ток, а также из-за того, что данному датчику не требуется аналоговый вход для снятия показаний.

3.4 Выбор источника питания

Для работы устройства, необходим источник питания, для того чтобы устройство было автономным было принято решение отказаться от варианта внешнего блока питания в пользу внутреннего источника питания. Были рассмотрены варианты использования солевых, щелочных и литьевых элементов питания. Из ранее приведённых таблиц видно, что потребляемый ток может достигать значений более 100 мА, что является не желательным режимом работы для солевых элементов питания, напряжение щелочных элементов питания может сильно варьироваться, на протяжении цикла разряда, что может негативно сказаться на стабильности работы радио модуля ESP-12f, лучше всего в данной категории показывают литиевые

элементы питания, они обладают большой ёмкостью и достаточно стабильным напряжением [26].

Однако литьевые элементы питания достаточно дороги в эксплуатации, поэтому было решено всё же использовать наиболее распространённый тип аккумуляторных батарей в переносных устройства – Li-ion (литий-ионные) аккумуляторы.

3.4.1 Выбор питающего элемента

Литий ионные аккумуляторы имеют ряд преимуществ, такие как:

- Высокая ёмкость
- Малые габариты
- Небольшая масса
- Отсутствует эффект "памяти"
- Низкая степень саморазряда
- Большое количество циклов заряда/разряда
- Возможность форсированной (быстрой) зарядки
- Возможность изготовления любой формы и конфигурации

На данный момент на рынке присутствует достаточно большое количество типов литий-ионных аккумулятор, они отличаются как по составу, так и по форме исполнения.

В качестве формата питающего элемента было принято решение использовать тип 18650, аккумуляторная батарея диаметром 18 мм, длинной 65мм и типом формы 0, то есть цилиндрической [35].

Так же для данного источника питания потребуется специальный батарейный отсек, в который вставляется аккумулятор для работы, он показан на рисунке 12

Рисунок 12 - батарейный отсек для аккумуляторов типа 18650

3.4.2 Выбор схемы зарядки аккумулятора

Для работы аккумуляторных батарей требуется их зарядка, после каждого цикла разряда. Обычно за это отвечает специальная схема, которая контролирует как напряжение, подаваемое на аккумулятор, так и ток зарядки, а после достижения некоторого порогового значения отключает зарядку аккумулятора, во избежание его перезаряда.

Для реализации схемы зарядки аккумулятора было решено использовать микросхему TP4506, которая подходит для зарядки одной ячейки литий-ионной батареи. Данная микросхема работает в линейном режиме, стабилизируя напряжение, подаваемое на аккумулятор, но и так же

ограничивает ток зарядки, что указано в её документации [23] и наглядно видно на схеме цикла зарядки, взятой из документации, представленной на рисунке 13

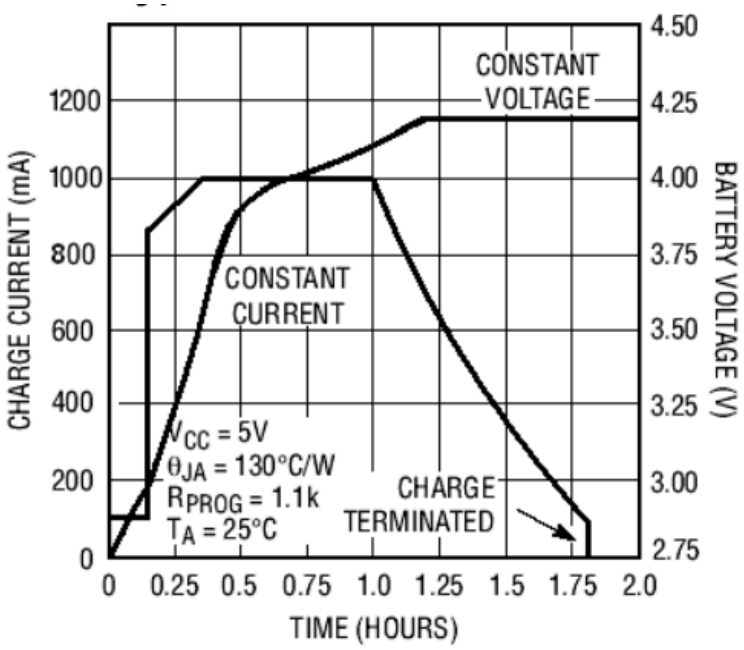
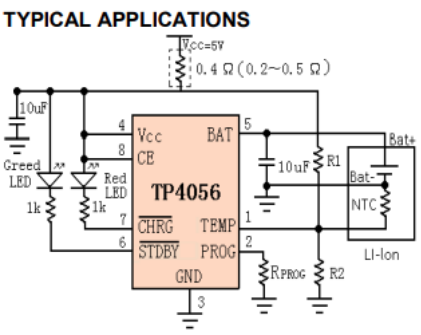


Рисунок 13 - Схема цикла полной зарядки одной ячейки аккумулятора ёмкостью в 1000 мАч

Схема подключения достаточно проста и требует минимальное количество компонентов обвязки, она представлена на рисунке 14 а)



Rprog Current Setting

R_{PROG} (k)	I_{BAT} (mA)
10	130
5	250
4	300
3	400
2	580
1.66	690
1.5	780
1.33	900
1.2	1000

а) б)

Рисунок 14 - а) схема подключения TP4056 б) таблица значений резистора, задающего максимальный ток зарядки

Так же возможно задать максимальной ток зарядки, с помощью резистора Rprog, его можно определить с помощью таблицы, представленной на рисунке 14 б), так же, он вычисляется по формуле:

$$I_{BAT} = \frac{1\text{В}}{Rprog} \times 1200$$

Сам же ток зарядки аккумулятора, регламентируется в самой документации к аккумуляторной батарее.

Однако, из расчёта что аккумулятор будет является сменной частью устройства, и чтобы пользователь смог выбрать аккумулятор любой ёмкости, предположим, что минимальная ёмкость элемента питания 1000мАч, а соответственно и номинальный ток составить 0.2 от ёмкости аккумулятора, [36] то есть 200 мА. Согласно формуле, выбрали задающей резистор, его сопротивление должно составить 6 кОм.

3.4.3 Выбор схемы защиты аккумулятора

Несмотря на то, что в микросхеме TP4506 уже присутствует контроль за перезарядкой аккумулятора, аккумулятор всё ещё не защищён от коротких замыканий и от переразряда. Обычно на этом специализируется отдельная схема, состоящая из управляющего устройства и силовых ключей, которые производят коммутацию и при срабатывании защиты размыкают цепь.

Для этих целей было решено использовать микросхему DW01A и силовую сборку, состоящую из двух MOSFET транзисторов FS8205A, схема их подключения, согласно документации, [7] изображена на рисунке 15

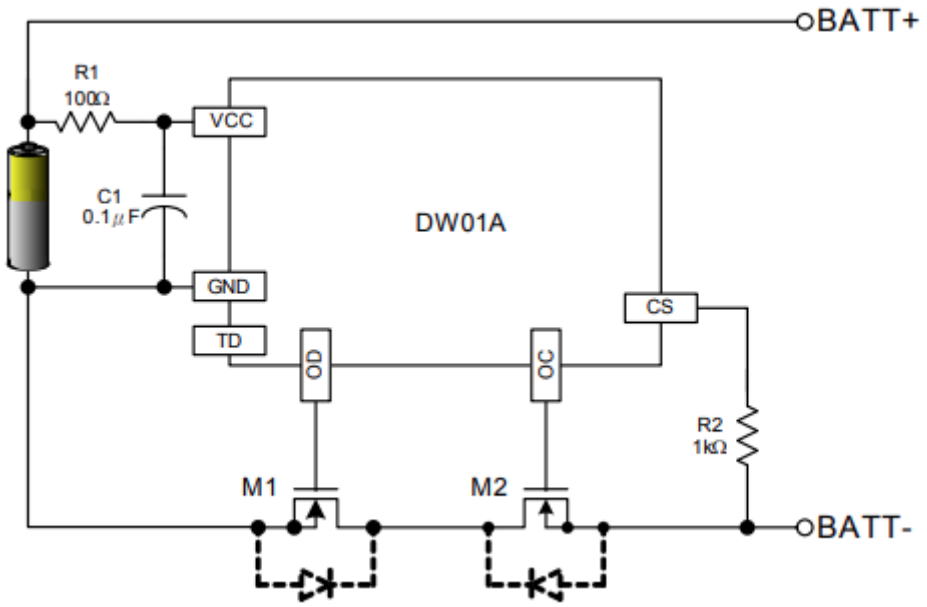


Рисунок 15 - схема подключения DW01A и FS8205A

3.4.4 Выбор стабилизатора напряжения

Выходное рабочее напряжение литий-ионного аккумулятора колеблется в диапазоне 3.6 - 4.2 В, что превышает рабочее напряжение датчиков и микроконтроллера, это значит, что необходим стабилизатор напряжения, рассчитанный на выходное напряжение 3.3 В. Для этой цели подходит линейный стабилизатор ams1117 модификации на 3.3 В. Данный стабилизатор поддерживает входное напряжение до 12 В, а максимальный ток до 1 А [1].

Так же в официальной документации есть схема подключения данного стабилизатора, она представлена на рисунке 16

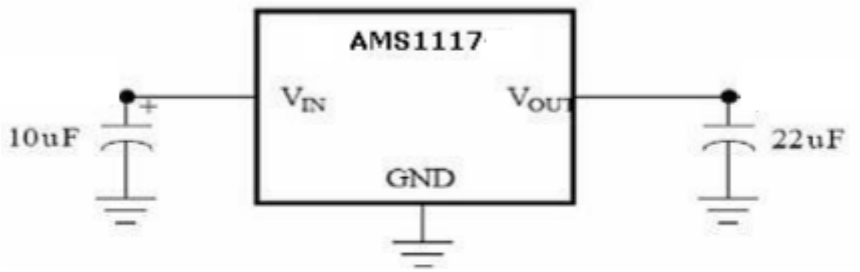


Рисунок 16 - схема стабилизатора напряжения на ams1117

3.5 Составление структурной схемы

Согласно подобранным компонентам, была составлена полная структурная схема, представленная на рисунке 17

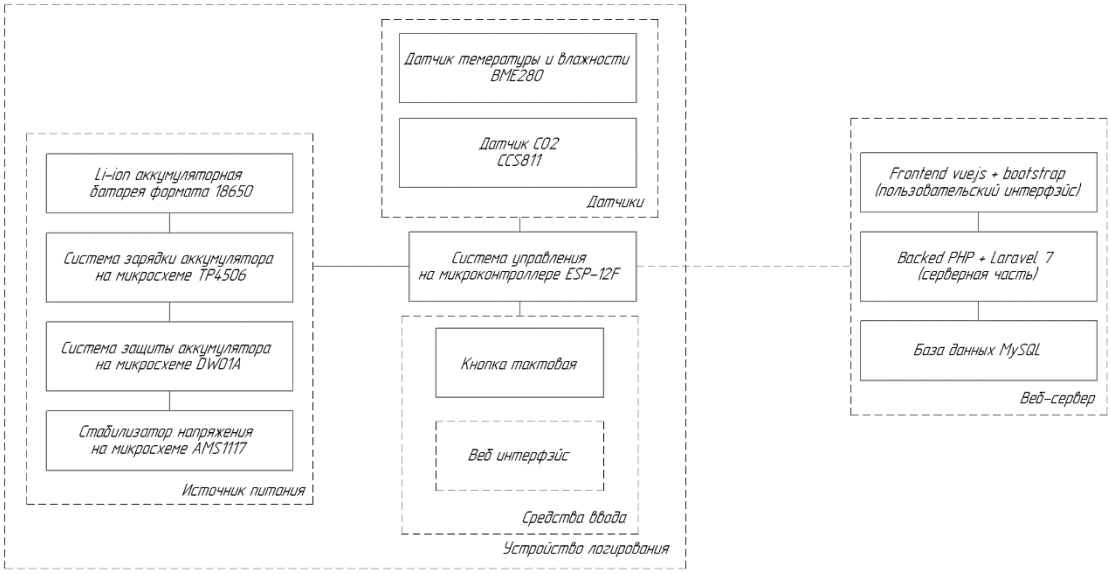


Рисунок 17 - Структурная схема устройства

Вывод

Были подобраны компоненты для системы: микроконтроллер, датчики, источник питания, система защиты и зарядки аккумулятора, средства ввода, определены их ключевые характеристики, некоторые из компонентов были выбраны в результате компромиссного решения, была составлена итоговая структурная схема системы.

4 Описание разработки ПО для МК

Обычно, для того чтобы микроконтроллер выступал как система управления, в его внутреннюю память необходимо записать программу, которая будет исполнятся. Для написания программы под микроконтроллер ESP-12F можно использовать разные языки программирования, такие как Micropython, Lua, JavaScript и др. Однако, в связи с тем, что большая часть кода самого SDK (пакета разработки) написана на C++ и C, было решено использовать именно данные языки программирования. Так же для разработки было решено использовать фреймворк (программную платформу) Arduino[2].

4.1 Обзор инструментов разработки

Для разработки программы, в качестве IDE (интегрированной среды разработки), было решено применить связку из Visual Studio Code (VS Code) и расширения PlatformIO. VS code, их интерфейс представлен на рисунках 18 и 19

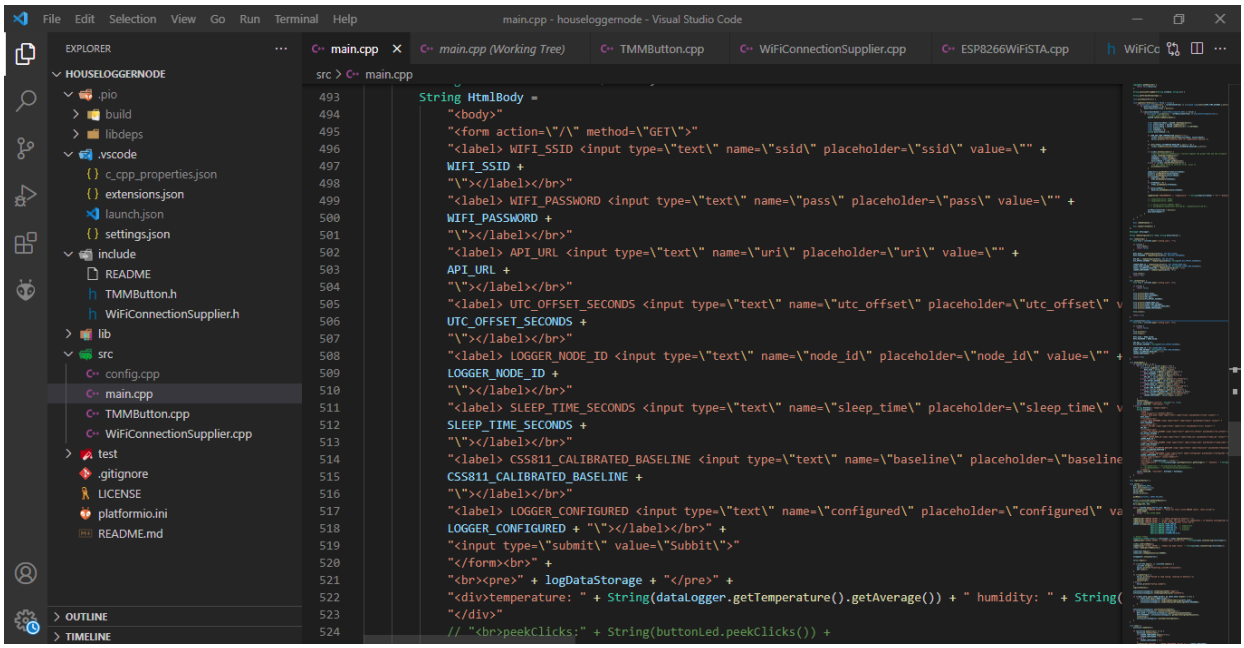


Рисунок 18 - Изображение интерфейса среды разработки VS Code

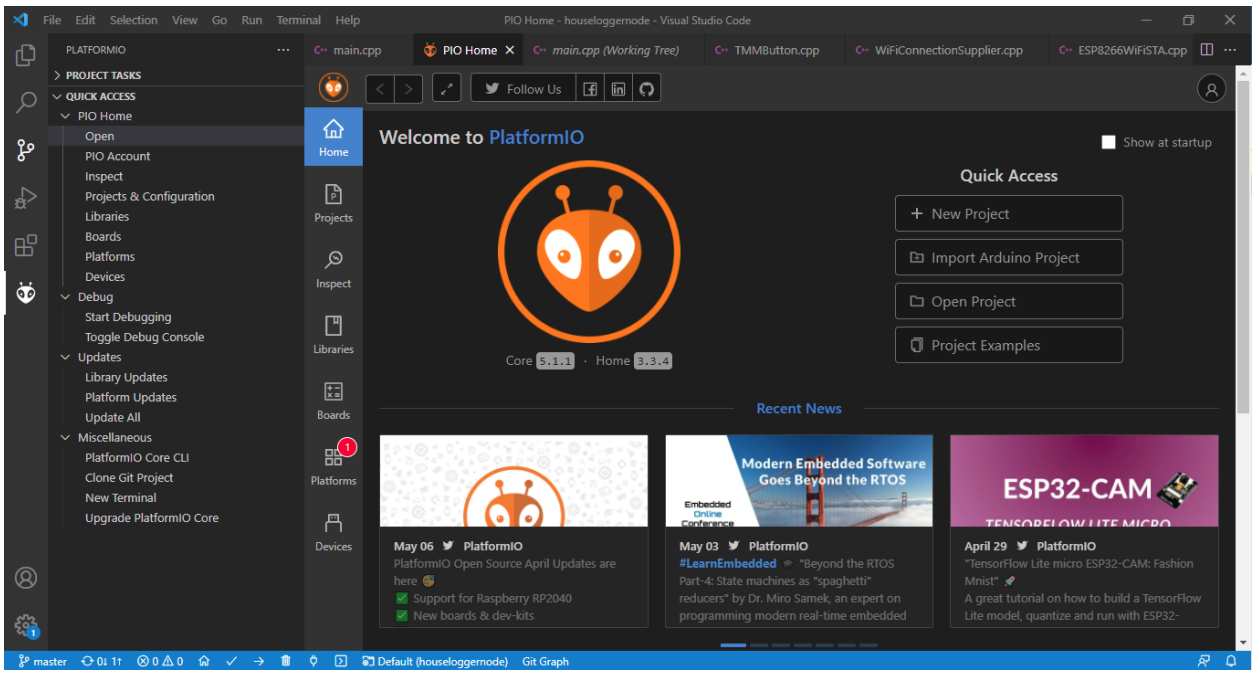


Рисунок 19 - Изображение интерфейса расширения PlatfromIO

Данные инструменты разработки позволяют быстро создавать проект, не заботясь о загрузке SDK, дополнительных библиотек и их сопутствующих инструментов, но и так же позволяют включать сторонние библиотеки кода лишь одной строчкой в конфигурационном файле.

Так же при разработке было принято решение использовать систему контроля версий (СКВ) Git вместе с веб сервисом хранения кода и совместной разработки GitHub. Т.к. работа над разработкой кода производилась лишь одним участником, то было принято решение не использовать какую-либо модель рабочего процесса, например gitflow или же githubflow, а делать запись изменений в репозиторий (коммит) непосредственно в ветку master, с внесением поясняющего комментария.

4.2 Обзор алгоритма работы

Согласно данным рассмотренным ранее был составлен алгоритм работы программы, представленный на рисунке 20 ниже

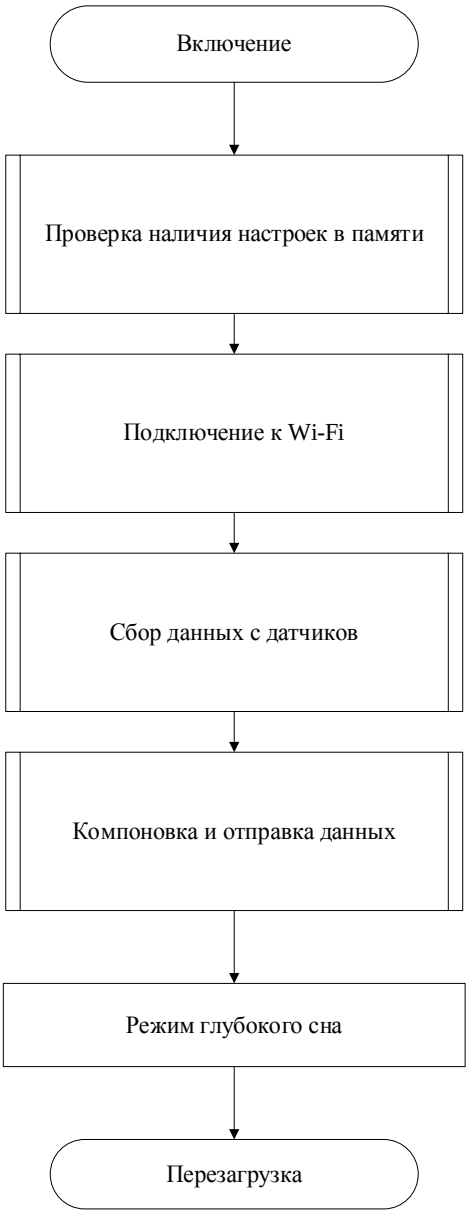


Рисунок 20 - Алгоритм работы программы (общий вид)

Далее, согласно алгоритму, была разработана сама программа.

В данной программе были использованы сторонние библиотеки:

- ArduinoJson – Библиотека позволяющая работать с JSON
- Adafruit BME280 – Библиотека, позволяющая по шине i2c получать данные у устанавливать режим работы датчика BME280
- SparkFun CCS811 - Библиотека, позволяющая по шине i2c производить обмен данными и устанавливать режим работы датчика CCS811
- NTPClient – Библиотека, позволяющая при наличии интернет-подключения синхронизировать время

Так же для, в связи с наличием опыта и ранних наработок в разработки под фреймворк Arduino, были написаны и подключены собственные классы:

- TMMButton – отвечает за обработку событий нажатия клавиши
- WiFiConnectionSupplier – отвечает за установку Wi-Fi соединения
- LogingData – отвечает за хранение данных одного, определённого типа (например, температуры)
- DataLogger – отвечает за обработку данных, хранимых в экземплярах класса LogingData и дальнейшую их отправку

При разработке кода были применены стандартные для фреймворка Arduino библиотеки: Arduino, ArduinoJson, ESP8266HTTPClient, ESP8266HTTPUpdateServer, ESP8266WebServer, ESP8266WiFi, FS, LittleFS, NTPClient, SPI, WiFiUdp, Wire [16],[20].

4.3 Проверка наличия настроек в памяти

После включения микроконтроллер выполняет множество задач, такие как: инициализация Wi-Fi модуля, датчиков, выводов GPIO, веб сервера, клиента синхронизации времени, установка их режима работы. Кроме всего прочего происходит инициализация файловой системы (ФС) LittleFS.begin() и её форматирование LittleFS.format(), в случае если ФС повреждена или не инициализирована. После её удачной инициализации, происходит попытка загрузки данных настроек, что указано в алгоритме на рисунке 21, ниже

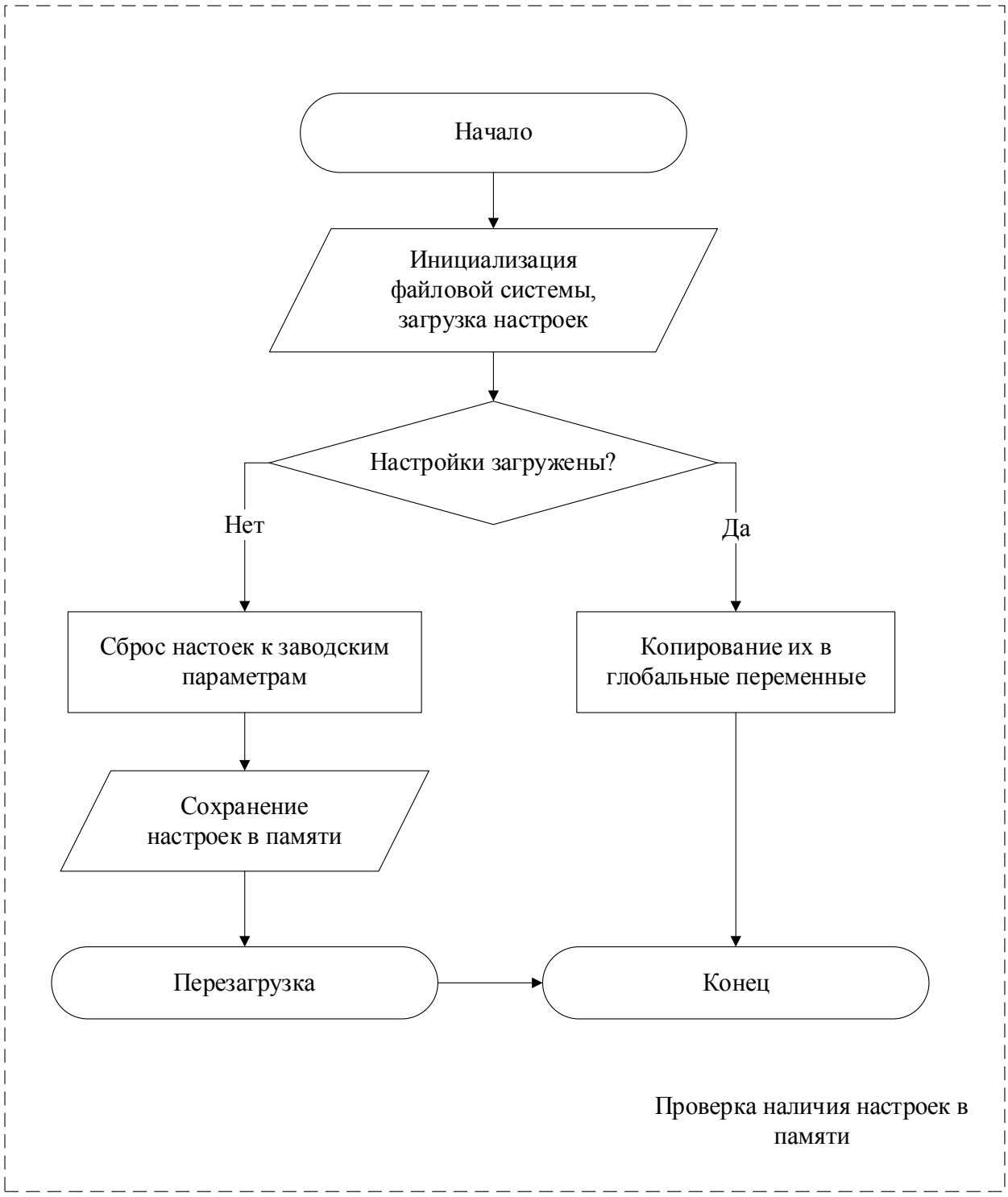


Рисунок 21 - Схема алгоритма проверки наличия настроек в памяти

Согласно данному алгоритму, происходит обращение к файлу config.json с помощью функции LittleFS.open(), далее, происходит построчное считывание данных из файла с помощью функции

readStringUntil('\n'), в глобальные переменные. После того как все данные будут считаны файл закрывается с помощью функции file.close().

4.4 Подключение к Wi-Fi

После инициализации и загрузки настроек, согласно общему алгоритму, вызывается процедура подключения к Wi-Fi, её логику реализует написанный мною класс WiFiConnectionSupplier, из экземпляра которого вызывается функция startConnectionSetup(), часть логики которого можно увидеть на рисунке 22, ниже

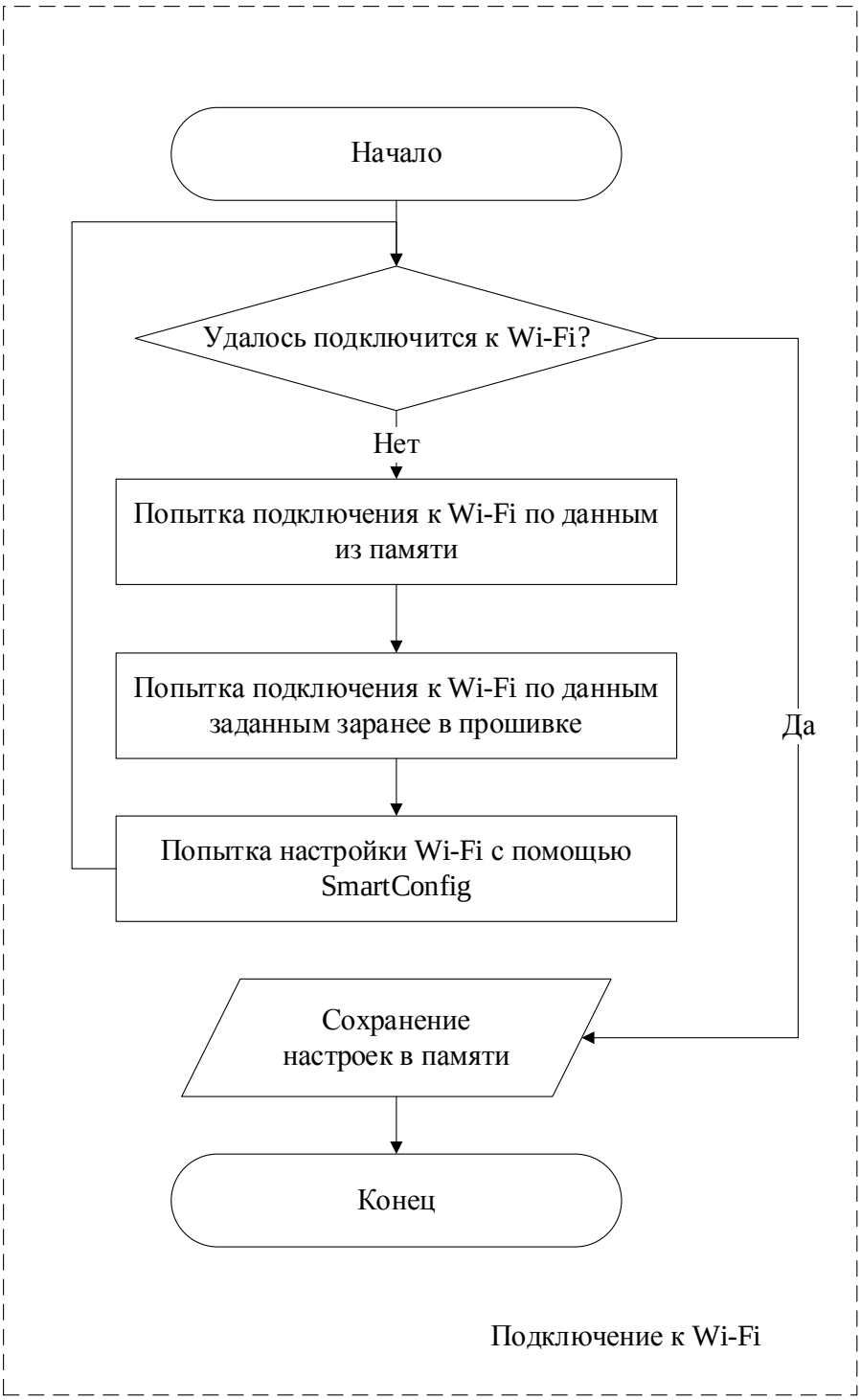


Рисунок 22 - Схема алгоритма подключения в Wi-Fi

Прежде всего производится проверка, если ли подключение к Wi-Fi, с помощью условия WiFi.status() != WL_CONNECTED, если же оно установлено, то происходит пропуск всех остальных процедур и сохранение текущих настроек в памяти.

В противном случае, если соединение ещё не установлено поэтапно, в цикле, производится попытка подключения несколькими способами.

Если при загрузке настроек, SSID и пароль точки доступа был успешно загружен, то происходит попытка подключится по этим данным, с помощью функции startConnectionAttemptsToWiFiFromStorage(), у данной функции имеется заранее заданное время, 60с, по истечение которого, ожидание прекращается, и происходит выход из функции, передавая управление следующей функции.

Если попытка подключения к Wi-Fi точке доступа по данным из памяти, по тем или иным причинам неудачна, вызывается следующая функция startConnectionAttemptsToEmergencyAP(), её логика работы схожа с предыдущей функций, разница заключатся в том, что данные, по которым происходит попытка подключения, заранее задаются в коде, и считаются «резервными». Время же самой попытки подключения ограничено – 10с.

После попыток подключения этими двумя методами, вызывается функция startSmartConfig(), она отвечает за настройку подключения к Wi-Fi методом SmartConfig, по протоколу ESP-Touch [15].

Данный метод отличается тем, что данные подключения, с помощью ESP-Touch могут быть получены из вне, например с помощью мобильного приложения [34].

Таким образом, сначала происходит ожидание получения данных по протоколу ESP-Touch, в течении 120 с, если в течении этого времени данные не были получены, цикли повторяется и вновь происходит попытка подключения к Wi-FI по данным из памяти.

Если же данные были получены, то происходит их проверка, путём попытки подключения к Wi-Fi по ним, попытка длится 60 с, по истечении которых происходит возврат в начало цикла, если данные полученные по ESP-Touch верны, и соединение установлено, происходит сохранение

настроек во внутреннюю память и переход к основному в Arduino циклу void loop().

4.5 Исследование проблемы при подключении к Wi-FI

Для подключения к Wi-Fi используется собственный специальный класс, WiFiConnectionSupplier, написанный ранее, однако при работе с ним возникли проблемы. Ранее данный класс применялся в проекте для ESP-32, которая имеет очень схожий вызов функций, а соответственно совместимость и изменить его исходный код было достаточно просто. После удачной компиляции и запуска получившийся программы было обнаружено, что по некоторой причине микроконтроллер прекращает выполнение программы после чего выводит трассировку стека в последовательный порт, что представлено на рисунке 32, ниже

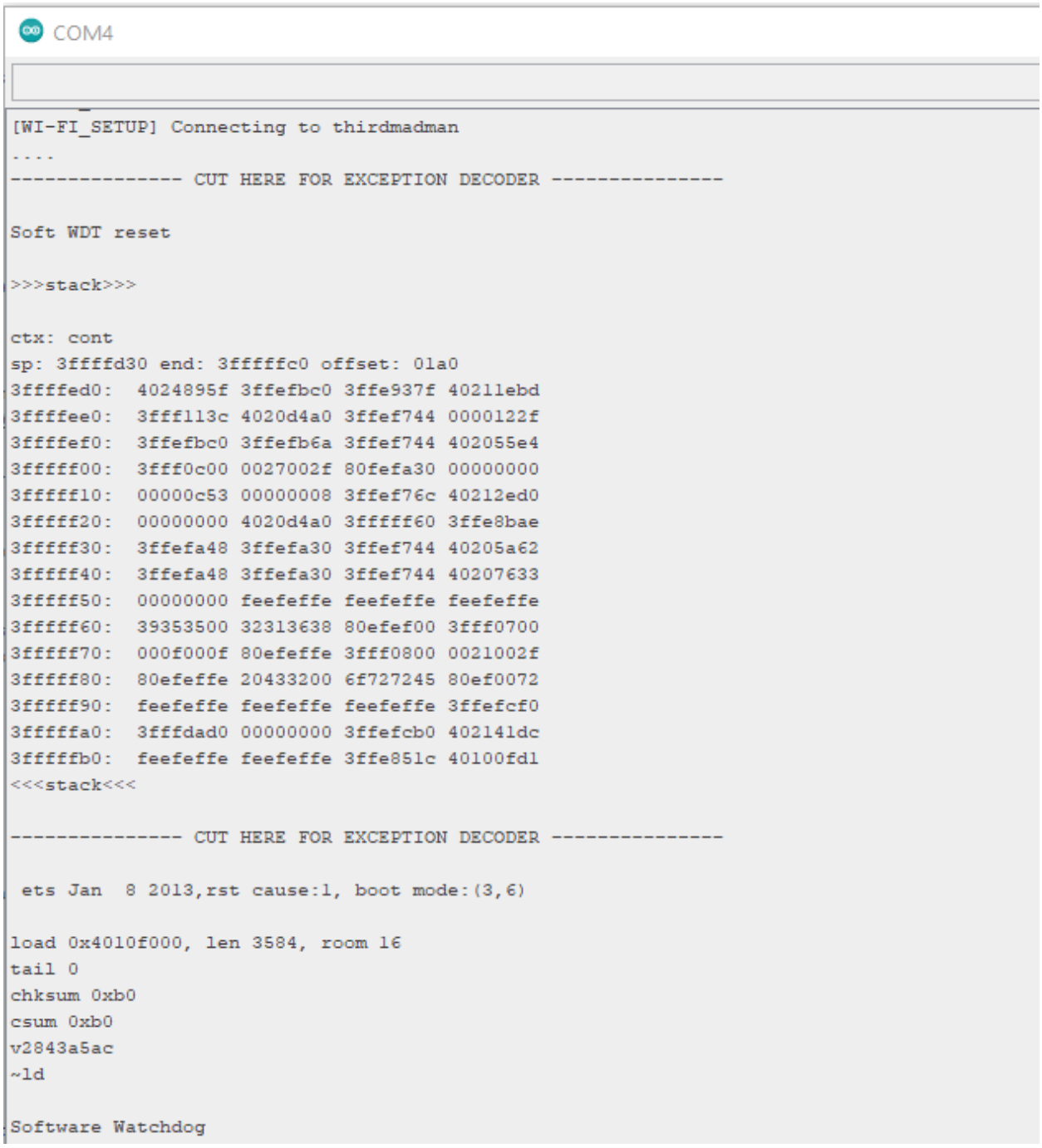


Рисунок 23 - Изучение проблемы подключения к Wi-FI, снимок экрана последовательного порта при возникновении проблемы

Как видно на рисунке 23, микроконтроллер сообщает что произошло событие «Soft WDT reset», затем выводит трассировку стека, после чего видны строки сообщающие информацию о старте, загрузчика, и далее первым сообщением после этих срок идёт сообщение «Software Watchdog», это результат того, что в начале программы, после инициализации

последовательного интерфейса идёт вывод причины перезагрузки, с помощью вызова функции ESP.getResetReason(). Оба сообщения «Soft WDT reset» и «Software Watchdog» имеют один смысл, и говорят о том, что был задействован сторожевой таймер [37] «аппаратно реализованная схема контроля над зависанием системы», хотя в данном случае он реализован программно.

Фреймворк Arduino в ESP8266 работает на основе RTOS (Real-Time Operating System, с англ. Операционная Система Реального Времни), она имеет закрытый исходный код и большая её часть недоступна для прямого обращения к ней. Именно в данной ошибке проявляется работа RTOS и её программного сторожевого таймера.

Целью исследования стало выяснение причин данного поведения и устранение проблем, которые мешают подключению к Wi-FI.

В качестве метода исследования было решено использовать подход «от простого к сложному», то есть было решено провести серию экспериментов, которые бы показали на каком этапе написания программы при её усложнении возникла проблема.

Изначальный код функции вызывавшей проблемы представлен ниже

```
bool WiFiConnectionSupplier::startConnectionAttemptsToWiFiFromStorage() {
    bool out = false;
    if (!ssidFromStorage.equals("") && ssidFromStorage.length() >= 1) {
        Serial.println(serialLoggingPrefix + "Starting connection to wifi by data from storage, ssid: " + ssidFromStorage + " password: " + paswordFromStorage);
        unsigned long lastSerialLoggingUpdateTime = 0;
        const unsigned long serialLoggingUpdateInterval = long(500);

        WiFi.disconnect();
        if (WiFi.getPersistent() == true) WiFi.persistent(false);
        WiFi.mode(WIFI_OFF);

        WiFi.mode(WIFI_STA);
```

```
        Serial.println(serialLoggingPrefix + "Connecting to " +
ssidFromStorage);
        WiFi.begin(ssidFromStorage.c_str(), paswordFromStorage.c_str());

        unsigned long connectionAttemptSmartTime = millis();
        while (1) {
            if (WiFi.status() != WL_CONNECTED) {
                if (millis() >= lastSerialLoggingUpdateTime +
serialLoggingUpdateInterval) {
                    Serial.print(".");
                    lastSerialLoggingUpdateTime = millis();
                }
                if ((millis() >= (connectionAttemptSmartTime +
smartConfigWiFiConnectionAttemptDuration)) || (WiFi.status() ==
WL_CONNECT_FAILED)) {
                    Serial.println("\n" + serialLoggingPrefix + "Connection
to Wi-Fi by saved data attempt timeout");
                    WiFi.disconnect();
                    out = false;
                    break;
                }
            } else {
                Serial.println();
                Serial.println(serialLoggingPrefix + "Successfully connected
to Wi-Fi by saved data");
                logConnectionData();
                out = true;
                break;
            }
        }
    }
    return out;
}
```

Первым делом было решено проверить возможность подключения микроконтроллера к Wi-Fi, ранее она уже тестировалась, однако это было обязательным этапом.

Для тестирования был использован код, представленный в официальной документации микроконтроллера [14].

Он представлен ниже:

```
#include <ESP8266WiFi.h>

void setup()
{
  Serial.begin(115200);
  Serial.println();

  WiFi.begin("network-name", "pass-to-network");

  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {}
```

Его запуск показал, что проблемы с оборудованием отсутствуют и сам микроконтроллер может подключится к Wi-FI. Из этого был сделан вывод, что проблема была все же в самом коде.

Далее было решено приступить к постепенному усложнению кода, и вставки в него фрагментов из функции, описанной ранее. Таким образом код приобрёл следующий вид:

```
#include <ESP8266WiFi.h>

void setup() {
    Serial.begin(115200);
    Serial.println();

    WiFi.disconnect();
    if (WiFi.getPersistent() == true) WiFi.persistent(false);
    WiFi.mode(WIFI_OFF);

    WiFi.mode(WIFI_STA);
```

```
    WiFi.begin("network-name", "pass-to-network");

    Serial.print("Connecting");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println();

    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
}

void loop() {}
```

Вставленные строчки имеют определённый необходимый функционал:

WiFi.disconnect(); - отключает Wi-Fi модуль от точки доступа

if (WiFi.getPersistent() == true) WiFi.persistent(false); - проверяет включена ли запись параметров Wi-Fi во внутреннюю память и если да, то отключает. Это необходимо для того, чтобы уменьшить количество циклов записи во FLASH память, а соответственно увеличить продолжительность её работы, т.к. количество циклов записи ограничено.

WiFi.mode(WIFI_OFF); - полностью отключает Wi-Fi модуль, это необходимо для того чтобы убедится что микроконтроллер отключился от любой точки доступа и завершил все предыдущие процессы связанные с Wi-Fi.

WiFi.mode(WIFI_STA); - вновь включает Wi-Fi и переводит его в режим станции, чтобы затем подключится к точке доступа

После запуска данного кода микроконтроллер вновь смог подключится к Wi-Fi без проблем. Тогда имеющийся код был усложнён до следующего кода:

```
#include <ESP8266WiFi.h>

void setup() {
```

```
    Serial.begin(115200);
    Serial.println();

    unsigned long lastSerialLoggingUpdateTime = 0;
    const unsigned long serialLoggingUpdateInterval = long(500);
    unsigned long connectionAttemptSmartTime = millis();

    WiFi.disconnect();
    if (WiFi.getPersistent() == true) WiFi.persistent(false);
    WiFi.mode(WIFI_OFF);

    WiFi.mode(WIFI_STA);

    WiFi.begin("network-name", "pass-to-network");
    while (1) {
        if (WiFi.status() != WL_CONNECTED) {
            if      (millis()      >=      lastSerialLoggingUpdateTime      +
serialLoggingUpdateInterval) {
                Serial.print(".");
                lastSerialLoggingUpdateTime = millis();
            }
            if   ((millis()   >=   (connectionAttemptSmartTime   +   60000))   ||
(WiFi.status() == WL_CONNECT_FAILED)) {
                Serial.println("Connection  to  Wi-Fi  by  saved  data  attempt
timeout");
                WiFi.disconnect();
                break;
            }
        } else {
            Serial.println();
            Serial.println("Successfully connected to Wi-Fi by saved data");
            break;
        }
    }
    Serial.println();

    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
}

void loop() {}
```

В данном случае были добавлены строки отвечающие за ожидание подключения к Wi-Fi, и отсечки по времени. После их добавления проблема повторилась. Стало понятно, что проблема заключалась где-то в них. Тогда был принято решение вновь упростить получившийся код, чтобы, поэтапно усложняя выяснить какая конкретно строчка вызывает проблему. Код приобрёл следующий вид:

```
#include <ESP8266WiFi.h>

void setup() {
    Serial.begin(115200);
    Serial.println();

    unsigned long lastSerialLoggingUpdateTime = 0;
    const unsigned long serialLoggingUpdateInterval = long(500);
    unsigned long connectionAttemptSmartTime = millis();

    WiFi.disconnect();
    if (WiFi.getPersistent() == true) WiFi.persistent(false);
    WiFi.mode(WIFI_OFF);

    WiFi.mode(WIFI_STA);

    WiFi.begin("network-name", "pass-to-network");
    while (1) {
        if (WiFi.status() != WL_CONNECTED) {
            Serial.print(".");
        }
    }
    Serial.println();

    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
}

void loop() {}
```

После его запуска проблема повторилась. Тогда из этого кода были убрана строка «Serial.print(".");» стало понятно, что проблема в самом цикле.

Тогда само условие цикла было изменено, на то что тестировалось ранее и код приобрёл следующий вид:

```
#include <ESP8266WiFi.h>

void setup() {
    Serial.begin(115200);
    Serial.println();

    unsigned long lastSerialLoggingUpdateTime = 0;
    const unsigned long serialLoggingUpdateInterval = long(500);
    unsigned long connectionAttemptSmartTime = millis();

    WiFi.disconnect();
    if (WiFi.getPersistent() == true) WiFi.persistent(false);
    WiFi.mode(WIFI_OFF);

    WiFi.mode(WIFI_STA);

    WiFi.begin("network-name", "pass-to-network");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
    }

    Serial.println();

    Serial.print("Connected, IP address: ");
    Serial.println(WiFi.localIP());
}

void loop() {}
```

Однако и данное измене не помогло решить проблему исходный код был ещё раз проанализирован и было выяснено что разница заключается в наличии вызова функции delay(500), которая приостанавливает выполнение программы на 500 мс. После добавления в код и проверки подключения к Wi-Fi, проблем обнаружено не было. Однако использование функции delay считается некоренным решением и обычно применяется только для прототипирования т.к. оно прерывает выполнение всей программы.

После поиска в сети интернет по множеству ресурсов связанных с ESP8266, было обнаружено несколько упоминаний данной проблемы [24], [25], [13].

Из них было выявлено, что одно из самых важных отличий классического микроконтроллера Arduino от ESP8266, это то, что последний выполняет множество служебных функций в фоновом режиме, таких как передача данных в встроенный Wi-Fi модуль, управление стеком IP/TCP и выполнение других функций. Блокирование их выполнения может привести к сбою в работе ESP8266 и последующий перезагрузке. Чтобы избежать этого необходимо отказаться от выполнения длительных блокирующих циклов в коде.

Если же в коде всё-таки содержится такой цикл, можно добавить в него функцию dealy() которая пускай и с задержкой, но позволяет выполнять необходимые задачи микроконтроллера в фоне.

Что более примечательно, разработчики библиотеки Arduino так же реализовали функцию yield(), которая вызывает выполнение фоновых задач.

Таким образом, было выявлено предполагаемое решение проблемы, включено непосредственно в финальный код, полученный в результате исследования, представленный ниже:

```
bool WiFiConnectionSupplier::startConnectionAttemptsToWiFiFromStorage() {
    bool out = false;
    if (!ssidFromStorage.equals("") && ssidFromStorage.length() >= 1) {
        Serial.println(serialLoggingPrefix + "Starting connection to wifi by data from storage, ssid: " + ssidFromStorage + " password: " + paswordFromStorage);
        unsigned long lastSerialLoggingUpdateTime = 0;
        const unsigned long serialLoggingUpdateInterval = long(500);

        WiFi.disconnect();
        if (WiFi.getPersistent() == true) WiFi.persistent(false);
        WiFi.mode(WIFI_OFF);

        WiFi.mode(WIFI_STA);
```

```
        Serial.println(serialLoggingPrefix + "Connecting to " +
ssidFromStorage);
        WiFi.begin(ssidFromStorage.c_str(), paswordFromStorage.c_str());

        unsigned long connectionAttemptSmartTime = millis();
        while (1) {
            yield();
            if (WiFi.status() != WL_CONNECTED) {
                if (millis() >= lastSerialLoggingUpdateTime +
serialLoggingUpdateInterval) {
                    Serial.print(".");
                    lastSerialLoggingUpdateTime = millis();
                }
                if ((millis() >= (connectionAttemptSmartTime +
wifiAttemptDuration)) || (WiFi.status() == WL_CONNECT_FAILED)) {
                    Serial.println("\n" + serialLoggingPrefix + "Connection
to Wi-Fi by saved data attempt timeout");
                    WiFi.disconnect();
                    out = false;
                    break;
                }
            } else {
                Serial.println();
                Serial.println(serialLoggingPrefix + "Successfully connected
to Wi-Fi by saved data");
                logConnectionData();
                out = true;
                break;
            }
        }
    }
    return out;
}
```

В результате его тестирования, проблем выявлено не было и данное решение в последующем было удачно применено в случаях, когда возникала подобная ошибка.

4.6 Сбор данных с датчиков

В теле цикла loop() происходят действия, связанные с обработкой нажатий на кнопку, поступающих запросов на веб сервер, но и так же происходит обработка данных, поступающих с датчиков.

Т.к. для обработки запросов на веб сервер и нажатий кнопки, обходим регулярный вызов функций их обработчиков, дальнейший код был написан в неблокирующем виде, а в класс DataLogger, была так же добавлена функция обработчик updateCurretValues(), для вызова в loop().

Алгоритм сбора данных с датчиков, приведённый в упрощённый вид, представлен на рисунке 24, ниже

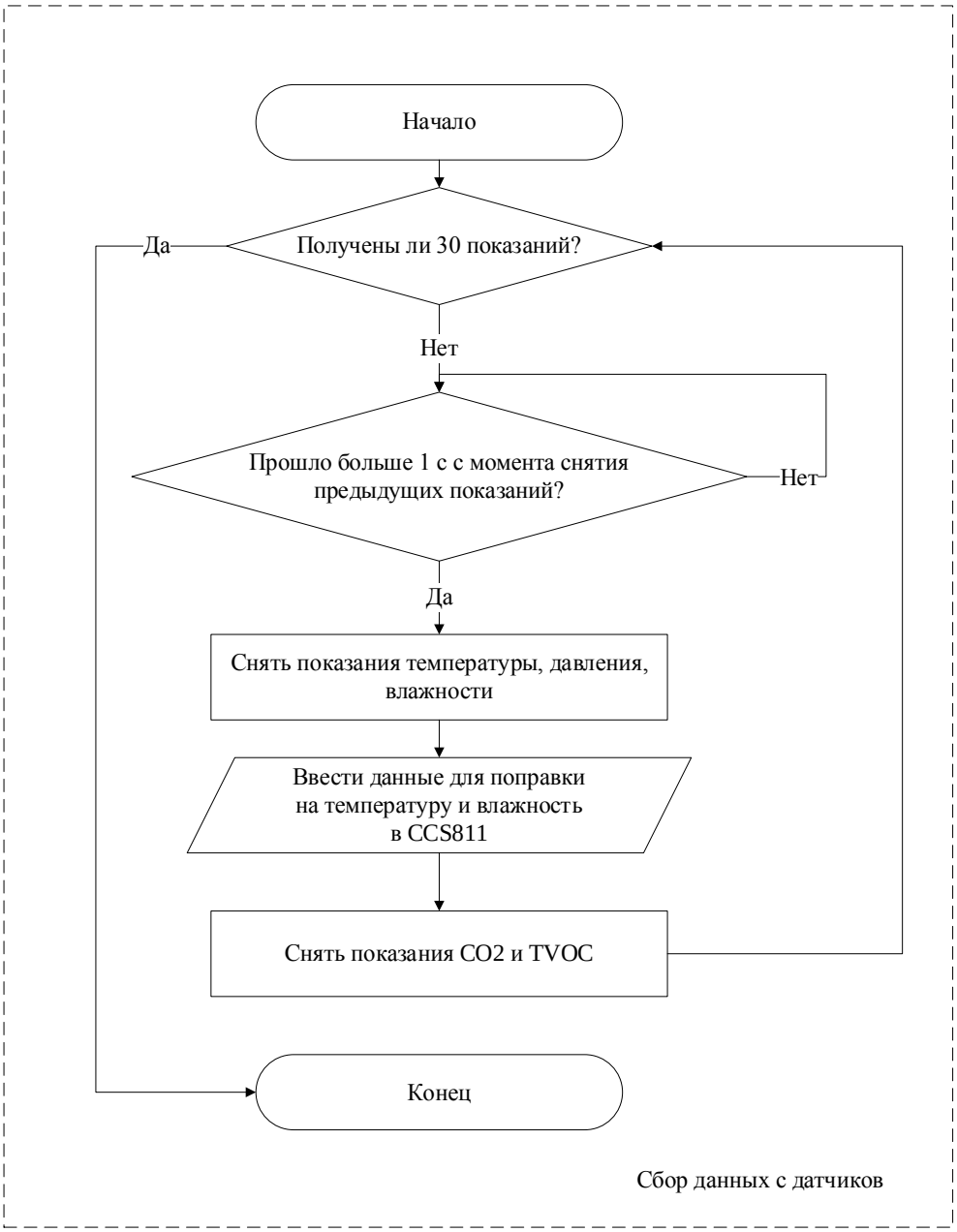


Рисунок 24 - Алгоритм сбора данных с датчиков

Согласно алгоритму, снимается 30 показаний с датчиков. Между снятиями показаний есть задержка в одну секунду.

Задержка между показаниями реализована методом условий – время, в которое были получены последние показания записывается в приватную переменную, внутри экземпляра класса DataLogger, при каждом прохождении через цикл loop() вызывается функция обработчика, в ней и

производится проверка, условия, прошла ли с момента последнего снятия показаний одна (или более) секунды.

Сначала считываются данные с датчика BME280 с помощью функций readTemperature(), readHumidity(), readPressure(), экземпляра класса Adafruit_BME280, после, данные отправляются в датчик CCS811, с помощью функции setEnvironmentalData(), экземпляра класса CCS811, таким образом вводится поправка на температуру и влажность воздуха, что в дальнейшем может обеспечить более точные показания. Затем снимаются данные с самого датчика CCS811, с помощью функций getCO2() и getTVOC(). Все данные запиваются в оперативную память и цикл начинается заново, пока не накопится 30 показаний.

4.7 Компоновка и отправка данных

Так же, в цикле loop() проверяется и необходимость отправки показаний, за этого отвечает функция isNeedToSend(), экземпляра класса DataLogger. Данная функция проверяет отправлялись ли показания ранее, накопилось ли достаточное количество показаний, если данные условия выполняются далее вызывается функция sendCurrentData(). В данной функции все показания усредняются, с помощью библиотеки ArduinoJson создаётся JSON объект, в который записываются текущие показания, далее, с помощью функции serializeJson() происходит преобразование объекта, в строковую переменную. Далее инициализируется HTTP соединение, с помощью экземпляров классов WiFiClient, HTTPClient, задаются заголовки HTTP запросу и происходит отправка методом POST данных, ранее преобразованных в строку, после производится закрытие HTTP соединения. Данную последовательность действий можно наблюдать на рисунке 25, ниже

Рисунок 25 - Схема алгоритма отправки данных

После отправки данных происходит переход в режим Deep sleep (глубокий сон) [10],данный режим позволяет сократить потребление энергии во время отсутствия полезной активности у микроконтроллера, однако, механизм работы данного режима выключает все модули микроконтроллера, кроме модуля RTC (часов реального времени), они необходимы для отсчёта времени работы в этом самом режиме. После истечения времени, RTC посылает сигнал микроконтроллеру на перезагрузку, и вся программа выполняется заново. Несмотря на то, что, со стороны экономии энергии данный режим является самым выгодным - микроконтроллеру в нем

требуется всего 20 мкА, согласно документации [12], применять его во всех частях кода невозможно, это обусловлено тем, что в данном режиме очищается так же и оперативная память.

4.8 Обзор веб интерфейса

Для удобства работы с устройством так же был разработан веб интерфейс, необходимый для установки параметров и обновления программы.

Для его работы используется библиотека ESP8266WebServer. В экземпляр класса ESP8266WebServer с помощью функции on() передаётся функция обработчик, которая вызывается по устанавливаемую пути, например server.on("/", handleBody), где "/" это путь по которому перейдёт клиент, а handleBody это функция, которая будет вызвана в качестве обработчика.

В самой же функции handleBody происходит генерация ответа клиенту, он же в свою очередь зависит от входных аргументов, их количество можно получить, вызвав функцию args().

Таким образом, если аргументов нет, то генерируется ответ в виде HTML станицы содержащий форму ввода настроек. Так же страница содержит текущие показания датчиков. После того как страница была скомпонована, она передаётся клиенту в виде ответа в формате HTTP.

На странице, отдаваемой клиенту, присутствует простая HTML форма, в полях которой выводятся текущие настройки. В эти же поля, с заменой, производится ввод новых настроен. Чтобы настройки были сохранены, необходимо нажать кнопку «Отправить», поле чего, произойдёт отправка всех данных, находящихся в полях ввода, на адрес текущий адрес, методом GET.

Для веб сервера же это выглядит как HTTP GET запрос, с аргументами указанными в полях ввода, в форме. Таким образом, если входные аргументы

присутствуют, то они проверяются на полноту данных, после чего сохраняются во внутреннюю память микроконтроллера.

Веб интерфейс можно увидеть на рисунке 26, ниже

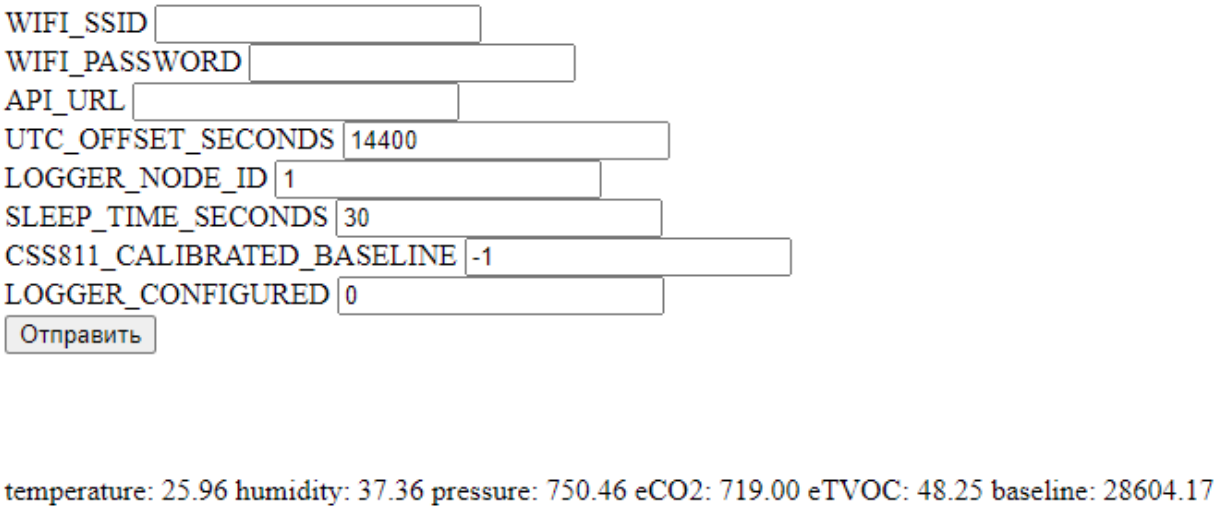


Рисунок 26 - Веб интерфейс для настройки устройства

Сам веб интерфейс выполнен в минималистичном стеле, т.к. любые дополнительные изображения, скрипты, файлы, которые должен отдавать сервер, занимают дополнительное место во внутренний памяти микроконтроллера, а значит уменьшают пространство доступное для реализации логики программы.

Отдельно необходимо обметить наличие возможности обновлений по воздуху (OTA update), которое позволяет обновлять программу микроконтроллера, не подключая к нему USB TTL конвертора, для передачи прошивки. Эта возможность появляется благодаря встроенной библиотеке ESP8266HTTPUpdateServer, для её работы, требуется лишь раз передать в экземпляр класса ESP8266HTTPUpdateServer, ссылку на экземпляр класса самого веб сервера ESP8266WebServer. После запуска веб-сервера, необходимые для её работы пути будут зарегистрированы автоматически. Сама же страница обновления доступна по адресу «/update» и выглядит как показано на рисунке 27, ниже

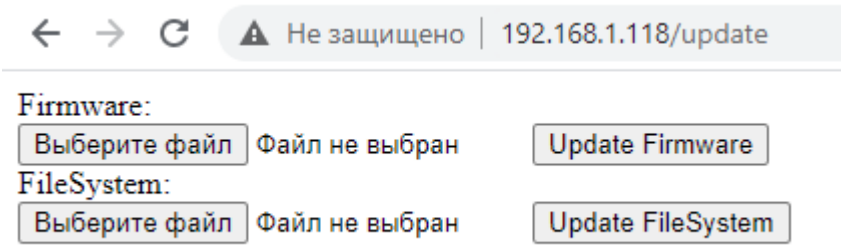


Рисунок 27 - Страница обновления программы микроконтроллера

4.9 Обработка нажатий

Тактовая кнопка, представленная как одно из средств ввода ранее, использует специальный класс TMMButton, в нем присутствует функция обработчик readState(), которая помещается в цикл loop(). Для каждой из кнопок объявляется свой экземпляр класса, а в конструкторе передаётся номер цифрового входа микроконтроллера, к которому подключена кнопка.

При каждом вызове функции обработчика readState(), происходит считывание состояние указанного ранее входа, изменение переменных состояние кнопки, переменных, отвечающих за сохранения времени регистрации последнего фронта сигнала кнопки и спада. Таким образом на одну кнопку можно привязать несколько действий т.к. мы можем узнать не только текущее состояние кнопки – «нажата» или «не нажата» но и время последнего нажатие, время в течении которого кнопка была нажата, количество нажатий время которых не превышает определённые интервалы.

Таким образом, чтобы узнать сколько раз была нажата кнопка, после вызова обработчика вызываются функция peekClicks(), которая возвращает количество «нажатий», отвечающих определённым условиям.

Другим же событием является длительное нажатие кнопки, время в течении которого в последний раз была нажата кнопка, можно получить вызовом функции getPressedTime().

4.10 Исследование способов сокращения энергопотребления

Как ранее упоминалось в данной работе, в одном из вариантов, питание будет автономным, а соответственно целью данного исследования, стало поиск и внедрение способов сокращение до минимума энергопотребления устройства, для увеличения времени продолжительности автономной работы от аккумуляторной батареи.

Прежде всего было необходимо выяснить количество энергии, потребляемой устройством. Для этого было решено произвести эксперимент - устройство было подключено к аккумуляторной батарее через мультиметр, включенный в режим измерения тока с пределом до 200 мА. Было выяснено, что при напряжении на аккумуляторной батарее в 4.2 В, устройство потребляет около 97 мА, а мощность составила 0.4074 Вт. Если использовать аккумуляторную батарею ёмкостью в 3000 мАч, то её хватит приблизительно на 31 час непрерывной работы устройства, однако этот показатель может ухудшится в зависимости от износа батареи.

Для увеличения времени автономной работы было решено выявить части устройства с наивысшим потреблением тока, для этого была произведена серия экспериментов - от устройства были отключены датчики, затем был измерен ток потребления, он оказался приблизительно равен 73 мА, а мощность составила 0,3066 Вт.

Затем был подключён датчик BME280, и уже вместе с ним был измерен так потребления – он оказался равен приблизительно 74 мА, а мощность составила 0,3108 Вт.

После этого был использован блок питания позволяющий регулировать напряжение, и устройство было подключено к нему, однако блок питания был подключен уже после стабилизатора напряжения, таким образом было решено выявить возможные потери на стабилизаторе и других компонентах, касающихся работы с аккумуляторной батареей. На блоке питания было выставлено напряжение 3.3 В, что соответствует линии питания на

устройстве, к которой был подключен блок питания. Было выявлено что ток в таком случае приблизительно равен 69 мА, однако сравнивать данный ток с предыдущими результатами не совсем корректно т.к. напряжение отличается, поэтому результаты так же были пересчитаны в Ватты, потребляемая мощность при этом составила 0,2277 Вт, что несколько меньше, чем без стабилизатора напряжения и других компонентов, однако это разница межу мощностями не столь значительно. В результате было выявлено, что основным потребителем тока является сам микроконтроллер, а соответственно необходимо оптимизировать не конструкцию, а программный код, который управляет этим микроконтроллером.

При изучении официальной документации, было выявлено, что наиболее вероятно большую часть этой мощности потребляет встроенный в микроконтроллер Wi-Fi модуль, это наглядно видно из рисунков 29, 29 представленных ниже:

Parameters	Min	Typical	Max	Unit
TX802.11 b, CCK 11 Mbps, P_{OUT} = +17 dBm	-	170	-	mA
TX802.11 g, OFDM 54Mbps, P_{OUT} = +15 dBm	-	140	-	mA
TX802.11 n, MCS7, P_{OUT} = +13 dBm	-	120	-	mA
Rx802.11 b, 1024 bytes packet length, –80 dBm	-	50	-	mA
Rx802.11 g, 1024 bytes packet length, –70 dBm	-	56	-	mA
Rx802.11 n, 1024 bytes packet length, –65 dBm	-	56	-	mA

Рисунок 28 - Снимок таблицы из документации – Потребление тока радио модулем

Power Mode	Description	Power Consumption
Active (RF working)	Wi-Fi TX packet	Please refer to Table 5-2.
	Wi-Fi RX packet	
Modem-sleep①	CPU is working	15 mA
Light-sleep②	-	0.9 mA
Deep-sleep③	Only RTC is working	20 uA
Shut down	-	0.5 uA

Рисунок 29 - Снимок таблицы из документации –Потребление тока микроконтроллером в разных режимах

Из данных снимков документации видно, что в режиме Modem-sleep, когда Wi-Fi модуль выключен, микроконтроллер потребляет 15 мА, а потребляемый Wi-Fi модулем ток при приёме сигнала может достигать 56 мА.

Было решено проверить потребление тока устройством при выключенном Wi-Fi модуле, для этого в начале функции setup(), был внесён следующий код:

```
WiFi.disconnect();
WiFi.forceSleepBegin();
```

Данный код отключает микроконтроллер от точки доступа и переводит Wi-Fi модуль в режим сна, форсированно.

После этого программа была загружена в микроконтроллер, и устройство вновь было к аккумуляторной батарее и были произведены измерения: при напряжении в 4.2 В, ток составил 34 мА, а потребляемая мощность соответственно 0,1428 Вт. Данное решение может снизить ток потребления в тех интервалах работы, когда устройство не снимает показания, однако даже его ещё недостаточно, ведь устройству все равно необходимо подключатся к Wi-Fi время от времени для передачи данных на сервер.

Из прошлых экспериментов уже было выявлено, что подключение датчика CCS811 значительно увеличивает потребляемую мощность, устройства, поэтому, был решено найти способ уменьшить его потребление.

Дополнительно изучив документацию на датчик [4], было выявлено, что он может работать в нескольких режимах:

Режим 0 – «Холостой ход», слаботочный режим, измерения не производятся

Режим 1 – Режим постоянной мощности, измерение качества воздуха в помещении каждую секунду

Режим 2 – Импульсный режим прогрева датчика, измерение качества воздуха в помещении каждые 10 секунд

Режим 3 – Режим импульсного нагрева с низким энергопотреблением, измерение качества воздуха в помещении каждые 60 секунд

Режим 4 – Режим постоянной мощности, измерение качества воздуха каждые 250 мс

Так же в документации прилагается таблица, фрагмент которой представлен на рисунке 30, ниже, которая описывает потребляемую мощность в каждом из режимов:

Power Consumption	Idle Mode 0 at V_{DD}= 1.8V		0.034		mW
	Mode 1 & 4 at V_{DD}= 1.8V		46		mW
	Mode 2 at V_{DD}= 1.8V		7		mW
	Mode 3 at V_{DD}= 1.8V		1.2		mW

Рисунок 30 - Снимок из документации к датчику CCS811, фрагмент таблицы электрических характеристик

Однако, согласно документации, если перевести датчик в режим 0, то для последующего включения и ввода в нормальную работу (прогрева) ему потребуется не менее 10 минут, поэтому было решено реализовать

использование режима 3 в те время как датчик не используется, а во время снятия измерений использовать режим 1.

Далее было проведён ещё один эксперимент, устройство было подключено к аккумуляторной батарее, Wi-FI модуль микроконтроллера был выключен, датчик CCS811 был переведён в режим 3, были сняты показания потребления тока: при напряжении на батарее 4,2 В, ток составил 20 мА, а потребляемая мощность 0,084 Вт. Таким образом работу устройства условно удалось поделить на «активный режим» в котором работает Wi-Fi, происходит снятие показаний с датчиков и «пассивный режим» в котором данные не снимаются с датчиков, Wi-Fi отключен, CCS811 переводится в режим 3.

Как видно из предыдущего эксперимента, потребление тока можно снизить, и достичь этого можно аппаратно-программными средствами. Так же, в ранее рассмотренной документации на микроконтроллер ESP8266, присутствует один особенный режим работы, в котором, судя по документации затрачивается очень малое количество энергии – DeepSleep.

DeepSleep – это особый режим работы микроконтроллера ESP8266, в котором если рассматривать с технической точки зрения, то микроконтроллер выключается, полностью и в включённом состоянии остаётся только модуль RTC. RTC управляют пробуждением из режима DeepSleep, производя отсчёт времени, которое задаётся перед переводом микроконтроллера в данный режим. После истечения заранее заданного времени, RTC подают сигнал на GPIO16, который в свою очередь должен быть подключен к RESET микроконтроллера, для того чтобы вывести его из режима DeepSleep. Для того чтобы ввести микроконтроллер в данный режим, достаточно вызвать функцию ESP.deepSleep() и в качестве аргумента указать время сна в мк.с.

Далее имеющийся код был модифицирован для использования режима глубокого сна и был произведён эксперимент, в котором был измерен ток

потребления в устройством с микроконтроллером в режиме DeepSleep и датчиком CCS811 переведённым в режим 3, в результате, ток в «пассивном режиме» работы устройства составил 6 мА, при напряжении на входе, 4,2 В, а мощность составила 0,0252 Вт соответственно. Получившаяся схема режимов работы устройства представлен на рисунке 31, ниже

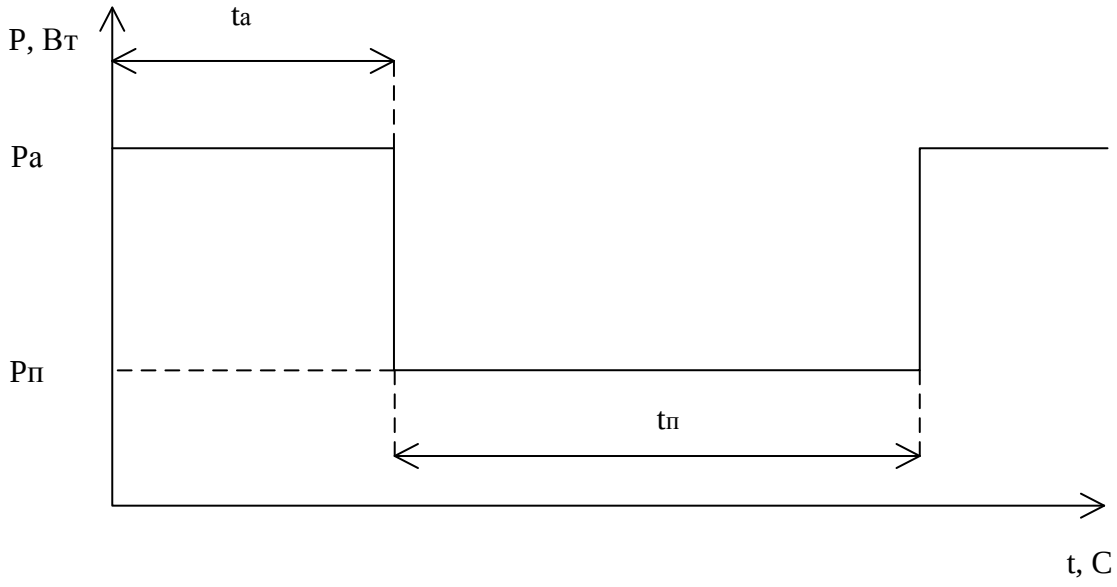


Рисунок 31 - Схема циклов режима работы устройства

Согласно данной схеме, видно, что устройство сначала работает в активном режиме веря $t_{а,}$ потребляя при этом мощность $P_а$, затем переходит в пассивный режим и находится в нём $t_п$, потребляя при этом мощность $P_п$, затем цикл повторяется.

Таким образом можно рассчитать среднюю мощность потребления $P_{ср}$

$$P_{\text{ср}} = \frac{P_{\text{а}} * t_{\text{а}} + P_{\text{п}} * t_{\text{п}}}{t_{\text{а}} + t_{\text{п}}}, \quad (2)$$

Ёмкость же литий-ионного аккумулятора в Вт*ч можно узнать из документации на сам аккумулятор или рассчитать по формуле, зная ёмкость в А*ч, и значение среднего рабочего напряжения аккумулятора:

$$Q_{\text{Вт*ч}} = U_{\text{ср. акб}} * Q_{\text{А*ч}} \quad (3)$$

Время разряда аккумулятора $t_р$ при нагрузке определённой мощности можно рассчитать по формулам, (где $Q_{акб}$ – это ёмкость аккумулятора в Вт*ч):

$$t_р = \frac{Q_{акб}}{P_{ср}} \quad (4)$$

$$t_р = \frac{U_{ср.акб} * Q_{А*ч} * (t_а + t_а)}{P_а * t_а + P_п * t_п} \quad (5)$$

При условии, что устройство ранее уже было подключено к Wi-Fi сети и находится в её зоне покрытия, время работы устройства в активном режиме можно принять за 40 с, среднее рабочее напряжение на литий-ионном аккумуляторе 3,7 В, то остаётся как переменная время работы в пассивно режим. Это время устанавливается самим пользователем, оптимальным его диапазон является 15-60 мин., т.е. 900-3600 с.

Таким образом, до оптимизации программного кода, используя более точный расчёт, при ёмкости аккумуляторной батареи в 3000 мАч и средней потребляемой мощности 0.4074 Вт время работы устройства составило бы 27 ч 15 м. После оптимизации кода, если устройство будет находится в пассивном режиме 900 с (15 мин), то продолжительность автономной работы удастся увеличить до 267 ч 42 мин.

После, этого, для уточнения экспериментальных данных было решено снять осцилограммы тока потребления с помощью цифрового осциллографа. Для этого, в разрыв цепи питания устройства был подключён шунт, состоящий из двух последовательно включённых резисторов в 0,5 Ом каждый с погрешностью ±1%, для получения общего сопротивления в 1 Ом, при таком номинале сопротивлений, напряжение на шунте будет равно протекающему через него току. Далее была снята осциллограмма напряжения на шунте. При изучении осциллограммы было выявлено, что потребление тока устройством имеет импульсный характер, и наиболее вероятно, что некоторые импульсы находятся за пределами частотного диапазона осциллографа. Тогда было решено применить RC фильтр низких частот для получения более точной осциллограммы. В качестве ёмкости был

выбран электролитический конденсатор на 16 В и 1000 мкФ, а сопротивление 3,6 кОм.

$$f_{\text{ср}} = \frac{1}{2\pi RC} \quad (5)$$

Таким, согласно формуле 5 частота среза данного фильтра будет равна 0,044 Гц.

После подключения RC фильтра к шунту, а устройства к аккумуляторной батарее, были сняты осциллограммы токов, одновременно, до фильтра и после, представленные на рисунках 32 и 33. Время работы в пассивном режиме было установлено 30 с.

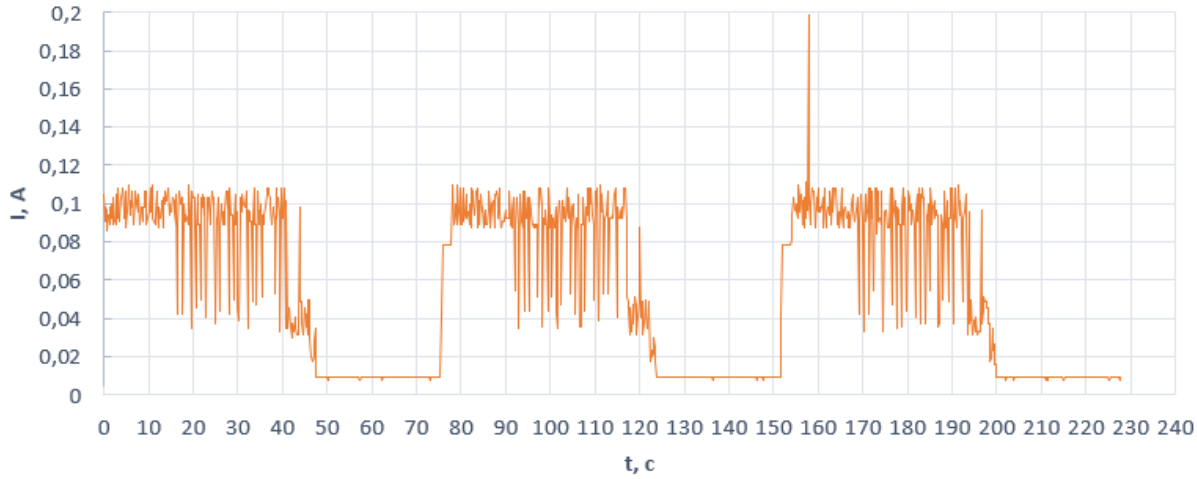


Рисунок 32 - Осциллограмма тока потребления (без фильтра)

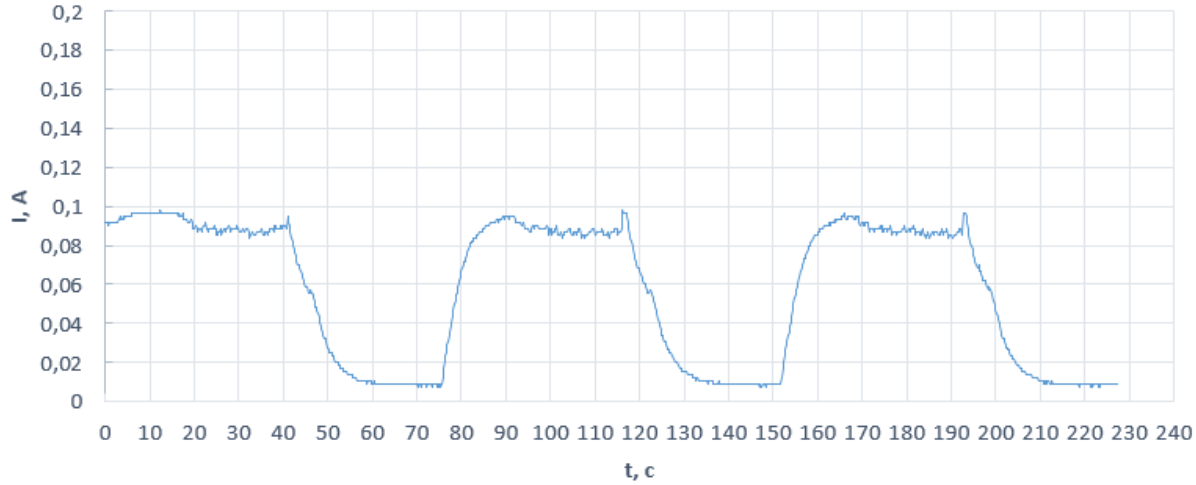


Рисунок 33 - Осциллограмма тока потребления (с фильтром)

Так же, при измерении тока, были обнаружимы моменты пикового потребления длительностью в 0,2 мс, превышавшие остальные показания в 2 раза, но скважность этого сигнала достаточно высока, поэтому он не влияет на средние потребление, что зафиксировано на рисунке 34, ниже

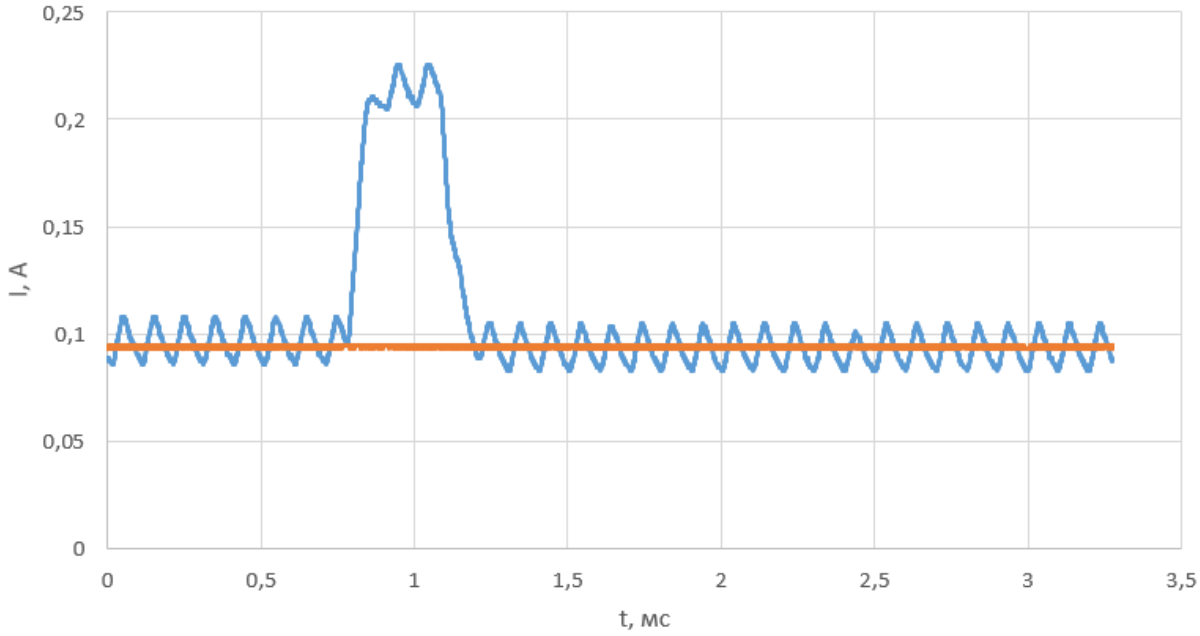


Рисунок 34 - Осциллограмма тока потребления, оранжевым ток после RC фильтра синим ток дол RC фильтра

Согласно осциллограммам, было вычислено среднее потребление в активном режиме, оно составило 87,79 мА, а мощность составила 0,351 Вт. В пассивном режиме среднее потребление тока составило 8,96 мА, а мощность 0,036 Вт. Так же была уточнена продолжительность активного режима его длительность оказалась 48 с.

Таким образом, по уточнённым данным был вновь произведён расчёт средней потребляемой мощности, при условии, что в активном режиме устройство находится 48 с и потребляет 0,351 Вт, а в пассивном режиме 900 с и потребляет при этом 0,036 Вт. Средняя потребляемая мощность получилась 0,052 Вт, соответственно при ёмкости аккумуляторной батареи в 3000 мАч и рабочему напряжению в 3,7 В, общее время работы составило 213 ч 30 мин,

что несколько меньше ранее рассчитанного значения, однако значительно больше чем, время которое смогло бы беспрерывно проработать устройство, до оптимизации кода - 31 ч 40 м.

В результате данного исследования удалось выявить основные потребители тока в устройстве и на основе этого оптимизировать, увеличив тем самым время работы устройства в автономном режиме приблизительно в 6 раз.

Вывод

Для разработки ПО был выбран язык программирования C++, среда разработки VS Code и PlatfromIO, выбран фреймворк Arduino, был составлен алгоритм работы программы.

Была составлена программа для микроконтроллера, которая может подключать микроконтроллер к W-Fi точке доступа, в которой присутствует веб интерфейс настройки и обновления по воздуху, программа так же может получать данные с датчиков, производить их усреднение, компоновку и отправку на сервер. По ходу написания программы были произведены исследования, в результате них были решены проблемы работы программы, а также оптимизирована её работа для увеличения срока службы аккумуляторной батареи.

5 Описание разработки ПО для веб сервера

Для разработки программы веб-сервера было решено использовать язык программирования PHP совместно с фреймворком Laravel 8 [18]. Необходимо отметить, что сама программа разделяется на модули, backend (серверная часть) – отвечает за бизнес-логику, связь с базой данных, генерацию шаблона веб страницы, которая будет отдаваться веб сервером клиенту и frontend, отвечающий за клиентскую часть, которая будет показываться в веб браузере.

Для клиентской же части было решено использовать фреймворки как bootstarp, vue.js и Plotly JavaScript.

5.1 Обзор инструментов разработки

Для разработки кода было решено использовать IDE PhpStorm и его встроенные плагины, такие как .env fiels support, Laravel, Symfony Support, Vue.js. Интерфейс среды разработки можно увидеть на рисунке 35, ниже

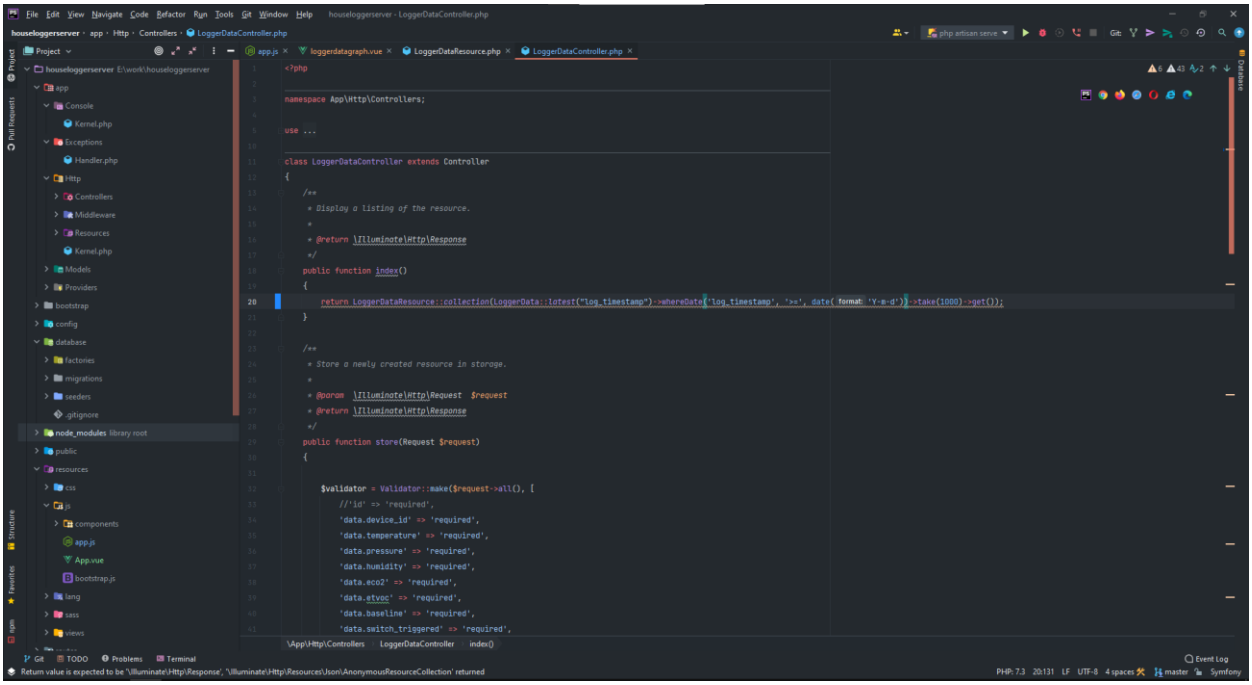


Рисунок 35 - Интерфейс среды разработки PhpStorm

В качестве системы контроля версий так же было решено использовать git, а в качестве удалённого хранилища кода GitHub.

В качестве хранилища данных было решено использовать базу данных (БД) MySQL, вместе с системой управления базами данных (СУБД) phpMyAdmin.

Дополнительным инструментом разработки стал Laragon, это контейнерное решение, предоставляющие возможность быстрого старта локального веб сервера, включающие в себя как основное серверное ПО, так и дополнительные программы, упрощающие разработку. Данное решение было принято не случайно, т.к. у самого фреймворка Laravel 8 если минимальные системные требования, для разработки:

- PHP >= 7.3
- BCMath PHP Extension
- Ctype PHP Extension
- Fileinfo PHP Extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Composer
- NodeJS >= 12.14.0
- MySQL

Весь этот набор расширений и программ присутствует в Laragon что упрощает разработку.

5.2 Обзор серверной части

Для разработки кода был применён шаблон программирования, который разделяет данные, пользовательский интерфейс и управляющую логику, но называется «Модель-Представление-Контроллер» (MVC), схема его работы представлена на рисунке 36, ниже

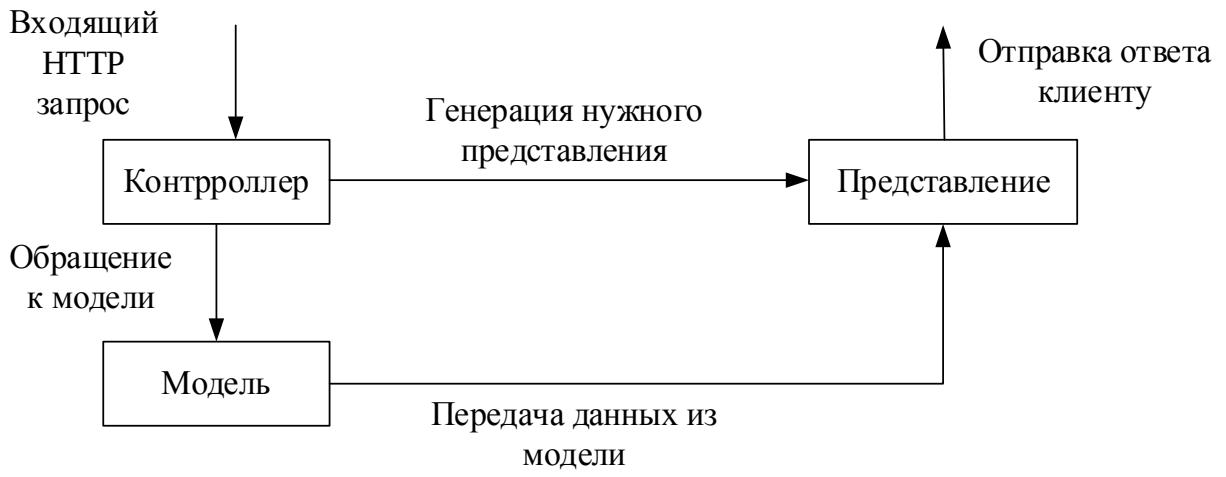


Рисунок 36 - Схема работы шаблона программирования Модель-Представление-Контроллер

Таким образом, при запросе клиентом из браузера определённой страницы, он обрабатывается веб сервером, затем поступает в контроллер, соответствующий пути, по которому поступил запрос от клиента, контроллер передаёт необходимые данные представление, а также если нужно обращается к модели, которая запрашивает данные из БД и затем так же передаёт в представление, которое затем компонуется и отдаётся клиенту.

Так же важно упомянуть, что не всегда необходима генерация HTML страницы, которая будет в понятном человеку, графическом варианте, например при отправке данных устройством, ответ которое получает устройство должен быть наиболее коротким и простым, к тому же в ответе устройству не нужна графическая часть. Обычно в таких случается создаётся отдельная часть веб приложения - программный интерфейс приложения (API), именно с ним взаимодействует устройство логирования и клиентская часть кода.

5.2.1 Описание объектов, их зависимостей

В данном проекте в качестве моделей (объектов) были рассмотрены основные сущности, затем они были объединены в иерархическую цепочку, которую можно наблюдать на рисунке 37, ниже

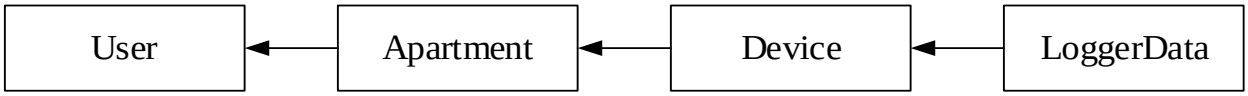


Рисунок 37 - Иерархия моделей

Таким образом User является главенствующий моделью в данной цепочке. Модель User представляет реального пользователя, имеющего emal, пароль, но прежде всего пользователь имеет несколько квартир, которые представлены в виде модели Apartment. В свою очередь в квартире может находиться одно или несколько устройств логирования в разных комнатах, они представлены в виде модели Device. Устройства же в свою очередь отправляют данные на сервер, данные принадлежат конкретному устройству с конкретным id, что и представляет модель LoggerData.

1.1 Описание клиентской части

Для быстрого развертывания клиентской части, а то есть веб интерфейса, который будет видеть пользователь в своём браузере, было решено использовать CSS фреймворк bootstrap. Его основным преимуществом является достаточно большой набор готовых элементов, часто встречающихся на веб сайтах. Подключив данный фреймворк в HTML коде достаточно просто найти необходимый элемент с официального сайта, затем скопировать его себе и изменить описание элементов.

Конечно, не каждый элемент будет идеально подходить под нужды конкретного проекта, однако гибкая система сеток и вёрстки, позволяет применять необходимые стили собственным элементам страницы.

За динамическое изменение страницы же отвечает фреймворк Vue.js. Он получает данные с ранее упомянутого API веб сервера в виде JSON объектов, затем согласно бизнес-логике отрисовывает их в браузере пользователя.

За построение графиков отвечает ещё один фреймворк Plotly JavaScript, он получает данные, загруженные Vue.js с API и затем преобразует их в графики.

Вывод

Для разработки ПО серверной части был выбран язык программирования PHP вместе с фреймворком Laravel, в качестве среды разработки была выбрана интегрированная среда программирования PHPStorm, была применён шаблон программирования MVC. Для разработки клиентской части пользовательского интерфейса был использован HTML, CCS фреймворк bootstrap, js фреймворки Vue.js и Plotly JavaScript.

Благодаря этому была составлена программа, которая может получать данные с устройств, представляющая веб интерфейс для просмотра полученных данных в графическом виде.

6 Описание разработки печатной платы

Для того чтобы объединить все ранее перечисленные технические решения было решено разработать и изготовить печатную которая бы была достаточно компактной и включала в себя все необходимые компоненты для работы. Для изготовления печатной платы сначала необходимо было разработать принципиальную схему.

6.1 Обзор используемых инструментов

Для разработки схемы и печатной платы, было решено использовать IDE EasyEDA. Это бесплатное веб приложение, которое позволяет заниматься разработкой непосредственно в самом браузере, её интерфейс представлен на рисунке 38, ниже

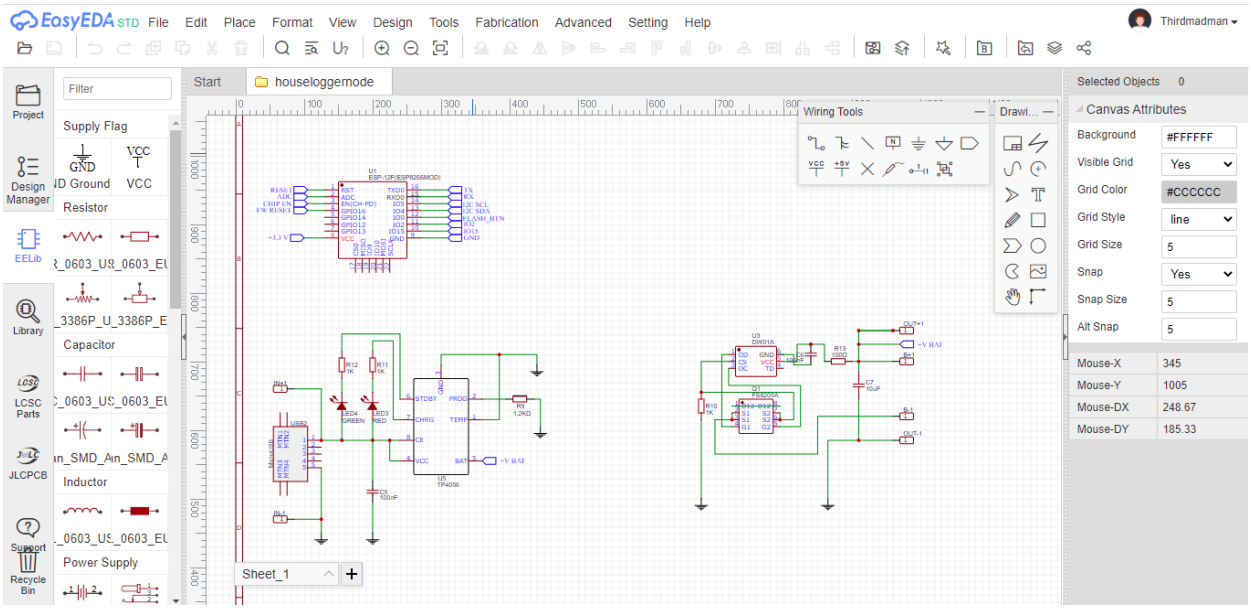


Рисунок 38 - Интерфейс EasyEDA, редактор схем

6.2 Разработка принципиальной схемы

По полученным ранее исходным данным была составлена схема, которая разделена на насколько логические части, которые представлены на рисунках 39-43 ниже

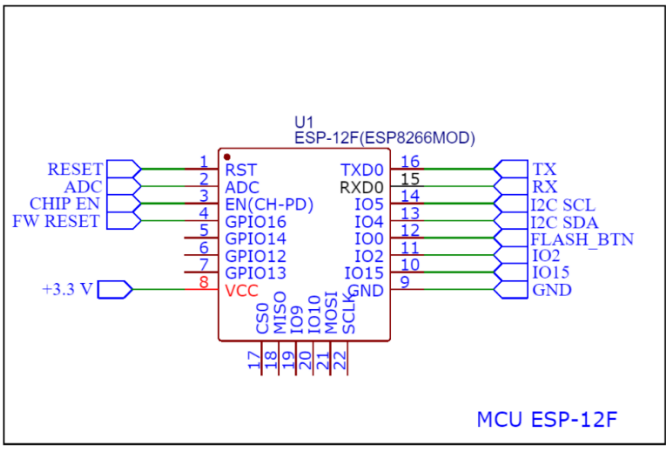


Рисунок 39 - Часть схемы из EasyEDA: микроконтроллер и его подключения

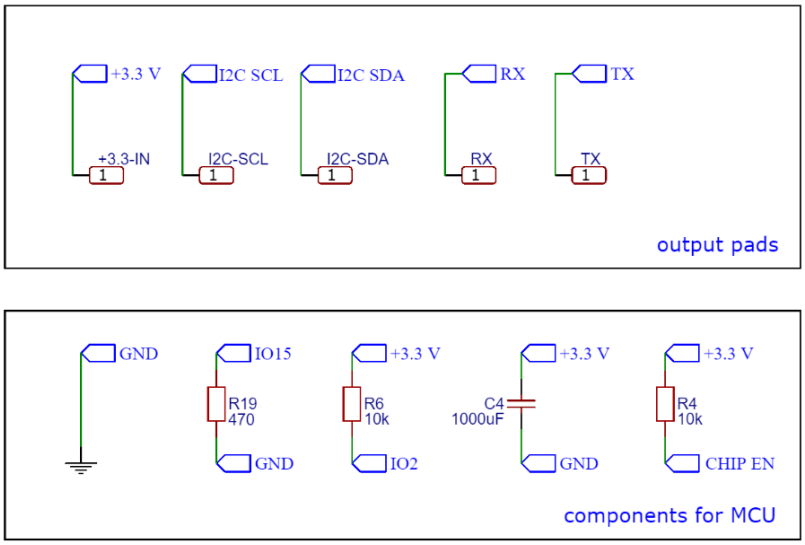


Рисунок 40 - Часть схемы из EasyEDA: сверху – выходные площадки, снизу – пассивные компоненты необходимые для работы микроконтроллера

На данном рисунке изображены резисторы R19, R6, R4, они отвечают за введение в режим загрузки с FLASH памяти [11], C4 является рекомендованным, для снижения пульсаций напряжения на входе.

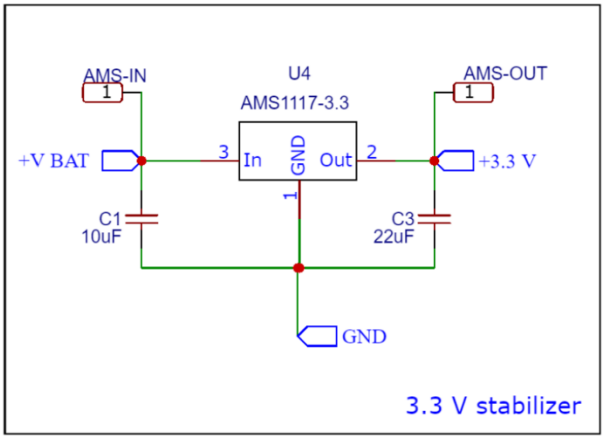


Рисунок 41 - Часть схемы из EasyEDA: стабилизатор напряжения

К стабилизатору согласно его документации подключены конденсаторы C1 на его вход и C3 на его выход, для снижения пульсаций напряжения. Так же ко входу и выходу стабилизатора подлечены выходные площадки AMS-IN и AMS-OUT, нужные для проверки напряжений.

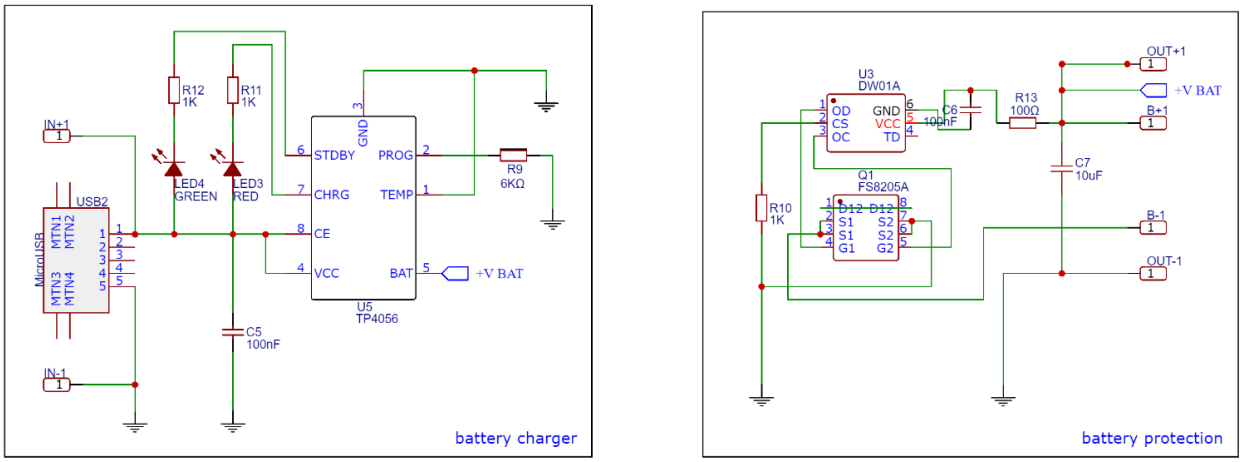


Рисунок 42 - Часть схемы из EasyEDA: слева модуль зарядки, справа модуль защиты аккумулятора

В схеме модуля зарядки R9 является задающим ток зарядки резистором, с рассчитанным ранее в этой работе сопротивлением. R12 и R11 являются токоограничивающими необходимым для корректной работы светодиодов LED4 и LED3 соответственно. Питание на микросхему зарядки поступает из разъёма, имеющего тип microUSB, C5 необходим для фильтрации пульсаций напряжения на входе в микросхему.

В схеме защиты аккумулятора R10 необходим для обнаружения короткого замыкания в цепи, превышения по току и обнаружения тока зарядки. C6 и R13 являются фильтром низких частот, для снижения пульсаций на входе питания DW01A, C7 является выходным фильтром напряжения.

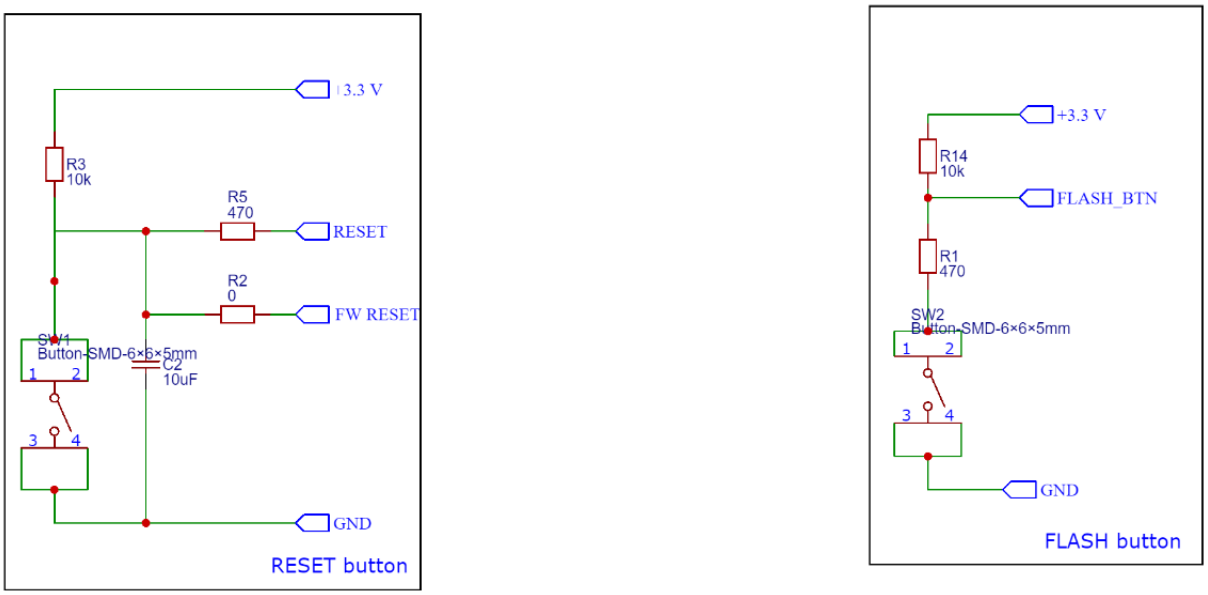


Рисунок 43 - Часть схемы из EasyEDA: слева кнопка перезагрузки, справа кнопка программирования

На данном рисунке видно, что вход RESET имеет токоограничивающий резистор R5, в то время как R3 подтягивает вход микроконтроллера в 3.3 В. Кнопка же после нажатия замкнёт цепь на GND, а C2 обеспечит задержку. R2 же выполняет функцию перемычки – если она впаяна, микроконтроллер сможет обеспечивать режим глубокого сна и последующий выход из него.

Кнопка для выбора режима загрузки микроконтроллера подключена через токоограничивающий резистор R1 ко входу IO0 (тут изображено в виде соединения FLASH_BUTTON), сам же цифровой вход микроконтроллера, по умолчанию подтянут на питание 3.3 В. Данная кнопка несёт двойной функционал, кроме как выбора режима загрузки, во время исполнения основной программы она является средством ввода.

6.3 Разработка схемы печатной платы

Изначально изготовление печатной платы планировалось произвести в домашних условиях с помощью лазерно-утюжного метода (ЛУТ), это могло бы ускорить разработку. Однако уже на этапе проектирования были выявлены проблемы – у ЛУТ метода есть особенности одна из них невозможность изготавливать проводники толщиной менее 0.4 мм. С другой же стороны изготовление двухсторонней платы обычно практически невозможно из-за сложности совмещения трафаретов с обоих сторон платы с достаточной точностью в домашних условиях. Эти два фактора усложнили разводку дорожек печатной платы и сделали невозможном соединение некоторых компонентов. В результате составить печатную плату удалось лишь как показано на рисунке 44, ниже

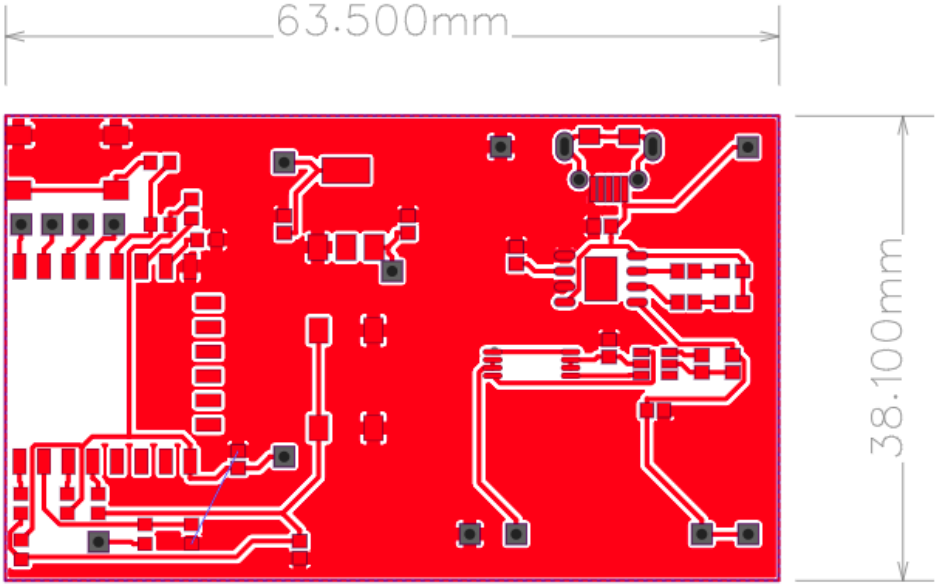


Рисунок 44 - Снимок из редактора печатных плат EasyEDA, первая версия платы

Так же из рисунка 44 видно, что плата получилась достаточно большой по габаритным размерам, что там же не является оптимальным. Однако, несмотря на это всё же были предприняты попытки перенести трафарет

данной печатной платы на стеклотекстолит, наилучший результат из всех попыток представлен на рисунке 45, ниже

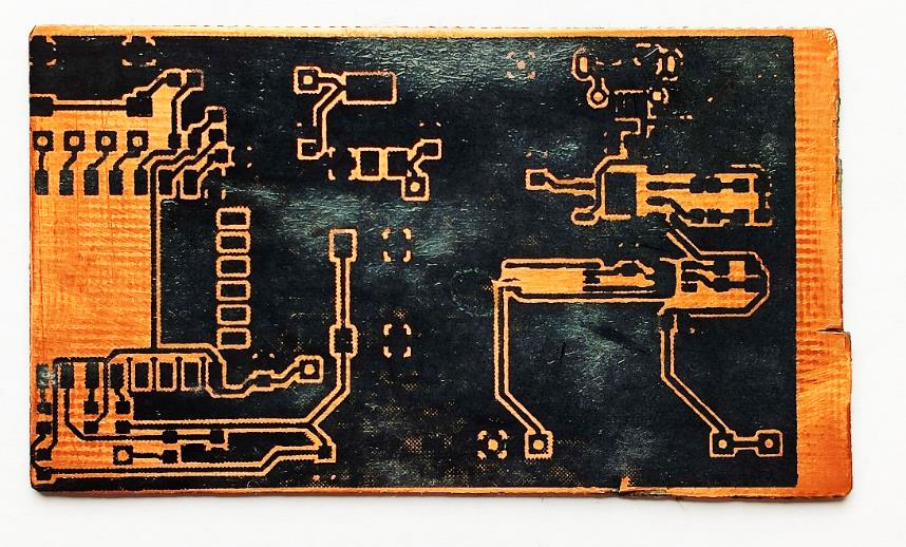


Рисунок 45 - Фотография результат наиболее удачного из всех переносов трафарета на поверхность фольгированного стеклотекстолита

На фотографии видно, что некоторые проводники расположены слишком близко друг другу, но их нет возможности перенести. Наиболее наглядно это прослеживается в правом верхним углу получившегося оттиска, где в последующим должен будет располагаться microUSB разъём. Дорожки bp из тонера, рассчитанные под проводники для этого разъёма, растеклись и соединились между собой, что в последующим бы, после изготовления платы вызвало короткое замыкание и выход и строя устройства.

После этого, было принято решение о заводском производстве печатной платы и была произведена разводка дорожек новой, двусторонней печатной платы, представленной на рисунках 46, 47 ниже

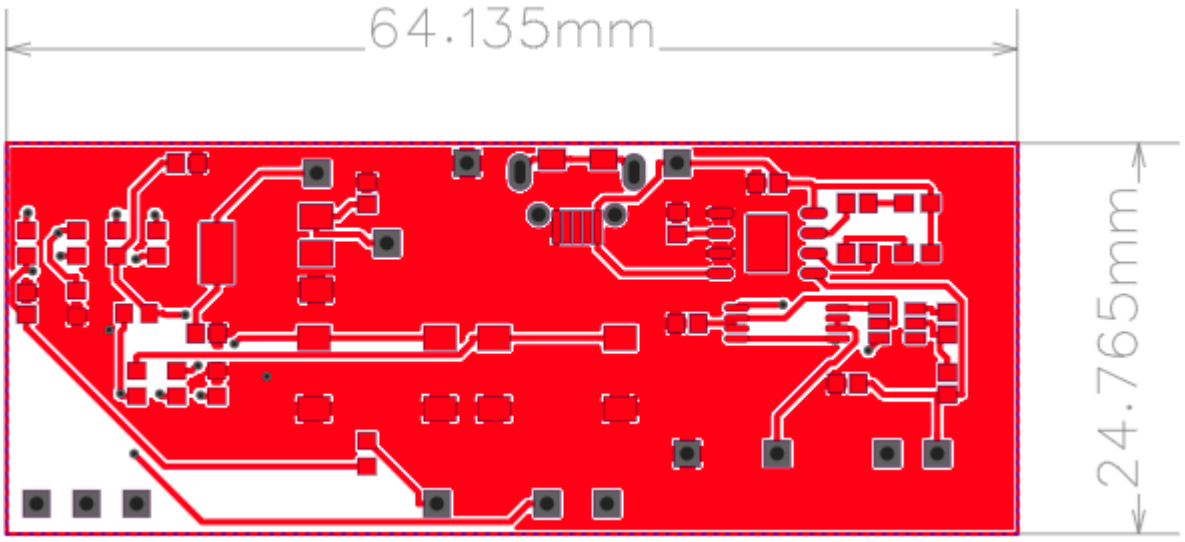


Рисунок 46 - Снимок из редактора печатных плат EasyEDA, вторая версия платы, верхняя сторона

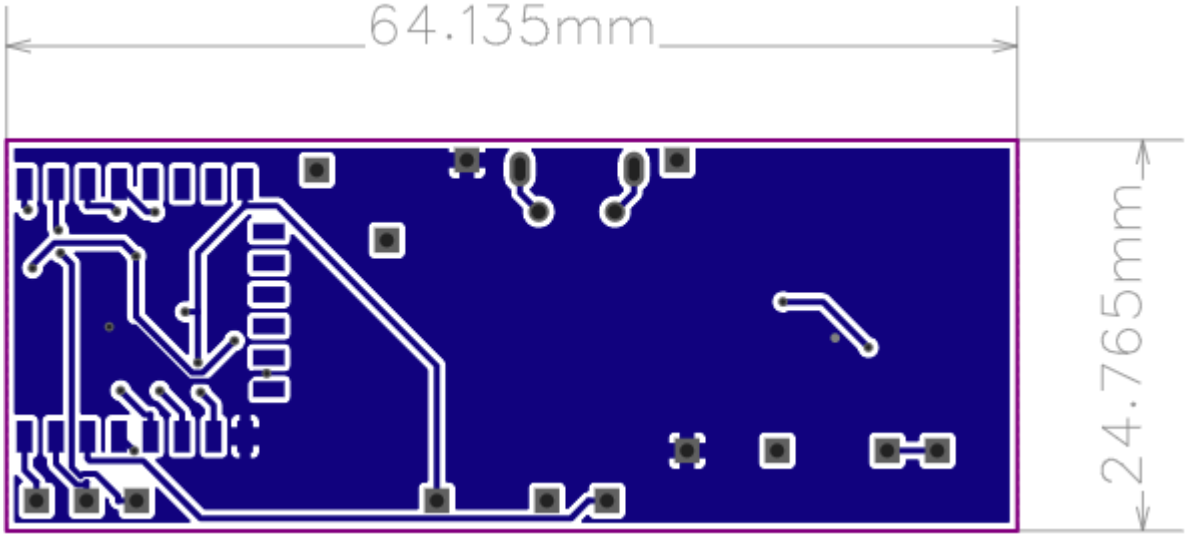


Рисунок 47 - Снимок из редактора печатных плат EasyEDA, вторая версия платы, нижняя сторона

Благодаря двухсторонней разводке дорожек удалось уменьшить габариты печатной платы, прошлая версия имела ширину 38.1 мм, текущая же 24.765 мм, однако длинна платы незначительно увеличилась с 63.5 мм до 64.135 мм.

6.4 Производство печатной платы

Благодаря применению современных средств разработки производство печатной платы упрощается в разы – в EasyEDA достаточно в редакторе печатных плат выбрать меню «Fabrication» и в нём щёлкнуть по пункту «PCB Order» после этого будет произведён переход на партнёрский сайт, где уже будет предложено выбрать параметры печатной платы, способ доставки, оплаты, а также финальную стоимость. Это возможно благодаря генерации в EasyEDA Gerber файла (файл, содержащий в себе всю необходимую информацию для автоматического изготовления печатной платы) и экспорта его непосредственно на сайт партнёра изготовителя печатных плат, однако, так же возможно и просто получить Gerber файл и затем предоставить его для изготовления любому другому производителю печатных плат.

Полученные в результате платы выглядят, как представлено на рисунке 48, ниже

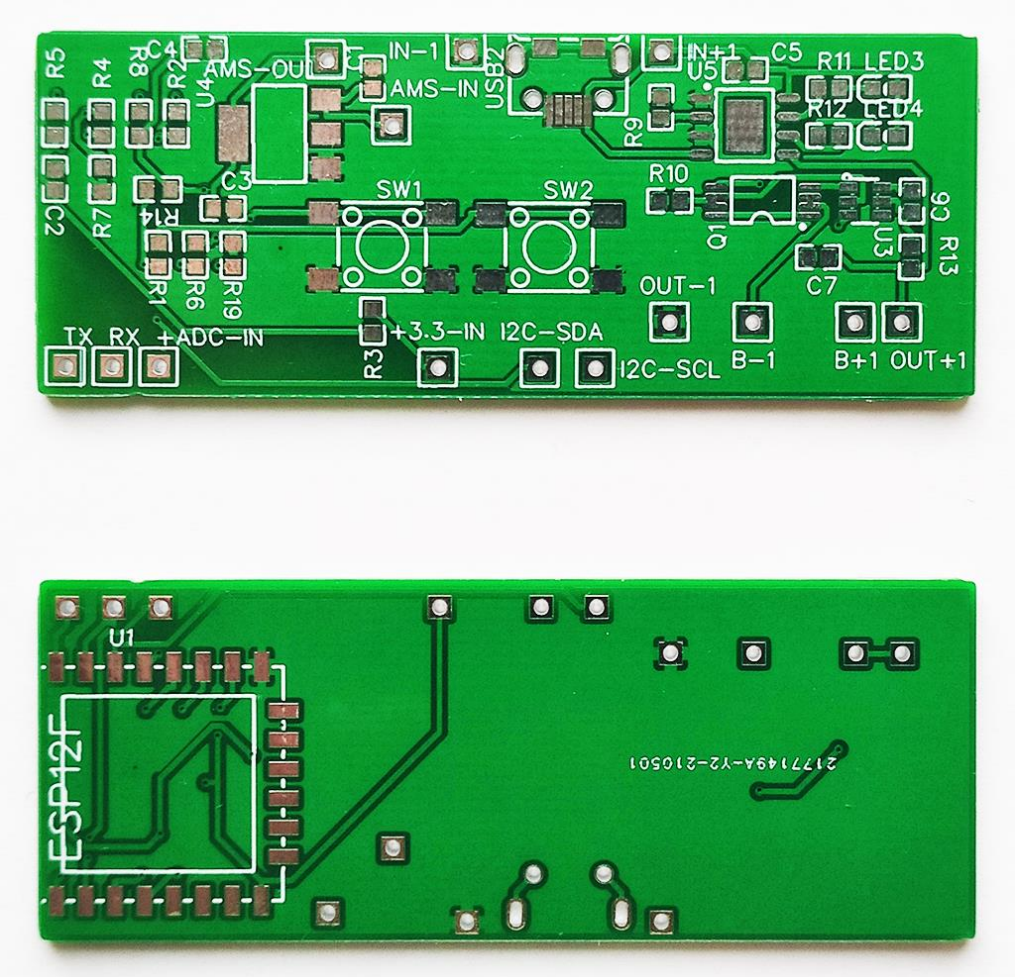


Рисунок 48 - Фотографии полученной в результате производства платы

Всего было получено 5 экземпляров, при первичной визуальной проверке механических повреждений выявлено не было.

Вывод

В результате была разработана принципиальная схема устройства, в веб среде разработки EasyEDA, после чего была предпринята попытка изготовления печатной платы, односторонней, в домашних условиях, полученный результат не удовлетворял требования по изготовлению, поэтому было решено изготовить плату заводским методом, и была произведена разводка уже двухсторонней печатной платы, изготовление которой было заказано на сайте JLCPCB.

7 Сборка устройства и тестирование

На этом этапе работы потребовалось собрать части устройства в единое цело, для этого потребовалось собрать печатную плату, произвести её тестирование, затем впаять остальные компоненты, для дальнейшей записи программы в микроконтроллер.

7.1 Сборка печатной платы и пайка элементов поверхностного монтажа

После того как заказанные печатные платы были доставлены, наступил этап сборки. Все необходимые комплектующие для сборки были куплены ранее или получены путём выпаивания их с плат-доноров с других устройств.

Пайка производилась с помощью паяльной станции с регулировкой температуры, как с помощью паяльника, так и с помощью паяльного фена. При пайке был использован припой компании KAINA, модель B-1, с содержание флюса 2 % и диаметром 0.6 мм, флюс в шприце фирмы AMTECH, тип NV-559-AMS, медная лента для удаления припоя компании REXANT диаметром 1 мм.

Пайка резисторов и конденсаторов производилась с помощью паяльника при температуре 280 °C, со смачиванием поверхности флюсом и добавлением небольшого количества припоя.

Пайка тактовых кнопок SW1 и SW2 и разъёма microUSB производилась при температуре 250 °C без использования дополнительного флюса, это уменьшило вероятность повреждения их пластиковых частей и попадания внутрь флюса.

Пайка остальных элементов поверхностного монтажа (кроме модуля ESP-12F) производилась в два этапа – сначала на все контактные площадки паяльником было нанесено небольшое количество припоя, затем производилось нанесение достаточного количества флюса и с помощью

паяльного фена при температуре потока воздуха 280 °С производился нагрев места пайки с самим компонентом до расплавления припоя. При необходимости поправки положения компонента поверхностного монтажа, использовался пинцет с острыми концами.

Модуль микроконтроллера ESP-12F единственный элемент, который располагается на нижней части печатной платы Его пайка была произведена вручную, с помощью паяльника при температуре 280 °С.

После пайки элементов необходимо приступить к соединению датчиков и батарейного отсека. Для этого было решено использовать кабель ленточный, гибкий, многожильный, 22 AWG (0.33 мм кв.). Для подключения батарейного отсека использовался кабель с двумя проводами, площадка, обозначенная на плате «B+1» подключается к контакту батарейного отсека обозначенным «+», а площадка «B-1» подключается к контакту батарейного отсека «-».

Модули датчиков BME280 и CCS811 подключаются одинаково, согласно таблице 4, ниже

Таблица 4 - Схема подключения датчиков к плате

Обозначение на основной плате	Обозначение на датчике
AMS-OUT	VCC
OUT-1	GND
I2C-SDA	SDA
I2C-SCL	SCL

Важно – при пайке датчика CCS811 необходимо избегать попадания капель или испарений флюса на верхнюю часть платы, где расположен сам датчик и его чувствительная часть. Простой способ заключается в том, чтобы наклеить кусочек скотча на верхнюю часть платы до пайки.

7.2 Тестирование печатной платы

Первое включение устройства было решено производить от регулируемого блока питания. Напряжение на выходе БП было установлено 4.2 В, ограничение по току 0.2 А. Заранее было известно, что устройство не может потреблять боле 0.2 в постоянном режиме, однако при включении было обнаружено что блок питания сразу показал ток в 0.2 А на выходе, что обозначало большую вероятность короткого замыкания. Устройство незамедлительно было выключено от БП, после чего было решено проверить наличие элементов в повреждениях корпуса, как обычно бывает поле выхода из строя электронного компонента – таковых выявить не удалось. Далее было выбран более точный метод определения наличия короткого замыкания на плате – «позвонка» мультиметром, в режиме диодной проверки самого прибора. Было выявлено короткое замыкание на цепи питания батареи, после чего, каждый элемент этой цепи был выпаян, с последующей проверкой сопротивления данной цепи. Последним был демонтирован TP4506, после чего между «B+1» и «B-1» контактных площадок батареи короткого замыкания не наблюдалось, однако оно сохранилось между контактных площадок входа на TP4506 IN-1 и IN+1, что было весьма необычно, однако – складывалось ощущение что короткое замыкание было где-то на самой плате. И это действительно оказалось так, к сожалению, несмотря на заводское производство при изготовлении платы была допущена ошибка, что привело к возникновению проводника, которого быть не должно, что можно наблюдать в сравнении, на рисунке 49

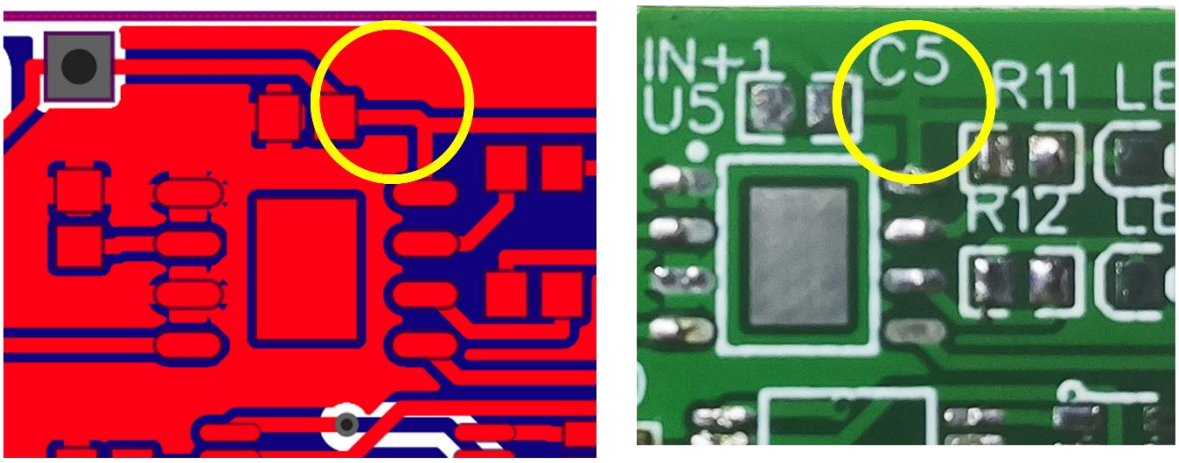


Рисунок 49 - Дефект печатной платы: слева снимок из редактора печатной платы в EasyEDA, то есть что должно быть, справа фотография изготовленной печатной платы

Данный дефект по неизвестной причине не повторился на остальных экземплярах печатных плат. Для исправления данного дефекта был использован канцелярский нож, которым несколько раз было проведён глубокий разрез по поверхности лишнего проводника, что разорвало электрическое соединение и более короткого замыкания не наблюдалось.

После данный процедуры все компоненты были впаяны обратно и вновь было произведено включение в БП платы, короткого замыкания не наблюдалось, микроконтроллер реагировал на перезагрузку с помощью кнопки RESET (SW1).

Далее была произведена компиляция полученного ранее исходного кода с последующей загрузкой его в микроконтроллер с помощью USB TTL конвертора, он был подключён к плате по схеме, указанной в таблице 5, ниже

Таблица 5 - Схема подключения USB TTL конвертора к плате

Контактная площадка на устройстве	Контактная площадка на USB TTL конверторе
+3.3-IN	3v3
OUT-1	GND
RX	TXD
TX	RXD

Ранее, при построении схемы печатной платы в неё были включены все необходимые подтягивающие резисторы для обеспечения режима программирования микроконтроллера (передачи прошивки в FLASH память), поэтому требуется лишь зажать тактовую кнопку SW2 до включения питания, и не отпуская подключить USB TTL конвертор к компьютеру, что автоматически переведёт микроконтроллер в режим программирования.

Загрузку скомпилированного исходного кода в микроконтроллер можно производить как с помощью специализированной программы от производителя - Flash DownLoad Tool, так и с помощью средств строенных в IDE VS Code. Разница заключается в том, что из IDE это удобно делать разработчику т.к. все необходимые инструменты у него уже загружены и установлены вместе с IDE, в то время как Flash DownLoad Tool самодостаточен и не требует других приложений, им удобнее будет пользоваться на заводской линии или в сервисном центре.

7.3 Тестирование устройства

После подготовки устройства, было решено произвести первичное тестирование.

Первым этапом проверки стала возможность подключится к Wi-Fi сети, подключение проверялась на домашней точке доступа, 2,4 ГГц, с помощью приложения для Android со смартфона, по технологии ESP-Touch. Устройство удачно подключилось и определилось в сети беспроводного маршрутизатора. Однако, при тестировании было замечено, что возникают проблемы при подключении к точке доступа, у которой длинна SSID меньше 8 символов. Программа была дополнительно изучена на наличие ошибок в коде и модифицирована, для исправления данной проблемы.

Следующим этапом стала проверка веб интерфейса устройства. Была проверена возможность изменять настройки, отображение текущих снимаемых показаний, возможность обновлять программу устройства по

воздуху. Проблем выявлено не было, все вводимые параметры успешно применялись и сохранялись во внутреннюю память, после перезагрузки устройства или отключения питания они успешно загружались. Функция обновления так же работает без нареканий и после загрузки обновления, устройство перезагружается, считывая уже новую программу из памяти.

После чего были проверены функции, которые были назначены на кнопку: по двойному нажатию устройство успешно переключает состояние параметра, отвечающего за режим настройки, по пятикратно нажатию устройство успешно запустило режим повторной настройки Wi-Fi по ESP-Touch, а по длительному нажатию более 15 с, устройство успешно сбросило настройки к предустановленным.

Далее была проверена работа датчиков, размещение устройства в непосредственной близости от источника тепла увеличило показания температуры в ожидаемых пределах, размещения же устройства в холодильной камере уменьшило текущие показания. После были проверены показания влажности, размещение устройства в камере с силикагелем, ожидаемо уменьшили показания влажности, размещение же устройства в камере с куском ткани, смоченным водой, значительно увеличили показания влажности. Так же было зафиксировано увеличение показателя eCO2 при выдохе на датчик, а при выносе устройства на открытый воздух этот показатель снизился. Так же было зафиксировано изменение показателя tVOC при тлении потушенной спички.

Далее была проверена работа взаимодействия устройства с сервером и отправка данных, устройство успешно подключилось к серверу и отправило ему усреднённые в формате JSON.

Так же была проверена работа веб интерфейса сервера, устройство отображение текущих данных, построение графиков, на этом этапе проверки проблем тоже не возникло.

Вывод

В результате устройство было собрано, протестировано, были выявлены и устранены проблемы в изготовлении печатной платы, после чего в устройство была загружена программа было произведено дальнейшие тестирование его базовых функций, подключение к Wi-Fi, изменение настроек, обновление по воздуху, функции клавиши, передача данных серверу и пользовательский интерфейс веб сервера.

Заключение

В ходе выполнения данной магистерской диссертации был произведён анализ известных в области мониторинга и логирования данных микроклимата в помещениях, были выявлены ключевые параметры разработки системы, были подобраны компоненты для проектируемого устройства, после чего составлена структурная схема.

Был составлен алгоритм для программы устройства, выбрана среда разработки и сопутствующие инструменты, была составлена и отлажена программа для устройства, было произведено исследование особенностей написания программы под конкретное схемотехническое решение, а также исследование, направленное на увеличение продолжительности автономной работы устройства.

Была выбрана среда разработки, необходимые инструменты для написания программы для серверной части кода, был развернут локальный сервер, была отлажена работа как серверной, так и клиентской части кода.

Разработана принципиальная схема устройства, разработана печатная плата в нескольких вариантах, заказана и произведена печатная плата заводского производства.

Собрана, спаяна и протестирована полученная печатаная плата, выявлен и устранён заводской дефект, загружена программа в микроконтроллер, произведено первичное тестирование работы всех компонентов.

В результате проделанной работы в короткие сроки и с малыми затратами временных и финансовых ресурсов получена и протестирована работоспособная платформа, которая в дальнейшем может быть развита в полноценное устройство.

Список используемой литературы

1. 1A Adjustable/Fixed Low Dropout Linear Regulator [Электронный ресурс] // ЧИП и ДИП - интернет-магазин приборов и электронных компонентов: [сайт]. [2021]. URL: https://static.chipdip.ru/lib/552/DOC001552809.pdf (дата обращения: 29.3.2021).

2. Michael Margolis B.J.N.R.W. Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects. O'Reilly Media; 3rd edition (25 Jan. 2016), 2016. 800 pp.

3. BME280 Датчик абсолютного давления, температуры и влажности [Электронный ресурс] // Купить Arduino, модули, датчики Ардуино и другие электронные компоненты: [сайт]. [2021]. URL: https://duino.ru/bme280-datchik-absoljutnogo-davlenija-temperatury-i-vlazhnosti.html (дата обращения: 17.3.2021).

4. CCS811 Ultra-Low Power Digital Gas Sensor for Monitoring Indoor Air Quality [Электронный ресурс] [2016]. URL: https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf (дата обращения: 15.4.2021).

5. DHT12 — I2C датчик влажности и температуры [Электронный ресурс] // MicroPi - Программирование микроконтроллеров, Banana Pi, Orange Pi, Raspberry Pi: [сайт]. [2018]. URL: https://micro-pi.ru/dht12-i2c-датчик-влажности-температуры/ (дата обращения: 19.3.2021).

6. Door/Window Sensor 2 | FIBARO Manuals [Электронный ресурс] // FIBARO Manuals | Smart home automation devices: [сайт]. [2021]. URL: https://manuals.fibaro.com/door-window-sensor-2/ (дата обращения: 7.3.2021).

7. DW01A [Электронный ресурс] // H&Msemi: [сайт]. [2021]. URL: http://hmsemi.com/downfile/DW01A.PDF (дата обращения: 28.3.2021).

8. ESP-01 [Электронный ресурс] // Поставка электронных компонентов | Ультран: [сайт]. [2021]. URL: http://ultran.ru/esp-01-0 (дата обращения: 15.3.2021).

9. ESP-12F [Электронный ресурс] // Поставка электронных компонентов | Ультран: [сайт]. [2021]. URL: http://ultran.ru/esp-12f (дата обращения: 8.3.2201).

10. ESP8266 (NodeMCU) - Как использовать Deep Sleep? [Электронный ресурс] // Smartideal - Веб дизайн, Умный дом, Скрипты для веб разработки, Arduino, Все для гиков: [сайт]. [2018]. URL: https://smartideal.net/kak-ispolzovat-deep-sleep-na-esp8266-nodemcu/ (дата обращения: 15.4.2021).

11. ESP8266 Error Messages And Exceptions Explained – Riktronics [Электронный ресурс] // Riktronics – Welcome to Riktronics!: [сайт]. [2020]. URL: https://riktronics.wordpress.com/2017/10/02/esp8266-error-messages-and-exceptions-explained/ (дата обращения: 16.4.2021).

12. ESP8266 Low Power Solutions [Электронный ресурс] [2016]. URL: https://www.espressif.com/sites/default/files/9b-esp8266-low_power_solutions_en_0.pdf (дата обращения: 16.4.2021).

13. ESP8266 Thing Hookup Guide - learn.sparkfun.com [Электронный ресурс] // Learn at SparkFun Electronics - learn.sparkfun.com: [сайт]. [2021]. URL: https://learn.sparkfun.com/tutorials/esp8266-thing-hookup-guide/using-the-arduino-addon (дата обращения: 12.4.2021).

14. ESP8266WiFi library — ESP8266 Arduino Core 3.0.0-20-g2185f9bd documentation [Электронный ресурс] // Welcome to ESP8266 Arduino Core’s documentation! — ESP8266 Arduino Core 3.0.0-20-g2185f9bd documentation: [сайт]. [2012]. URL: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html (дата обращения: 5.4.2021).

15. ESP-Touch Overview | Espressif Systems [Электронный ресурс] // Wi-Fi & Bluetooth MCUs and AIoT Solutions I Espressif Systems: [сайт]. [2021]. URL: https://www.espressif.com/en/products/software/esp-touch/overview (дата обращения: 1.4.2021).

16. Schwartz M. INTERNET OF THINGS WITH ESP8266. 1st ed. Packt Publishing, 2016. 226 pp.

17. Sinclair B. IoT Inc: How Your Company Can Use the Internet of Things to Win in the Outcome Economy. 1st ed. McGraw-Hill Education, 2017. 304 pp.

18. Stauffer M. Laravel: Up & Running: A Framework for Building Modern PHP Apps. 2nd ed. O'Reilly Media, 2019. 554 pp.

19. Modules | Espressif Systems [Электронный ресурс] // Wi-Fi & Bluetooth MCUs and AIoT Solutions I Espressif Systems: [сайт]. [2021]. URL: https://www.espressif.com/en/products/modules/esp3 (дата обращения: 8.3.2021).

20. Riley M. Programming Your Home Automate with Arduino, Android. Raleigh, North Carolina : The Pragmatic Bookshelf Dallas, 2014. 242 pp.

21. Qingping Technology CGS1 ClearGrass Air Monitor with 2.4G WiFi function User Manual [Электронный ресурс] // User Manual Search Engine: [сайт]. [2018]. URL: https://usermanual.wiki/Qingping-Technology/CGS1 (дата обращения: 1.3.2021).

22. SHT30, датчик влажности и температуры I2C вых 3% 3-5В | купить в розницу и оптом [Электронный ресурс] // ЧИП и ДИП - интернет-магазин приборов и электронных компонентов: [сайт]. [2021]. URL: https://www.chipdip.ru/product0/8675462603 (дата обращения: 19.3.2021).

23. NanJing Top Power ASIC Corp. TP4056 1A Standalone Linear Li-lon Battery Charger with Thermal Regulation in SOP-8 2018. URL: https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf (дата обращения: 25.3.2021).

24. wemosD1 Soft WDT reset. - Using Arduino / Project Guidance - Arduino Forum [Электронный ресурс] // Arduino Forum: [сайт]. [2018]. URL: https://forum.arduino.cc/t/wemosd1-soft-wdt-reset/563647/7 (дата обращения: 7.4.2021).

25. Вопрос - Переподключение к wifi | Сообщество ESP8266 [Электронный ресурс] // Сообщество ESP8266: [сайт]. [2018]. URL:

https://esp8266.ru/forum/threads/perepodkljuchenie-k-wifi.3149/ (дата обращения: 10.4.2021).

26. Грандиозное тестирование батареек / Блог компании Madrobots / Хабр [Электронный ресурс] // Все публикации подряд / Хабр: [сайт]. [2014]. URL: https://habr.com/ru/company/madrobots/blog/364773/ (дата обращения: 21.3.2021).

27. Датчик газа MQ-135 (углекислый газ, аммиак, бензин, дым) | AmperMarket.kz [Электронный ресурс] // Интернет-магазин Arduino и радиодеталей AmperMarket.kz: [сайт]. [2021]. URL: https://ampermarket.kz/sensors/gas/air-quality-sensor-mq-135/ (дата обращения: 19.3.2021).

28. Датчик качества воздуха CCS811 / купить в RoboShop [Электронный ресурс] // RoboShop: [сайт]. [2021]. URL: https://roboshop.spb.ru/sensors/ccs811 (дата обращения: 20.3.2021).

29. Датчик качества воздуха SGP30 [Электронный ресурс] // CompactTool.ru - магазин модулей разработки и паяльного оборудования: [сайт]. [2021]. URL: https://compacttool.ru/datchik-kachestva-vozdukha-sgp30 (дата обращения: 20.3.2021).

30. Логгер iPlug TH температура и влажность (одноразовый) [Электронный ресурс] // Логгеры - купить температурный логгер, регистратор влажности по доступной цене в Москве и РФ: [сайт]. [2021]. URL: https://iloggers.ru/logger-vlazhnosti (дата обращения: 1.3.2021).

31. Логгер данных температуры и влажности (Многоразовый) [Электронный ресурс] // Логгеры - купить температурный логгер, регистратор влажности по доступной цене в Москве и РФ: [сайт]. [2021]. URL: https://iloggers.ru/index.php?route=product/product&product_id=108 (дата обращения: 3.3.2021).

32. Логгеры (регистраторы) данных testo 184 Руководство по эксплуатации [Электронный ресурс] // Мой диск – Google Диск: [сайт].

[2020]. URL: https://drive.google.com/file/d/1mLueRJ53ThSyL44EtLtTYSqdoYeCjx6P/view (дата обращения: 2.3.2021).

33. Монитор качества воздуха HONEYWELL HAQ руководство пользователя [Электронный ресурс] // iG-store - магазин инноваций: [сайт]. [2021]. URL: https://ig-store.ru/files/products/3956/11d4edca149b3a082166a1a09b7d90560e9a0228.pdf (дата обращения: 1.3.2021).

34. Приложения в Google Play – EspTouch: SmartConfig for ESP8266, ESP32 [Электронный ресурс] // Google Play: [сайт]. [2019]. URL: https://play.google.com/store/apps/details?id=com.khoazero123.iot_esptouch_demo (дата обращения: 2.4.2021).

35. Современные типы литиевых батарей и их использование [Электронный ресурс] // Магазин электросамокатов и др. электротранспорта "Electric-Wheels": [сайт]. [2021]. URL: https://electric-wheels.ru/batterie/tipy-li-ion-akkumulyatorov (дата обращения: 22.3.2021).

36. Румянцев А. Способы заряда Li-ion аккумуляторов и батарей на их основе // Журнал Компоненты и технологии об электронных компонентах, датчиках, микросхемах, микроконтроллерах, светодиодах, DSP. 2012. URL: https://kit-e.ru/wp-content/uploads/136119.pdf (дата обращения: 25.3.2021).

37. Сторожевой таймер — Википедия [Электронный ресурс] // Википедия — свободная энциклопедия: [сайт]. [2016]. URL: https://ru.wikipedia.org/wiki/Сторожевой_таймер (дата обращения: 2.4.2021).