

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно–
вычислительных систем (КИБЭВС)

ИНТЕРНЕТ ВЕЩЕЙ

Отчет о выполнении промежуточного аттестационного этапа группового
проектного обучения (ГПО)
Проект ГПО – КИБЭВС–1904

Ответственный исполнитель
проекта:
Студент гр. 731–2
_____ А.Д. Коноваленко
«__» декабря 2024 г.

Проверил:
Руководитель проекта
Старший преподаватель каф.
КИБЭВС

_____ О.В. Пехов
(оценка) (подпись)
«__» декабря 2024 г.

Принял:
Ответственный за ГПО на кафедре
Доцент каф. БИС, к.т.н.
_____ И.А. Рахманенко
«__» декабря 2024 г.

Исполнители проекта ГПО КИБЭВС–1904:

Студент гр. <u>731–2</u>	_____ Е.В. Демиденко
Студент гр. <u>731–2</u>	_____ А.Д. Коноваленко
Студент гр. <u>722–1</u>	_____ Д.М. Ведениктов
Студент гр. <u>722–1</u>	_____ Д.В. Захаров
Студент гр. <u>722–1</u>	_____ А.С. Москвичеков
Студент гр. <u>722–1</u>	_____ М.М. Таганов

ф. ГПО–06 Дополненная

**Министерство науки и высшего образования Российской
Федерации**

Федеральное государственное автономное образовательное
учреждение высшего образования

**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ**

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Групповое проектное обучение

УТВЕРЖДАЮ

Зав. кафедрой КИБЭВС

Шелупанов Александр Александрович

«_____» _____ 20__ г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на выполнение проекта № КИБЭВС–1904

1. Основание для выполнения проекта: приказ № 3247ст от 26.06.2019.
2. Наименование проекта: Интернет вещей.
3. Цель проекта:
4. – изучение технологий модуляции LoRa и сетевого протокола LoRaWAN;
– создание IoT–сети, основанной на данных технологиях.
5. Основные задачи проекта на этапах реализации:
– изучение технологий модуляции LoRa и сетевого протокола LoRaWAN;
– изучение документации оборудования, которое будет использовано для реализации IoT–сети;
– произвести настройку данного оборудования;

– создание работоспособной IoT–сети.

6. Научная новизна проекта: Нет.

7. Планируемый срок реализации: Получение результатов ожидается к январю 2025 г.

8. Целевая аудитория (потребители): IoT–разработчики.

9. Заинтересованные стороны: ТУСУР.

10. Источники финансирования и материального обеспечения: Нет

11. Ожидаемый результат (полученный товар, услуга): работоспособная IoT–сеть.

12. Руководитель проекта: Пехов О.В

13. Члены проектной группы:

Коноваленко Александр Дмитриевич 731–2 (ответственный);

Демиденко Егор Вадимович 731–2;

Ведениктов Дмитрий Максимович 722–1;

Захаров Дмитрий Валентинович 722–1;

Москвичеков Александр Сергеевич 722–1;

Таганов Максим Михайлович 722–1;

14. Место выполнения проекта: ул. Красноармейская, д. 146, 7 этаж, ауд. 707.

15. Календарный план выполнения проекта:

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
1	Настройка виртуального сервера для быстрого развертывания	Написание автономных скриптов для автозапуска и редактирования настроек компонентов системы	06.10.2024	16.11.2024	Настроенный виртуальный сервер для быстрого развертывания.

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
2	Корректировка прошивки устройства	Доработка функционала прошивки, обеспечение автономности, оптимизация работы кода	06.10.2024	16.11.2024	Улучшенный функционал конечного устройства.
3	Разработка платы расширения для Nucleo F103RB	Разработка схемы платы расширения, разводка платы расширения в системе автоматизированного проектирования	08.09.2024	15.10.2024	Разработанная плата расширения для Nucleo F103RB
4	Доработка Web-приложения	Переработка интерфейса на уровне кода, реализация ролевого разграничения доступа на сайте, реализация масштабируемости интерфейса, добавление возможности фильтрации и выбора нескольких устройств, расширение функционала таблиц с данными, решение быстрого закрытия сессии и улучшение	06.10.2024	13.12.2024	Доработанное Web-приложение с переработанным интерфейсом, доработанными графиками данных и улучшенными таблицами

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
		общей визуальной части			
5	Разработка корпуса для конечного устройства	Создание чертежа корпуса устройства, моделирование и печать корпуса для конечного устройства	06.10.2024	16.11.2024	Готовый корпус для конечного устройства и его чертежи.
6	Отчётность	Написание отчета по ГПО, составление документации виртуального сервера	11.12.2024	18.12.2024	Отчёт по проделанной работе. Составленная документация

«19» декабря 2024 г.

Руководитель проекта:

Старший преподаватель каф. КИБЭВС

(должность)

(подпись)

Пехов О.В.

(расшифровка)

Члены проектной группы:

(подпись)

Коноваленко А.Д

(расшифровка)

(подпись)

Демиденко Е.В.

(расшифровка)

(подпись)

Ведениктов Д.М.

(расшифровка)

(подпись)

Захаров Д.В.

(расшифровка)

(подпись)

Москвичиков А.С.

(расшифровка)

(подпись)

Таганов М.М.

(расшифровка)

Реферат

Отчет содержит 86 страниц, 41 рисунок, 13 источников, 8 приложений.

LORA, LORAWAN, БАЗОВАЯ СТАНЦИЯ, МИКРОКОНТРОЛЛЕР, РАДИОМОДУЛЬ, СБОР ПАРАМЕТРОВ, АВТОРИЗАЦИЯ, СОЕДИНЕНИЕ С СЕРВЕРОМ, КЛИЕНТСКОЕ ПРИЛОЖЕНИЕ, АТАКИ НА LORA–СЕТЬ.

Объект исследования: Системы интернет вещей

Цели работы:

- изучение технологий модуляции LoRa и сетевого протокола LoRaWAN;
- изучение документации оборудования, которое будет использовано для реализации IoT–сети;
- произвести настройку данного оборудования;
- создание работоспособной IoT–сети.

Пояснительная записка к групповой проектной работе выполнена в текстовом редакторе Microsoft Word 2016.

Оформлено в соответствии с ОС ТУСУР 01 – 2021. [1]

Оглавление

Введение	9
1 ПРАКТИЧЕСКАЯ ЧАСТЬ	10
1.1 Алгоритм добавления устройств на сервер через IoT Vega Server	10
1.2 Настройка автономной работы виртуального сервера	11
1.3 Составление документации описывающей процесс работы и настройки виртуального сервера	14
1.4 Реализация проектов на языке программирования Python	14
1.5 Переработка интерфейса на уровне кода	15
1.6 Реализация масштабируемости интерфейса	16
1.7 Реализация ролевого разграничения доступа на сайте	18
1.8 Добавление возможности фильтрации данных	22
1.9 Множественные выбор устройств для отображения на графике	24
1.10 Расширение функционала таблицы с данными	25
1.11 Решение проблемы быстрого закрытия сессии пользователя	27
1.12 Улучшение общей визуальной части веб–приложения	28
1.13 Разработка платы расширения для Nucleo F103RB	29
1.14 Сравнение контроллеров Nucleo F103RB и BluePill	31
1.15 Схема устройства с отладочной платой BluePill	32
1.16 Изменения в прошивке устройства	34
1.17 Адаптация прошивки под контроллер BluePill	37
1.18 Оптимизация работы кода	38
1.19 Обеспечение автономности устройства	40
1.20 Разработка корпуса устройства	41
Заключение	49
Список источников	51
Приложение А (обязательное) Скрипт ручного запуска сервера Vega Server	53
Приложении Б (обязательное) Скрипт ручного запуска веб–сервера	54
Приложение В (обязательное) Скрипт запуска файла вывода логов	55
Приложение Г (обязательное) Файл routes.py	56
Приложение Д (обязательное) Файл forms.py	68
Приложение Е (обязательное) Файл main.crr	72
Приложение Ж (обязательное) Блок схема программы устройства	86
Приложение З (обязательное) Чертеж корпуса и крышки	

Введение

Интернет вещей – это система взаимосвязанных вычислительных устройств, которые могут собирать и передавать данные по беспроводной сети без участия человека.

Рынок интернета вещей растет очень быстро. В России объем рынка IoT за последние два года увеличился на 30% и составил 172 млрд рублей. Аналитики прогнозируют, что к 2030 году рынок может вырасти на 60% и достичь 276 млрд рублей. К концу 2024 года насчитывается примерно 102,3 млн всевозможных IoT-устройств, что на 19% больше по сравнению с предыдущим годом. Это говорит о значительной динамике роста рынка IoT [2].

Сбор данных с помощью устройств IoT достиг огромных масштабов. Происходит объединение науки о данных и машинного обучения для передовых решений и анализа данных интернета вещей, Big Data и искусственного интеллекта для сбора предварительно структурированных данных. Развитие технологий IoT в современное время уверенно движется вперед. Многие современные проблемы замедляют этот процесс, но не останавливают его.

Целью проекта «Исследование и создание IoT–сети, основанной на технологиях модуляции LoRa и сетевого протокола LoRaWAN» является настройка оборудования, способного взаимодействовать между собой при помощи данных технологий, для создания IoT–сети, по которой будет происходить передача данных с конечных устройств на сервер, а также изучение принципов обеспечения безопасности передачи этих данных в сети.

1 ПРАКТИЧЕСКАЯ ЧАСТЬ

1.1 Алгоритм добавления устройств на сервер через IoT Vega Server

Так как IoT Vega Server предоставляет мощные и гибкие возможности для управления сетью IoT устройств, для добавления устройств необходим собственный алгоритм. Для устройств приема-передачи данных был разработан следующий алгоритм: необходимо было задать DevEUI - уникальный номер устройства - строка из 16-ти шестнадцатеричных символов; AppEUI - EUI идентификатор приложения устройства - строка из 16-ти шестнадцатеричных символов; AppKey - ключ приложения устройства – это строка из 32-х шестнадцатеричных символов. Были сформированы следующие требования AppEUI - остается неизменным, а AppKey - формируется из неизменного AppEUI и уникального номера устройства DevEUI. Настройки для устройства представлены на рисунке 1.1.

Nº	Frequency	Enabled
1	FIXED	<input checked="" type="checkbox"/>
2	FIXED	<input checked="" type="checkbox"/>
3	FIXED	<input checked="" type="checkbox"/>
4	867100000	<input checked="" type="checkbox"/>
5	867300000	<input checked="" type="checkbox"/>

Рисунок 1.1 – Настройки устройств на основании требований

Процесс передачи данных для двух устройств представлен на рисунке 1.2.

Date	Type	Data	DR
11.04.2024 13:11:46	UNCONF_DOWN		SF7 BW125 4/5
11.04.2024 13:11:46	CONF_UP	2699330010	SF7 BW125 4/5
11.04.2024 13:11:14	UNCONF_DOWN		SF7 BW125 4/5
11.04.2024 13:11:14	CONF_UP	2699330010	SF7 BW125 4/5
11.04.2024 13:10:42	UNCONF_DOWN		SF7 BW125 4/5
11.04.2024 13:10:42	CONF_UP	2699330010	SF7 BW125 4/5
11.04.2024 13:10:10	UNCONF_DOWN		SF7 BW125 4/5
11.04.2024 13:10:10	CONF_UP	2699330010	SF7 BW125 4/5
11.04.2024 13:09:38	UNCONF_DOWN		SF7 BW125 4/5
11.04.2024 13:09:38	CONF_UP	2699330010	SF7 BW125 4/5
11.04.2024 13:09:06	UNCONF_DOWN		SF7 BW125 4/5
11.04.2024 13:09:06	CONF_UP	2799330010	SF7 BW125 4/5
11.04.2024 13:08:34	UNCONF_DOWN		SF7 BW125 4/5
11.04.2024 13:08:34	CONF_UP	2799330010	SF7 BW125 4/5
11.04.2024 13:08:02	UNCONF_DOWN		SF7 BW125 4/5
11.04.2024 13:08:02	CONF_UP	2799330010	SF7 BW125 4/5

Рисунок 1.2 – Передача данных устройства Dev2

Согласно вышеописанному алгоритму, были добавлены три устройства с настройкой согласно требованиям, что представлено на рисунке 1.3.

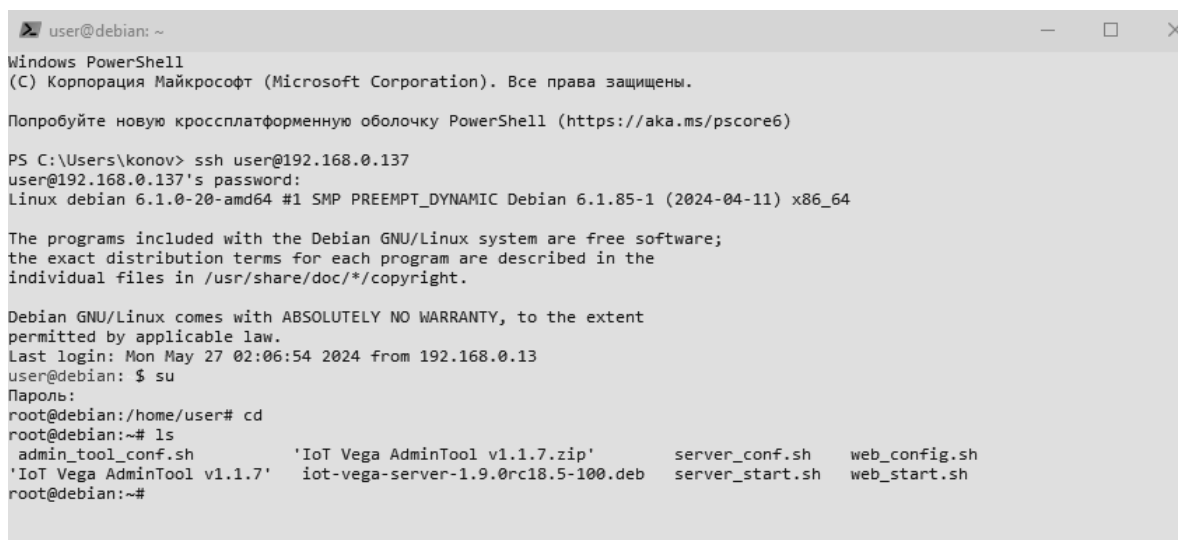
	#	Device name	DevEUI	Last connection ^	Group
<input type="checkbox"/>	1	Dev3	AC1F09FFFE015302	11.04.2024 13:11:46	
<input type="checkbox"/>	2	Dev2	AC1F09FFFE0152F9	11.04.2024 13:17:10	
<input type="checkbox"/>	3	Dev1	AC1F09FFFE015305	11.04.2024 13:17:01	

Рисунок 1.3 – Добавленные устройства

1.2 Настройка автономной работы виртуального сервера

Ранее была настроен виртуальный сервер на операционной системе Debian12 на основе требований аппаратных ресурсов и с возможностью

подключения через SSH. Процесс подключения представлен на рисунке 1.4.



```
user@debian: ~  
Windows PowerShell  
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.  
Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)  
  
PS C:\Users\konov> ssh user@192.168.0.137  
user@192.168.0.137's password:  
Linux debian 6.1.0-20-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.85-1 (2024-04-11) x86_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon May 27 02:06:54 2024 from 192.168.0.13  
user@debian: $ su  
Пароль:  
root@debian:/home/user# cd  
root@debian:~# ls  
admin_tool_conf.sh      'IoT Vega AdminTool v1.1.7.zip'      server_conf.sh      web_config.sh  
'IoT Vega AdminTool v1.1.7'  iot-vega-server-1.9.0rc18.5-100.deb  server_start.sh      web_start.sh  
root@debian:~#
```

Рисунок 1.4 – Подключение через SSH

Для настройки виртуального сервера был предустановлен программный комплекс IoT Vega Server, Admin Tool и скрипт веб-приложения. Все необходимые для развертывания системы скрипты были инициализированы в системе в рамках автозапуска. Скрипт для автоматического запуска IoT Vega Server представлен в приложение А. Скрипт для автоматического запуска веб-приложения представлен в приложении Б. Также каждый их скриптов поддерживает проверку на запущенные ими ресурсы, при отсутствии запущенного ресурса скрипт рекурсивно инициализирует запуск. Также на виртуальном сервере имеются скрипты для ручного запуска настройки и запуска IoT Vega Server, Admin Tool и веб-приложения. Результат выполнения скриптов ручного запуска и настройки IoT Vega Server представлен на рисунке 1.5.

```

root@debian:~# IOT Vega Server 1.9.0rc18.5 [100 devices]
"WARN [27-05-2024 02:16:18.678](void CLogFile::start():257) - LOG: current file for log messages [./history_1.log]"
INFO [27-05-2024 02:16:18.677](void CFreqPlanHandler::reinitFreqPlanList():131) - Prepare to reinitialize frequency plan
files...
INFO: [05-27 02:16:18.810](void CCustomDbConnectionRoutine::startSecureScanning(DB_UNI::SDBRequestParams):6109) - INFO: D
B-secure scanner started...
INFO: [05-27 02:16:18.811](void CCustomDbConnectionRoutine::startSecureScanning(DB_UNI::SDBRequestParams):6236) - INFO: D
B-secure scanner successfully finished
DEBUG-INFO[27-05-2024 02:16:18.812](void CSecurityScannerHandler::setDeviceCountInfo(const CDevicesCountInfo&):330) - set
Counter: vega[0], totalNonVega[100], usedNonVega[0]
INFO: [05-27 02:16:18.836](void CCustomDbConnectionRoutine::getGatewayStatistics(DB_UNI::SDBRequestParams):5968) - Collec
ting gateways statistics is starting...
INFO: [05-27 02:16:18.836](CCustomDbConnectionRoutine::getGatewayStatistics(DB_UNI::SDBRequestParams)::<lambda()>:5972) -
Collecting gateways statistic is ended
INFO [27-05-2024 02:16:18.853](void CUdpServer::start():103) - UDP socket has opened. IP[192.168.0.137:8001]
"DEBUG(int CCoreApplication::attach():22) - Started controller CCoreApplication"
INFO [27-05-2024 02:16:18.853](void CWebSocketServer::threadStartedSlot():116) - WebSocketServer has opened. Port[8002]
DEBUG-INFO[27-05-2024 02:16:18.853](void CUdpServer::start():122) - UdpServer handler is started

```

Рисунок 1.5 – Выполнение скрипта запуска IoT Vega Server

Выполнение скрипта для ручной настройки AdminTool представляет возможность изменения следующих параметров:

```

const address_ws = 'ws://192.168.0.137:8002';
const demo_user = false;
const select_server = false;

//const                                stock_address_ws                                =
['ws://192.168.0.246:8002','ws://127.0.0.1:8002'];

const map_tiles_leaflet = {
    url: "https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    options: {
        attribution:                                '&copy;                                <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
    }
}

```

Запуск AdminTool представлен на рисунке 1.6. Скрипт автоматического запуска и проверки на запущенный процесс представлен в Приложении А.

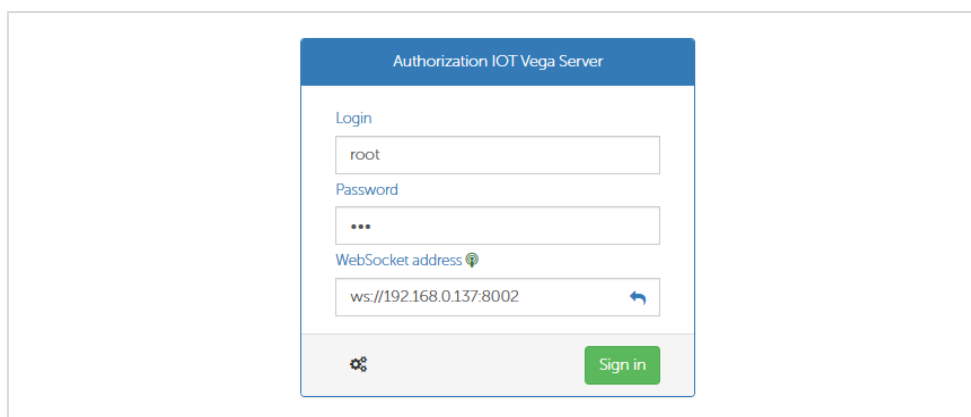


Рисунок 1.6 – Страница входа в AdminTool

1.3 Составление документации описывающей процесс работы и настройки виртуального сервера

Так как данный проект ГПО рассматривается как создание конечного продукта, то для правильного использования системы необходима составленная документация, правильно описывающая настройки и возможности. Полная версия документа настройки и описания работы виртуального сервера представлена в источнике [3].

В документации подробно описано следующее:

1. Общее описание системы;
2. Используемые файлы для ручной настройки и их описание;
3. Процесс добавления исполняемых файлов в автозапуск системы;
4. Процесс установки необходимых компонентов для работы системы;
5. Запуск и тест компонентов виртуального сервера.

1.4 Реализация проектов на языке программирования Python

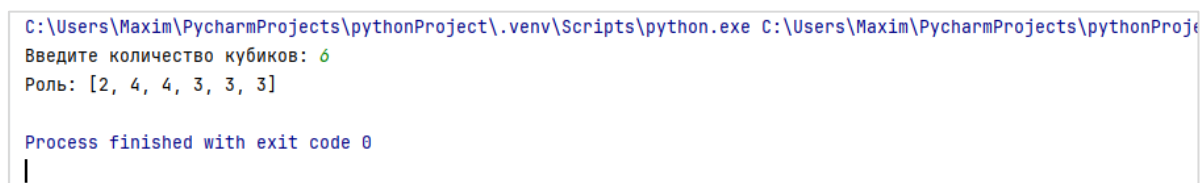
В связи с переходом на веб приложение, а также понимания структуры и работы серверной части веб–приложения было необходимо

изучить основы языка программирования Python, поскольку ранее этот язык не изучался. После изучения соответствующих курсов было реализовано несколько проектов на Python. Всего было реализовано три проекта, некоторые из которых описаны ниже (рисунки 1.7 – 1.8). [4]



```
Run asd x
C:\Users\Maxim\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\Maxim\PycharmProjects\pythonProject\.venv\asd.py
3002
10
/
300.2
Process finished with exit code 0
```

Рисунок 1.7 – Реализация проекта «Самописный калькулятор»



```
C:\Users\Maxim\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\Maxim\PycharmProjects\pythonProject\...
Введите количество кубиков: 6
Роль: [2, 4, 4, 3, 3, 3]
Process finished with exit code 0
|
```

Рисунок 1.8 – Реализация проекта «Игра в кости»

Реализация проектов, представленных выше, позволила разобраться в полученных при просмотре обучающих курсов знаниях об основах языка программирования Python и предоставила необходимые знания для возможного совершенствования и отладки серверной части веб–приложения, что позволит улучшить его функционал на поздних стадиях разработки.

1.5 Переработка интерфейса на уровне кода

В процессе разработки веб–приложения была выявлена существенная сложность, связанная с дальнейшим расширением функциональности. Основная проблема заключалась в монолитной

структура HTML–файла, то есть большая часть интерфейса находилась в одном HTML–файле, что затрудняло управление и внесение изменений. А также единая серверная функция, то есть логика обработки данных на стороне back–end также была сосредоточена в одной функции, что создавало трудности при добавлении новых функций и модулей.

Для устранения вышеуказанных проблем было принято решение провести переработку интерфейса и серверной логики. Основные шаги включали: разделение интерфейса на несколько HTML–файлов и разделение функций на back–end.

Первый шаг включал в себя вынос каждого независимого модуля в отдельный HTML–файл. Второй шаг включал в себя разбиение общей функции обработки страницы на функции, которые отвечали за решение конкретных задач.

Листинг Python–кода представлен в Приложении Г.

1.6 Реализация масштабируемости интерфейса

В процессе разработки веб–приложения возникла необходимость адаптировать интерфейс для работы в условиях расширения функциональности, роста объема данных и разности размеров мониторов и экранов разных устройств. Первоначальная архитектура интерфейса обладала рядом ограничений, которые затрудняли дальнейшее развитие системы, но вследствие переработки интерфейса на уровне кода, процесс внедрения масштабируемого интерфейса облегчился.

В первую очередь была реализовано сворачивание кнопок навигации в выпадающий список при сужении экрана. Далее было проработано расположение блоков интерфейса при сужении экрана. Пример работы на примере вкладке добавления нового пользователя представлен на рисунках 1.9 и 1.10.

Масштабирование было реализовано при помощи стилей css [5].

Интернет вещей КИБЭВС-1904

Подключенные устройства

Список пользователей

Данные с устройств

Создать учетную запись

Добавить устройство OTAA

Добавить устройство ABP

Выход

Логин:

Пароль:

Доступ к устройству: SELECTED

Работа с консолью: ☐

Список устройств: AC1F09FFFE015305
AC1F09FFFE0152F9
AC1F09FFFE015302

Список ролей: Администратор

Unsolicited: ☐

Направление: UPLINK

С MAC командами: ☐

Внести

Рисунок 1.9 – Интерфейс при широком экране

Интернет вещей КИБЭВС-1904

Логин:

Пароль:

Доступ к устройству: SELECTED

Работа с консолью: ☐

Список устройств:

- AC1F09FFFE015305
- AC1F09FFFE0152F9
- AC1F09FFFE015302

Список ролей: Администратор

Unsolicited: ☐

Направление: UPLINK

С MAC командами: ☐

Внести

Рисунок 1.10 – Интерфейс при сужении экрана

1.7 Реализация ролевого разграничения доступа на сайте

В ходе разработки сайта возникла необходимость организовать четкую систему контроля доступа пользователей к различным функциональным модулям и ресурсам. Ранее система предусматривала разграничение доступа через выдачу прав на команды, при выдаче которых администратор, создающий пользователя, мог ошибиться, что создавало следующую проблему: управление доступом выполнялось вручную, что увеличивало вероятность ошибок и снижало удобство работы администраторов.

Для решения этой проблемы было принято решение внедрить ролевое разграничение доступа на сайте. Было выделено 3 роли, а именно: администратор, оператор системы и гость, на рисунке 1.11 представлен фрагмент кода по соотношению ролей и прав на команды. Далее было внесено изменение в процесс добавления новых пользователей, а именно было изменён выбор команд доступных пользователю на выбор роли, что представлено на рисунке 1.12.

```
28 roles = [  
29     {  
30         'name': 'Администратор',  
31         'command_list': all_command_list,  
32     },  
33     {  
34         'name': 'Оператор системы',  
35         'command_list': [  
36             'get_devices',  
37             'get_device_appdata',  
38             'get_data',  
39             'server_info'  
40         ]  
41     },  
42     {  
43         'name': 'Гость',  
44         'command_list': [  
45             'get_device_appdata',  
46             'get_data',  
47             'server_info'  
48         ]  
49     }  
50 ]
```

Рисунок 1.11 – Соотношение ролей и прав на команды

The screenshot shows a web interface titled "Интернет вещей КИБЭВС-1904". It contains the following elements:

- Логин:** A text input field.
- Пароль:** A text input field.
- Доступ к устройству:** A dropdown menu with "SELECTED" selected.
- Работа с консолью:** An unchecked checkbox.
- Список устройств:** A list box containing three MAC addresses: AC1F09FFFE015305, AC1F09FFFE0152F9, and AC1F09FFFE015302.
- Список ролей:** A dropdown menu with "Администратор" selected.
- Unsolicited:** An unchecked checkbox.
- Направление:** A dropdown menu with "UPLINK" selected.
- С MAC командами:** An unchecked checkbox.
- Внести:** A green button at the bottom.

Рисунок 1.12 – Измененный процесс добавления новых пользователей

Описание ролей:

- Администратор имеет доступ к всему функционалу, который предоставляет сайт;
- Оператор системы имеет доступ к просмотру информации о устройствах и полный доступ к просмотру данных с устройств в виде графиков (выбор устройств, выбор периода, выбор количество выводимых данных);
- Гость имеет ограниченный доступ к просмотру данных с устройств в виде графиков, а именно он может произвести выбор устройств и может видеть последние 10 записей.

Работа ролевого разграничения доступа представлен на рисунках 1.13 – 1.15.

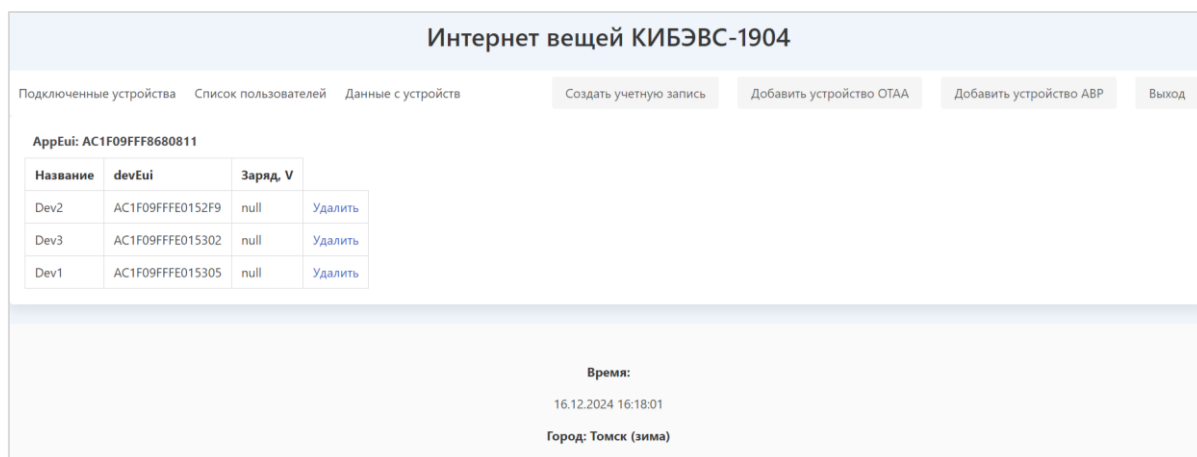


Рисунок 1.13 – Авторизация от пользователя с ролью «Администратор»

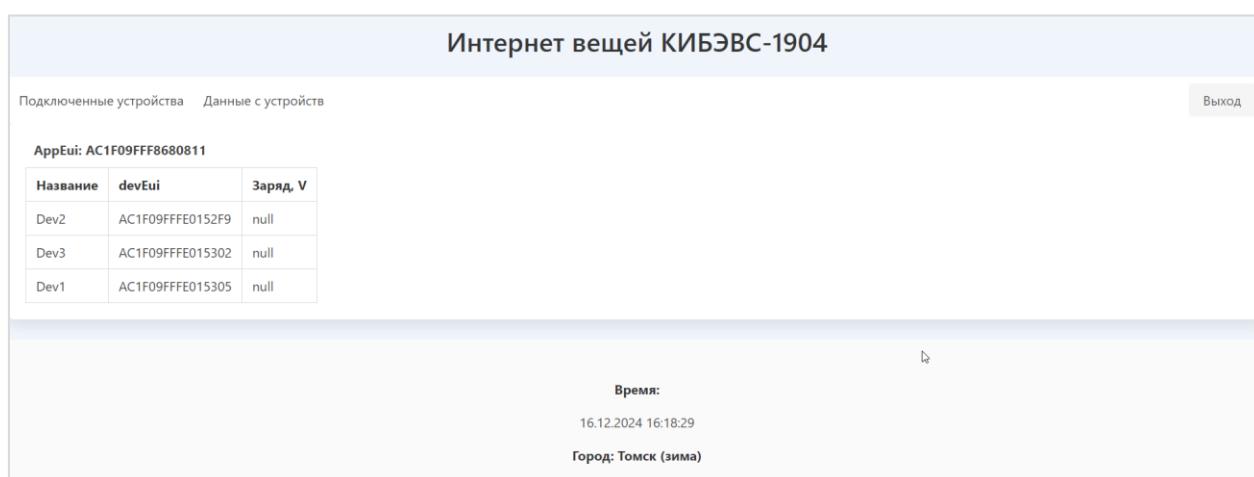


Рисунок 1.14 – Авторизация от пользователя с ролью «Оператор системы»

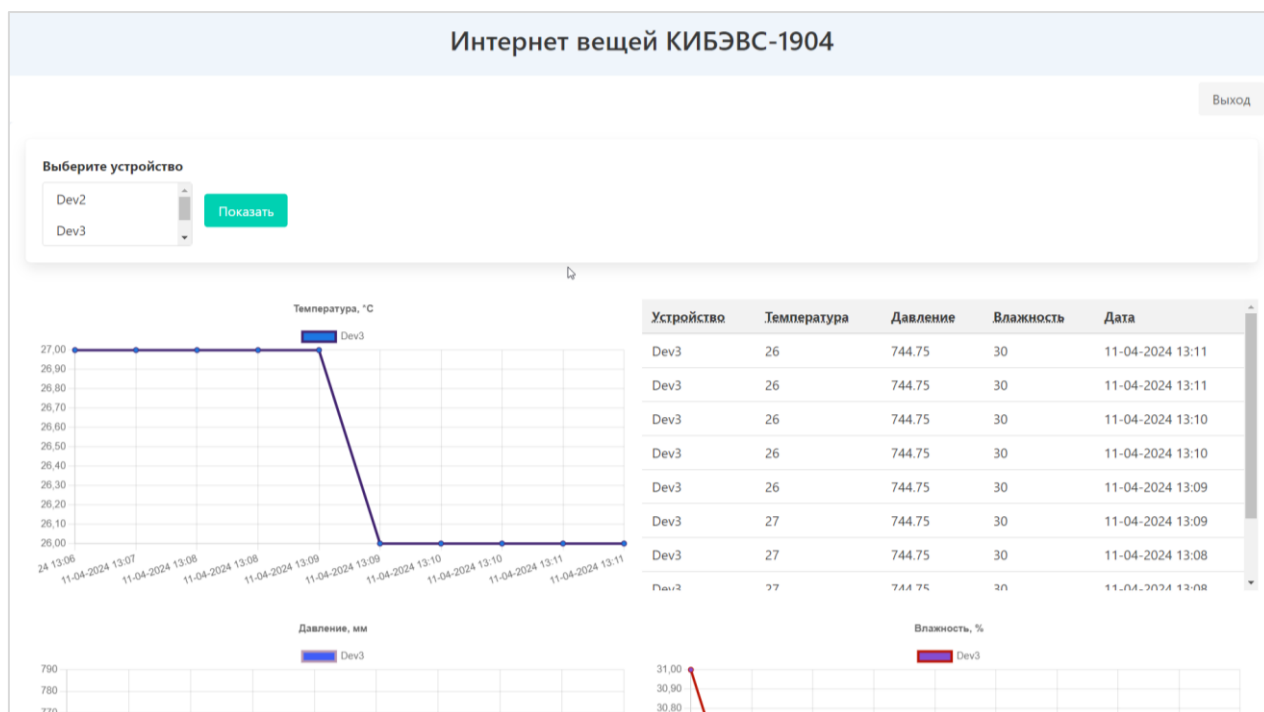


Рисунок 1.15 – Авторизация от пользователя с ролью «Гость»

1.8 Добавление возможности фильтрации данных

В рамках улучшения функциональности сайта и повышения удобства работы пользователей была реализована новая функция фильтрации данных, отображаемых на графике. Данное обновление позволяет пользователям осуществлять более точный и гибкий анализ информации, выбирая интересующий их период времени и количество отображаемых данных. Реализация включает в себя два основных фильтра: фильтр по периоду, позволяющий задать начало и конец интересующего временного интервала, и фильтр по количеству данных, позволяющий ограничить число точек, отображаемых на графике. Это позволяет оптимизировать визуализацию данных, улучшая читаемость графика, особенно при работе с большими объемами информации. В результате внедрения данной функциональности пользователи получили более

эффективный инструмент для анализа данных и построения наглядных графиков.

Фильтрация была реализована при помощи библиотеки WTForms [7]. Листинг кода на Python представлен в приложении Г и Д.

Пример выбора количества данных представлен на рисунке 1.16.

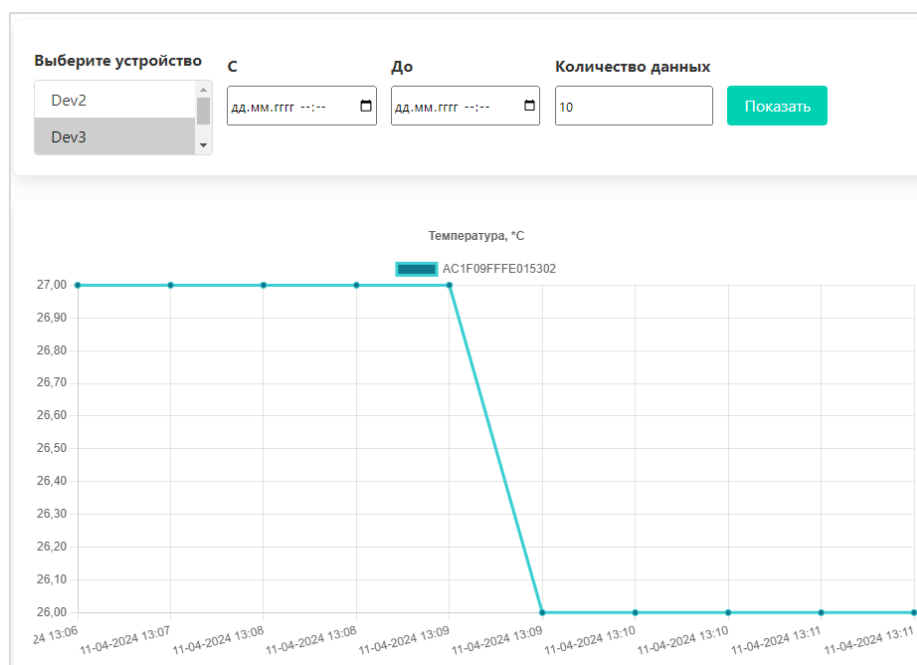


Рисунок 1.16 – Результат выбора количества данных

Пример выбора количества периода представлен на рисунке 1.17.

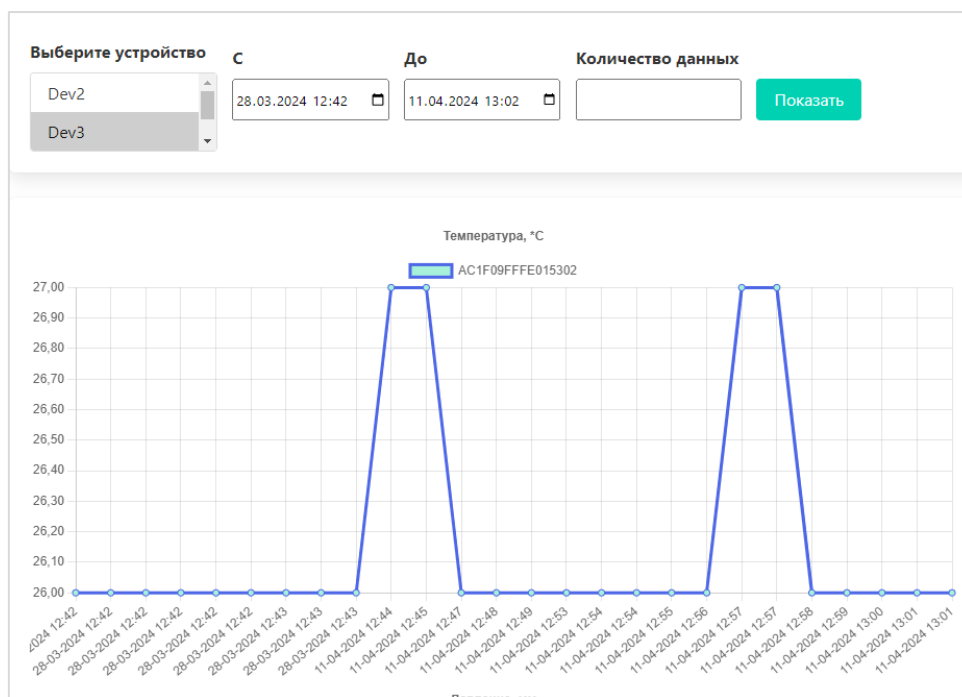


Рисунок 1.17 – Результат выбора периода данных

1.9 Множественные выбор устройств для отображения на графике

В ходе работы над сайтом потребовалось расширить функционал выбора устройств для отображения на графике. Ранее реализованный механизм позволял выбирать только одно устройство. В рамках текущей задачи была произведена модернизация, в результате которой пользователь получил возможность выбирать несколько устройств одновременно, что значительно повысило удобство работы и аналитические возможности системы. В результате модификации расширены возможности анализа данных и обеспечена гибкость в выборе отображаемой информации.

Множественный выбор устройства был реализован при помощи библиотек ChartJS[6] и WTForms [7].

Пример выбора нескольких устройств представлен на рисунке 1.18. Листинг кода на Python представлен в Приложении Д.

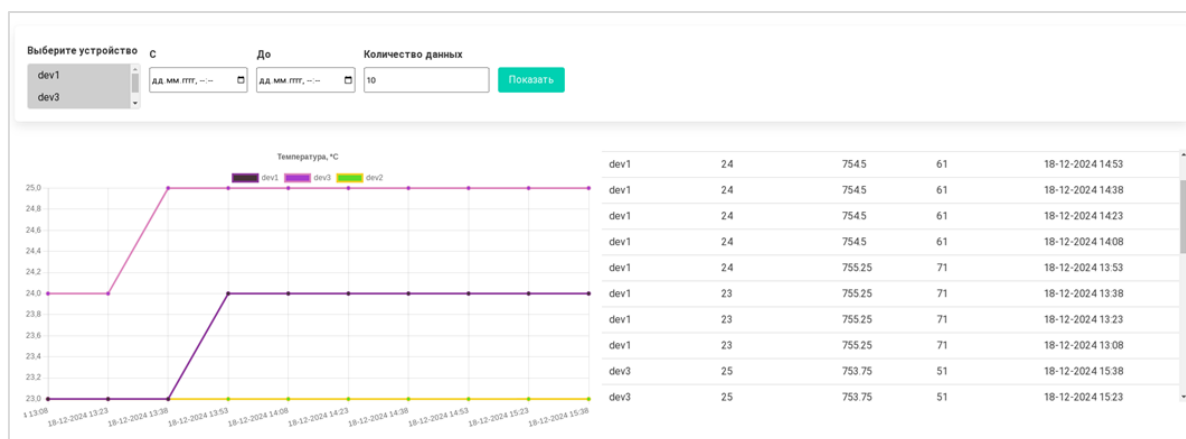


Рисунок 1.18 – Результат выбора нескольких устройств

1.10 Расширение функционала таблицы с данными

После анализа структуры кода с реализацией вывода таблиц с данными, было принято решение по двум на данный момент существующим таблицам в веб-приложении: для таблицы, содержащей данные с устройств, была добавлена возможность прокрутки, для более удобного её расположения на странице (рисунок 1.19); для таблицы, находящейся на первой странице веб-приложения, был добавлен столбец «Заряд, V», отражающий напряжение на аккумуляторе каждого подключенного устройства (рисунок 1.20). Код реализации столбца представлен в Приложении Г.

Код прокрутки таблицы:

```
<div class="box">
  <div class="table-wrapper">
    <table class="table is-fullwidth">
      <thead>
        <tr>
          {% for key in datachart.raw_data_list_keys %}
            <th><abbr title="{{ key }}">{{ key }}</abbr></th>
          {% endfor %}
        
```

```

</tr>
</thead>
<tbody>
  {% for key in datachart.raw_data_list %}
    <tr>
      {% for value in key.values() %}
        <td>{{ value }}</td>
      {% endfor %}
    </tr>
  {% endfor %}
</tbody>

```

Устройство	Температура	Давление	Влажность	Дата
Dev3	26	744.75	30	11-04-2024 13:11
Dev3	26	744.75	30	11-04-2024 13:11
Dev3	26	744.75	30	11-04-2024 13:10
Dev3	26	744.75	30	11-04-2024 13:10
Dev3	26	744.75	30	11-04-2024 13:09
Dev3	27	744.75	30	11-04-2024 13:09
Dev3	27	744.75	30	11-04-2024 13:08
Dev3	27	744.75	30	11-04-2024 13:08

Рисунок 1.19 – Новый формат таблицы с данными из устройств

AppEui: AC1F09FFF8680811			
Название	devEui	Заряд, V	
dev1	AC1F09FFFE015302	2.08	Удалить
dev3	AC1F09FFFE015306	null	Удалить
dev2	AC1F09FFFE015309	4.04	Удалить

Рисунок 1.20 – Добавленный столбец «Заряд, V» в таблицу с устройствами

1.11 Решение проблемы быстрого закрытия сессии пользователя

Вследствие изучения проблемы быстрого закрытия сессии пользователей было определено, что для поддержания работы сессии токен, поступающий в систему еще на этапе успешной авторизации, имеет жизненный цикл в одну минуту, что описано соответствующей документацией API VEGA–LoRa (рисунок 1.21). Токен автоматически обновляется, если с приложения на сервер приходит какой–либо запрос, включая простое обновление страницы [8]. За время бездействия больше одной минуты от приложения на сервер не поступает запросов, следовательно сессионный токен не обновляется и при выполнении любого действия на странице выполняется переход на страницу входа.

Так как документацией не предусмотрено какое–либо изменение времени существования сессионного токена, был реализован автоматический запрос на сервер (обновление страницы через каждый промежуток времени меньше минуты), что означает доступное обновление токена сессии.

Код автоматического запроса:

```
<script>  
    setTimeout(function() {  
        location.reload();  
    }, 45000);  
</script>
```

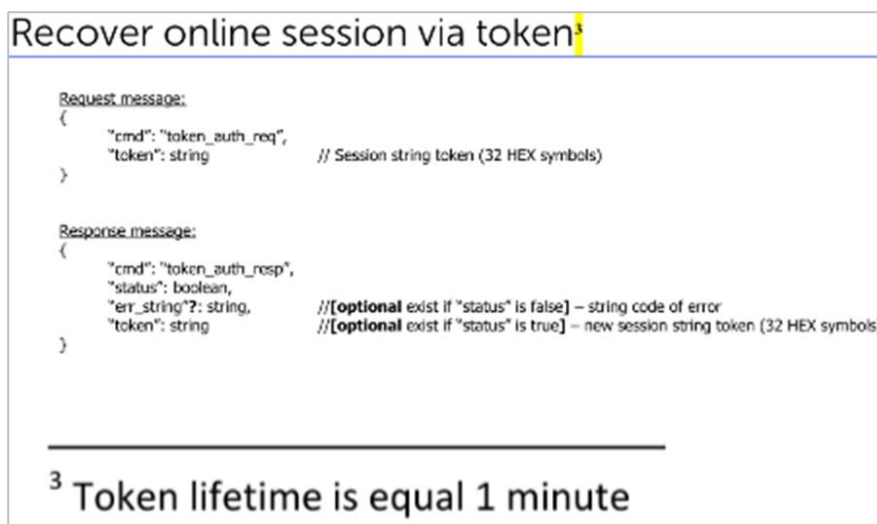


Рисунок 1.21 – Время существования токена

1.12 Улучшение общей визуальной части веб–приложения

В качестве улучшения визуальной части веб–приложения с помощью добавления стилей в html файлах, либо с помощью простого их редактирования, были убраны лишние пустые пространства между элементами, изменено расположение графиков на более оптимальное для уменьшения размеров страницы «Данные с устройств», настроены цвета шапки и фона страниц, а также отображаемые границы между элементами стали более четкими (рисунок 1.22 – 1.23).

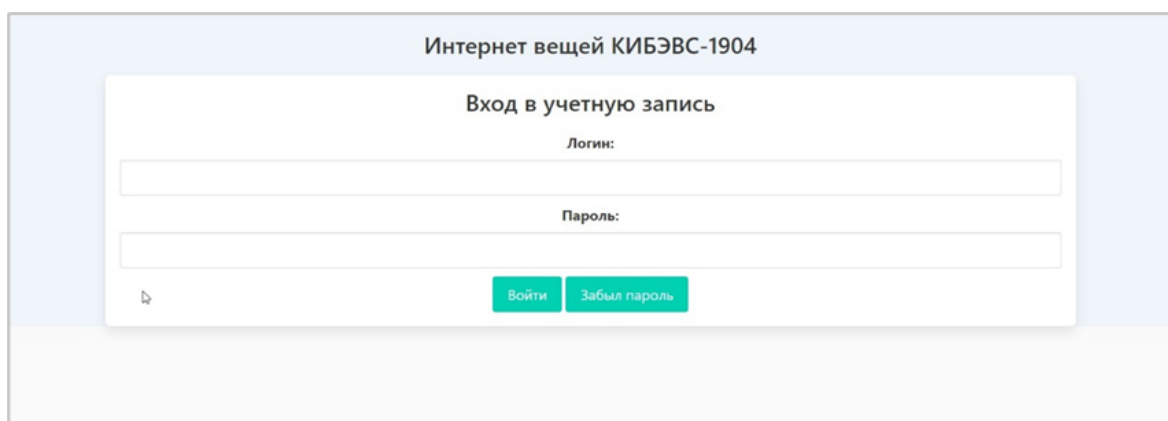


Рисунок 1.22 – Пример обновленного дизайна на странице входа

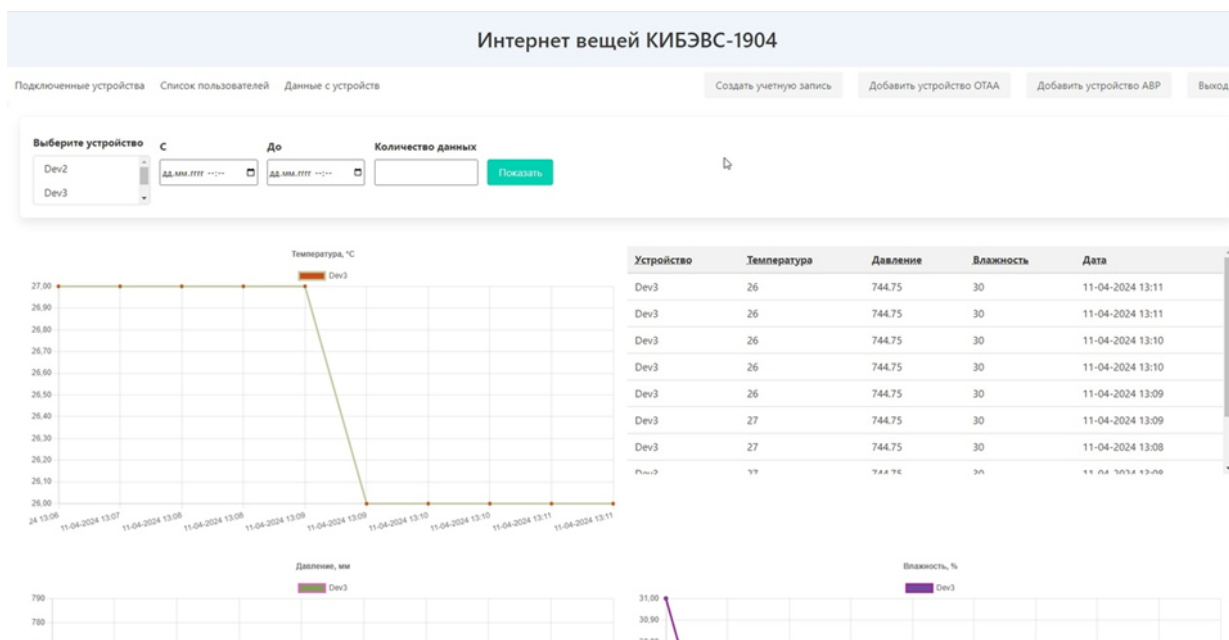


Рисунок 1.23 – Пример Обновленного дизайна на странице «Данные с устройств»

1.13 Разработка платы расширения для Nucleo F103RB

Для более компактного размещения компонентов внутри корпуса и более удобной сборки устройств возникла необходимость разработать плату расширения для отладочной платы Nucleo F103RB. Для выполнения этой задачи была использована система автоматизированного проектирования EasyEDA [9].

Плата подключается к контроллеру через посадочные контакты находящиеся на лицевой стороне платы. Схема подключения всех модулей представлена на рисунке 1.24. Модуль связи LoRaRAK использует подключение по последовательному порту. Модуль BME280 Подключен по шине I2C. Дополнительно на плате имеется индикаторный светодиод и пользовательская кнопка. Также предусмотрен делитель напряжения для измерения уровня заряда аккумулятора. Трехмерный вид платы можно увидеть на рисунке 1.25.



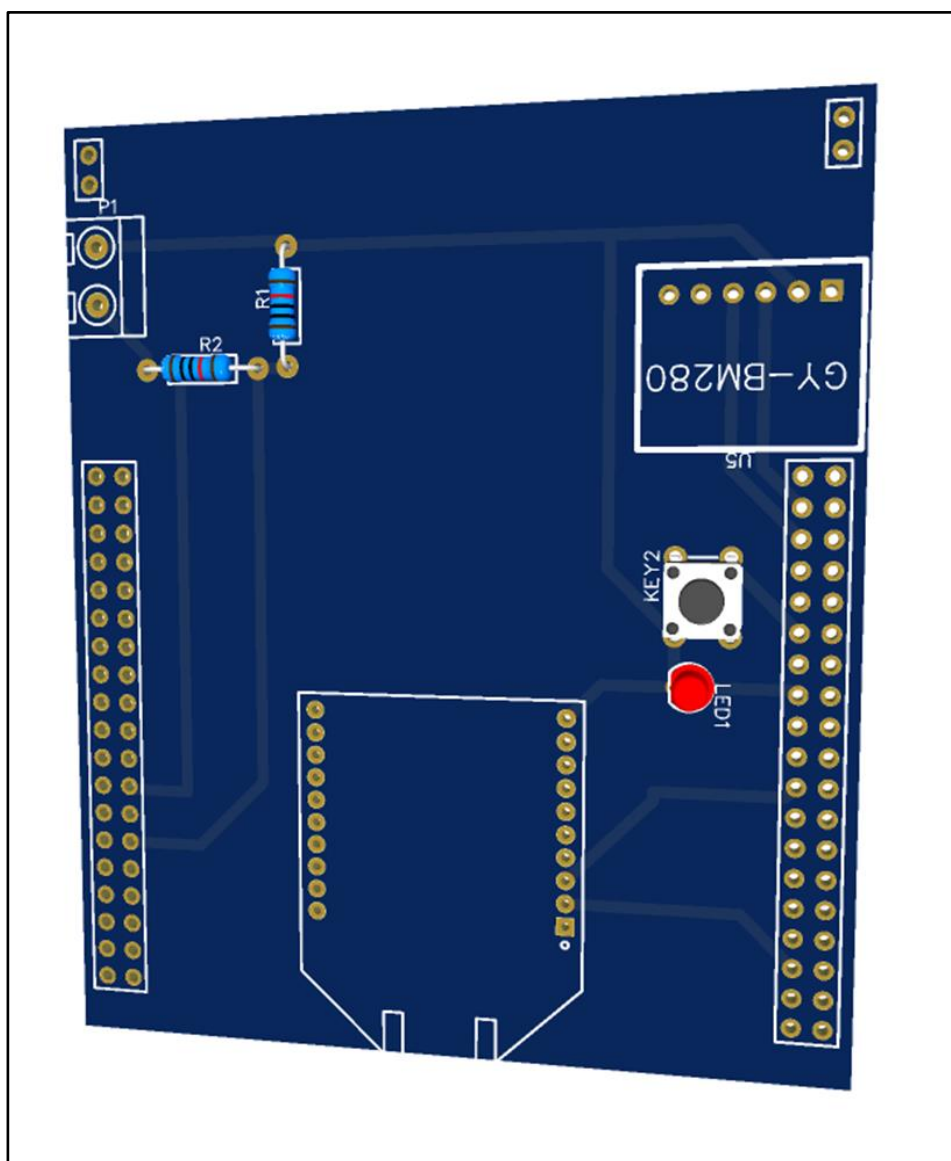


Рисунок 1.25 – Трехмерный вид платы

1.14 Сравнение контроллеров Nucleo F103RB и BluePill

В связи с невозможность изготовить плату по причине недостатка ресурсов и времени, предпринята попытка найти другое решение, которое может уменьшить размеры устройства.

Использование отладочной платы BluePill позволит уменьшить размеры устройства, так как плата имеет небольшие размеры. В таблице 1.1 Представлено сравнение характеристик двух отладочных плат. Как

можно увидеть платы идентичны по нескольким параметрам что позволяет минимизировать изменения в прошивке. Также плата BluePill обладает всеми необходимыми интерфейсами для работы с используемыми датчиками. Из минусов отсутствие встроенного программатора и менее широкий диапазон питающего напряжения и меньшее количество выводов, что не является проблемой.

Таблица 1.1 – Сравнение отладочных плат

Параметр	Blue Pill STM32F103C8T6	STM32 Nucleo F103RB
Микроконтроллер	STM32F103C8T6	STM32F103RB
Архитектура	ARM Cortex–M3	ARM Cortex–M3
Частота работы	До 72 МГц	До 72 МГц
Память Flash	128 КБ	128 КБ
ОЗУ	20 КБ	20 КБ
Питание	3.3 В (основное), USB–питание	3.3 В, USB, 5 В (через VIN или ST–Link)
Отладка и программирование	SWD (без встроенного программатора)	Встроенный ST–Link/V2–1
Кол–во выводов GPIO	37	51
Размеры платы	53 мм × 22 мм	68.6 мм × 25.3 мм
Интерфейсы	USART, SPI, I2C, CAN, USB	USART, SPI, I2C, CAN, USB

1.15 Схема устройства с отладочной платой BluePill

Разработана схема подключения основных узлов для новой отладочной платы BluePill. В новой схеме сохранены все основные

компоненты устройства изменениям подверглись только контакты, к которым подключаются устройства, использование универсальных интерфейсов облегчило переход на новый контроллер. На рисунке 1.26 представлена обновленная схема.

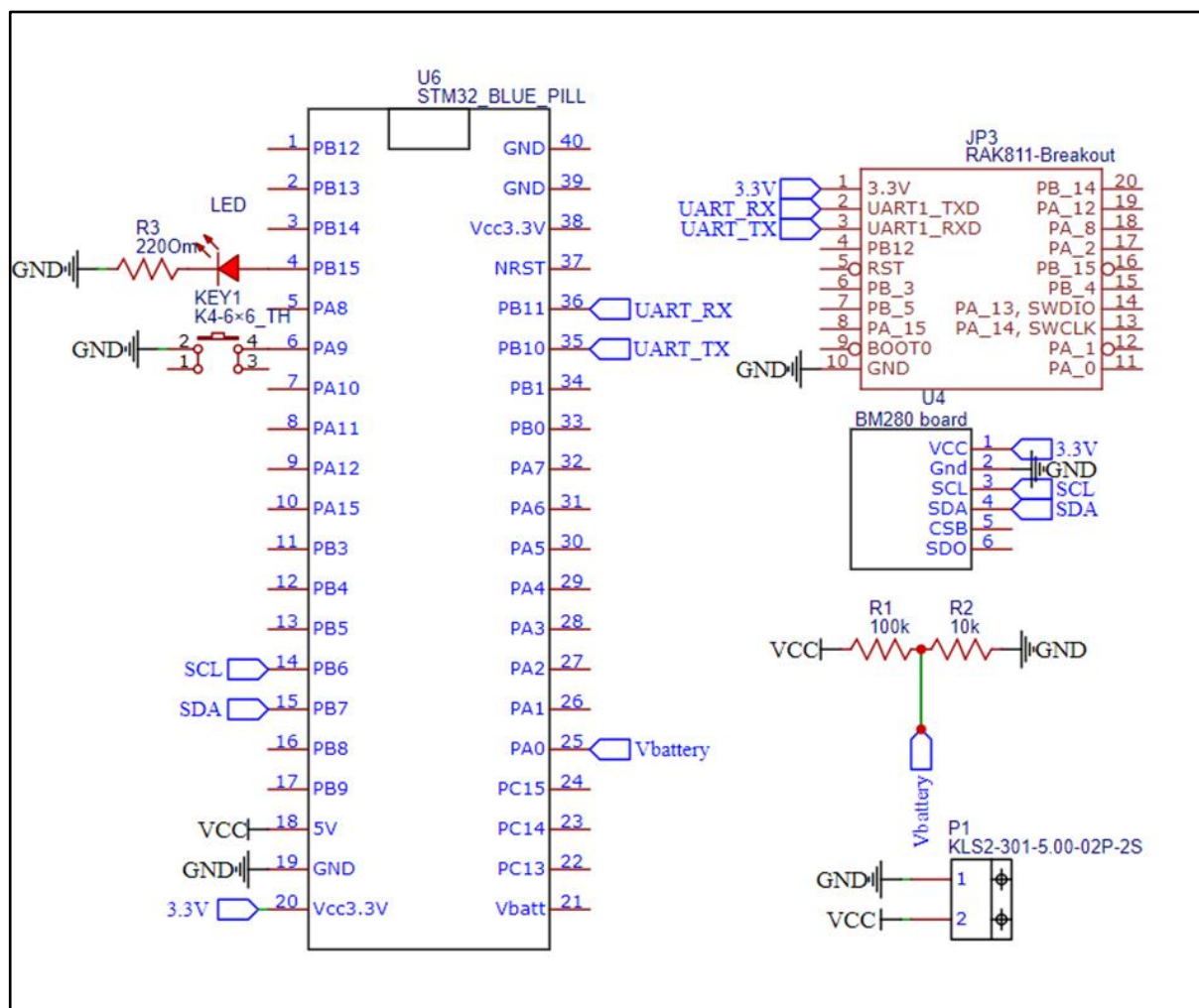


Рисунок 1.26 – Схема устройства с отладочной платой BluePill

Данная схема была собрана на макетной плате. На рисунке 1.27 представлено полностью собранная плата. В правой части располагается модуль связи LoraRAK и клеммы питания, в левой верхней части контроллер BluePill, ниже находится модуль BME280, кнопка и индикаторный светодиод.

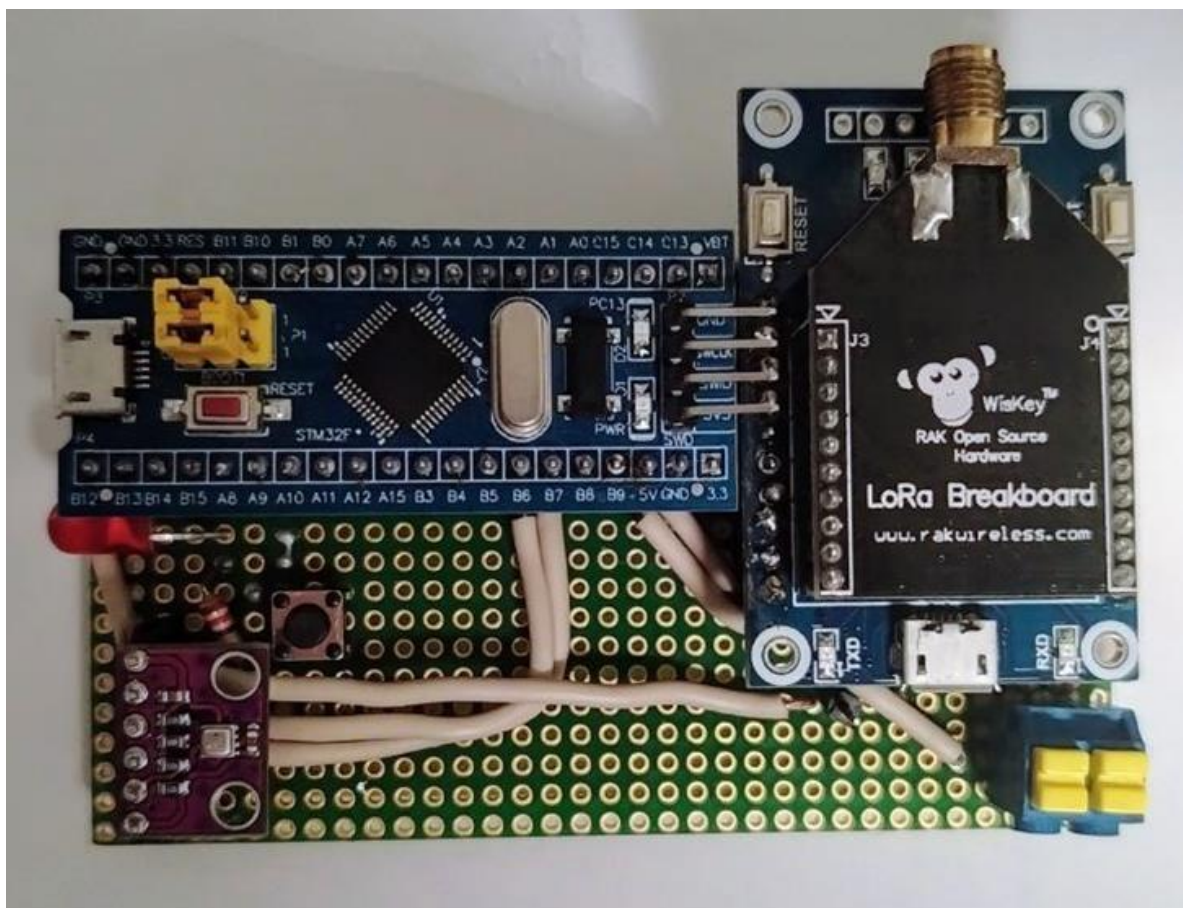


Рисунок 1.27 – Устройство собранное на плате

1.16 Изменения в прошивке устройства

Доработке подверглась прошивка устройства. Сейчас блок–схему программы можно увидеть в приложении Ж. Появилась отработка ошибок по подключению к сети, также появилась перезагрузка устройства каждые сутки. добавлен метод считывания заряда аккумулятора. Добавлена отработка кнопки, по которой запускается режим записи конфигурации, метод считывания команд из последовательного порта, метод записи и считывания конфигурации во Flash памяти. Переработан вывод логов в последовательный порт. полный листинг кода представлен в приложении Е.

Изменениям подвергся существующий класс LoraRAK используемый для связи с модулем передачи данных. На рисунке 1.28 представлена структура класса.

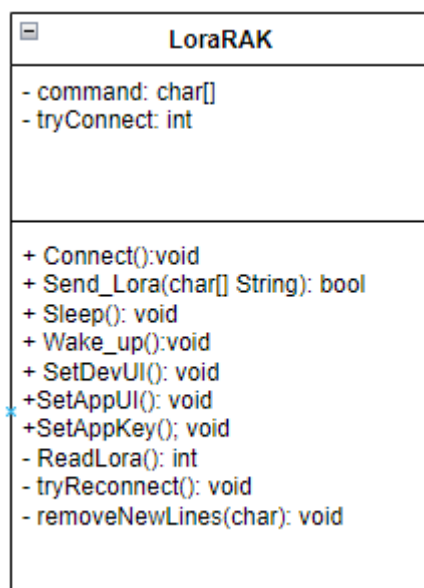


Рисунок 1.28 – Структура класса LoraRAK

Атрибут `Command` хранит в себе команду для отправки на модуль связи. Атрибут `tryConnect` хранит в себе количество нудачных подключений подряд.

Список функций и их назначение:

- `Connect` – отправляет команду на подключение к сети Lora;
- `Send_Lora` – отправляет строку для передачи по сети;
- `Sleep` – отправляет команду для перевода передатчика в режим ожидания;
- `Wake_up` – выводит передатчик из режима ожидания;
- `SetDevUI` – записывает `DevUI` на передатчик;
- `SetAppUI` – записывает `AppUI` на передатчик;
- `SetAppKey` – записывает `AppKey` на передатчик;
- `ReadLora` – считывает ответ на команды и обрабатывает ошибки возникшие у передатчика;

- tryReconnect – вызывается для обработки ошибок по подключению;
- removeNewLines – убирает лишние переходы на новую строку в строках обратной связи от передатчика.

Переработке подвергся класс обрабатывающий параметры окружающей среды. Структура класса представлена на рисунке 1.29.

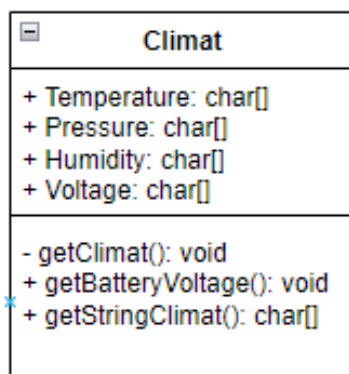


Рисунок 1.29 – Структура класса Climat

Добавлен атрибут Voltage Который хранит в себе напряжение батареи. Для записи в этот атрибут добавлен метод getBatteryVoltage, метод считывает аналоговый сигнал $U_{вх}$ с делителя напряжения и рассчитывает напряжение батареи по формуле $U_{вых} = K \cdot U_{вх}$, где K рассчитывается по формуле $K = R_2 / (R_1 + R_2)$. Сопротивления R_1 и R_2 , выбраны из имеющихся резисторов, 100 КОм и 10 КОм соответственно.

Метод getClimat вызывается когда нужно получить параметры окружающей среды с датчика BME280.

Метод getStringClimat формирует единую строку из имеющихся показателей, для дальнейшего использования.

Также реализована запись конфигурации во Flash память. Для этого создана структура, хранящая конфигурацию устройства, представлена на рисунке 1.30.

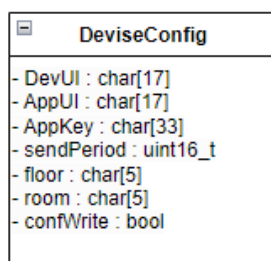


Рисунок 1.30 – Структура DeviceConfig

Далее была создана функция `enterConfigMode`, которая выводит в последовательный порт список команд и обрабатывает входящие команды, записывая настройки устройства в конфигурацию.

Написаны функции, работающие с флеш памятью. Функция `readFlash` запускается при запуске устройства и считывает конфигурацию из Flash памяти. В случае если конфигурации во Flash нет запускает режим конфигурации. Функция `writeFlash` записывает конфигурацию во Flash память по выходу из режима конфигурации.

Для реализации условного вывода отладочной информации написан макрос `LOG`. Этот макрос принимает строку форматирования и переменное число аргументов, аналогично функции `printf`. Его основная функция заключается в условном выводе отформатированного сообщения на стандартный поток вывода, что позволяет управлять отображением отладочной информации во время выполнения программы без изменения исходного кода.

Внесенные изменения расширяют функционал конечного устройства, предоставляя большие возможности по настройке устройства и более удобную эксплуатацию.

1.17 Адаптация прошивки под контроллер BluePill

В связи с переходом на новый контроллер прошивка устройства адаптирована под его особенности. Задачу переработки кода упростила

библиотека, выложенная на официальном сайте Mbed. Библиотека настраивает выводы Nucleo так чтоб их можно было использовать на BluePill. [10]

Для обеспечения совместимости прошивки с платой Nucleo–F103RB и успешной работы на микроконтроллере STM32F103C8T6, установленном на плате BluePill, библиотека реализует процедуру настройки системной тактовой частоты.

Функция конфигурации тактирования `confSysClock` обеспечивает запуск процедуры настройки системного тактирования в два этапа:

1. Сбрасывается конфигурация тактирования с помощью `HAL_RCC_DeInit()` для предотвращения возможных конфликтов с предыдущими настройками;

2. Вызывается функция `HSE_SystemClock_Config` для установки новой конфигурации системной частоты. В результате работы данной процедуры системная тактовая частота микроконтроллера STM32F103C8T6 устанавливается на 72 МГц. Шины АНВ, АРВ1 и АРВ2, а также периферийные устройства (USB и АЦП) получают стабильное и корректное тактирование, необходимое для работы прошивки.

Эта настройка позволяет плате Blue Pill работать с прошивками, изначально рассчитанными для Nucleo–F103RB, обеспечивая их полную функциональную совместимость.

Подключение и инициализация этой библиотеки позволила минимально изменить программный код, только в месте, где устанавливаются пины для работы с периферийными устройствами.

1.18 Оптимизация работы кода

В процессе работы над проектом был выявлена неправильная отправка данных. Значения давления, снимаемые с датчика, свыше 1000

отправлялись неправильно, приходили значения 000. Был проведен анализ прошивки устройства и выявлены ошибки.

Внесения поправок в код решило данную проблему и данные теперь отправляются корректно при любых значениях, снимаемых с датчика.

Была произведена отладка значений, снимаемых с датчика.

```
void getClimat()
{
    int temperature = bme.getTemperature();
    int pressure = bme.getPressure();
    int humidity = bme.getHumidity();
    int voltage = static_cast<int>(getBatteryVoltage() * 100);
    LOG("Temp: %d, Pressure: %d, Humidity: %d, Voltage: %d\n",
temperature, pressure, humidity, voltage);
```

И исправлена отправка значений, как показано ниже.

```
const char *getStringClimat()
{
    getClimat();
    static char mergedArray[4 * MAX_DIGITS + 3];
    snprintf(mergedArray,    sizeof(mergedArray),    "%s%s%s%s",
Temperature, Pressure, Humidity, Voltage);
    size_t len = strlen(mergedArray);
    mergedArray[len] = (len % 2 == 0) ? '0' : '1';
    mergedArray[len + 1] = '\0';
    return mergedArray;
}
```

Ознакомиться с кодом прошивки устройства можно в приложении Е.

1.19 Обеспечение автономности устройства

Для обеспечения автономности устройства в коде прошивки был выполнен метод для замера напряжения батареи, что позволит без проблем контролировать заряд аккумуляторов.

Создан метод для замера напряжения батареи и произведена отладка для напряжения батареи.

private:

```
char Temperature[MAX_DIGITS + 1];
char Pressure[MAX_DIGITS + 1];
char Humidity[MAX_DIGITS + 1];
char Voltage[MAX_DIGITS + 1];
float R1 = 100.0;
float R2 = 10.0;
float K = R2 / (R1 + R2);
float readV = 0;
float batteryVoltage = 0;
void getClimat()
{
    int temperature = bme.getTemperature();
    int pressure = bme.getPressure();
    int humidity = bme.getHumidity();
    int voltage = static_cast<int>(getBatteryVoltage() * 100);
    LOG("Temp: %d, Pressure: %d, Humidity: %d, Voltage: %d\n",
temperature, pressure, humidity, voltage);
    snprintf(Temperature, sizeof(Temperature), "%d", temperature);
    snprintf(Pressure, sizeof(Pressure), "%04d", pressure);
    snprintf(Humidity, sizeof(Humidity), "%d", humidity);
    snprintf(Voltage, sizeof(Voltage), "%d", voltage);
```



```

    }
public:
    float getBatteryVoltage()
    {
        readV = ain.read_voltage();
        batteryVoltage = readV / K;
        return batteryVoltage;
    }
    const char *getStringClimat()
    {
        getClimat();
        static char mergedArray[4 * MAX_DIGITS + 3];
        snprintf(mergedArray,    sizeof(mergedArray),    "%s%s%s%s",
Temperature, Pressure, Humidity, Voltage);
        size_t len = strlen(mergedArray);
        mergedArray[len] = (len % 2 == 0) ? '0' : '1';
        mergedArray[len + 1] = '\0';
        return mergedArray;
    }

```

Ознакомиться с кодом прошивки устройства можно в приложении Е.

1.20 Разработка корпуса устройства

При обновлении конфигурации устройства потребовалось создать новый корпус для 3D–печати. Для разработки был выбран CAD Solidworks, так как имеется опыт работы с данным CAD. В процессе работы периодически необходимо было обращаться к книге Д. Зиновьева

“Основы Моделирования в Solidworks” и видео–урокам “Solidworks для самых начинающих”. [11][12][13]

При разработке корпуса для устройства были учтены все недостатки предыдущего корпуса. В предыдущем корпусе процесс вскрытия корпуса для работы был весьма затруднительным, так как 4 болта были не весьма обоснованными, отверстие для антенны было немного тесноватым, так как антенна вкручивалась плотно и необходим был запас, а также было достаточное количество свободного пространства. Также, в случае с обновлением конфигурации аппаратной части, корпус должен уменьшиться минимум в 1,5 раза, что даст выигрыш в габаритах, по сравнению со старым корпусом.

Предыдущий корпус, нуждающийся в переделке, показан на рисунках 1.30–1.33.



Рисунок 1.30 – Предыдущий корпус, часть 1



Рисунок 1.31 – Предыдущий корпус, часть 2



Рисунок 1.32 – Предыдущий корпус, часть 3

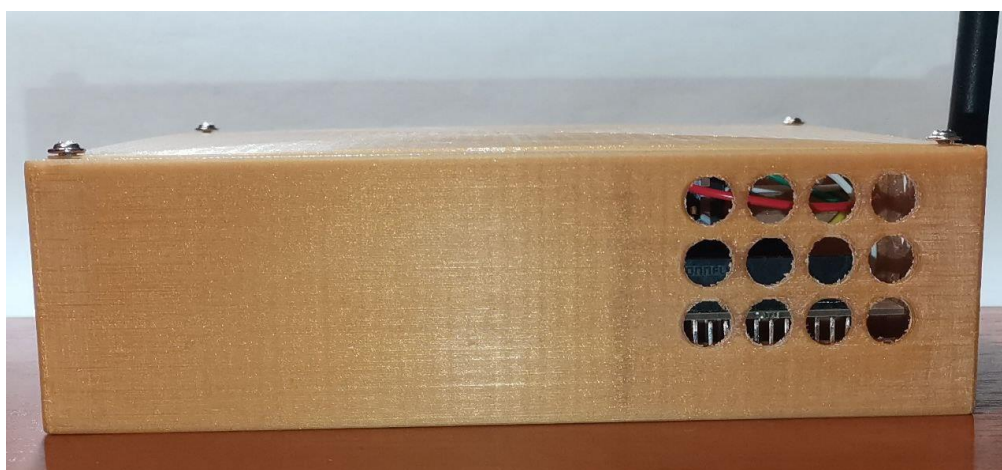


Рисунок 1.33 – Предыдущий корпус, часть 4

На рисунке 1.34 можно ознакомиться со старым расположением плат в корпусе.

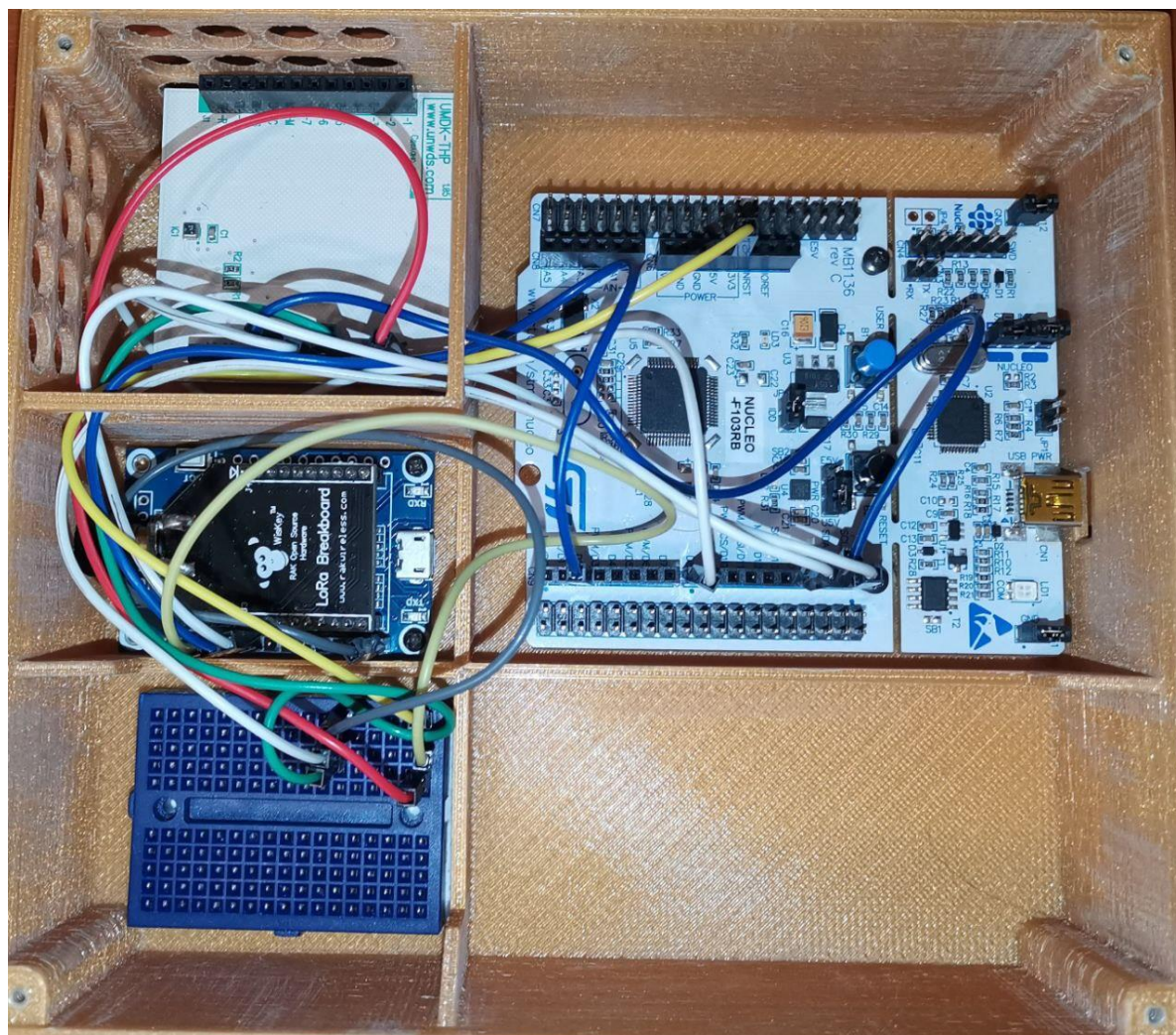


Рисунок 1.34 – Внутренняя часть предыдущего корпуса

В процессе создания корпуса с преподавателем были согласованы толщина стенок корпуса, габариты, места креплений плат.

Было принято решение сделать крышку на салазках, что позволит проводить более лёгкий демонтаж устройства. Было выполнено отверстие для диода (индикации работы устройства). Выполнены вентиляционные устройства, а также выход для антенны. Габариты корпуса: 120,5*100*39мм. Прямоугольный сквозной вырез на боковой части корпуса, как показано на рисунке 1.35, сделан для монтажа выключателя.

Внутренность корпуса разделена на 2 секции: отсек основной платы и батарейный отсек.

Ознакомиться с корпусом можно на рисунках 1.35–1.36. С чертежом корпуса и крышки можно ознакомиться в Приложении 3.

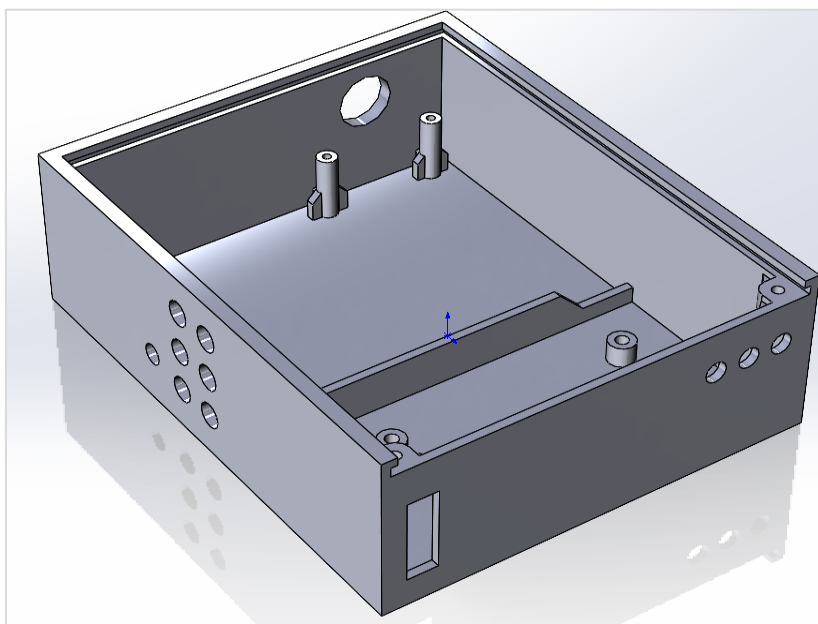


Рисунок 1.35 – Общий вид нового корпуса, часть 1

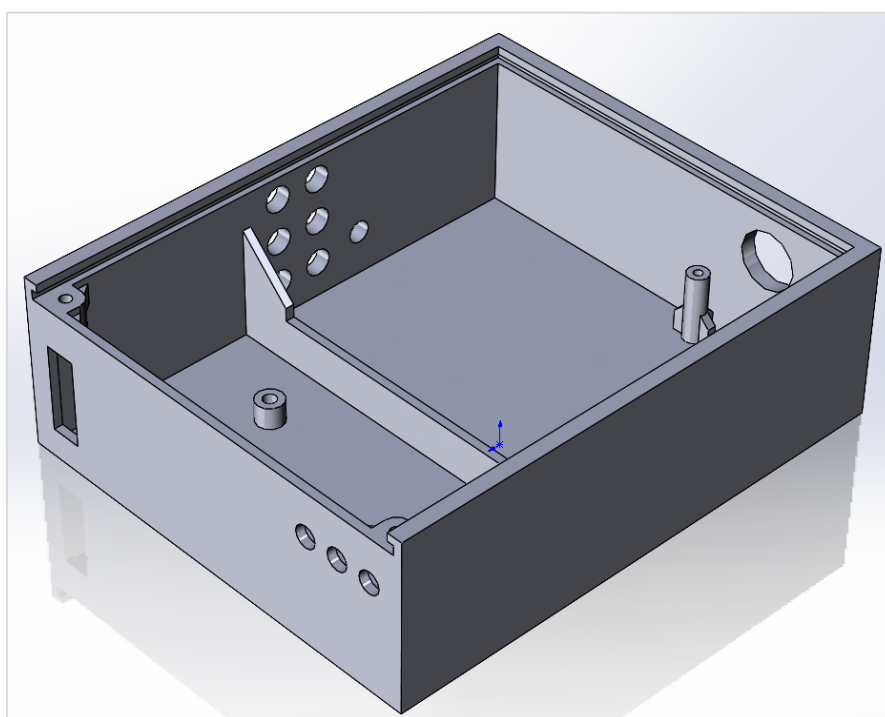


Рисунок 1.36 – Общий вид нового корпуса, часть 2

Корпус получился весьма удачным, все отверстия и крепления были сделаны и выверены точно. Фото распечатанного корпуса представлены на рисунках 1.37–1.41.

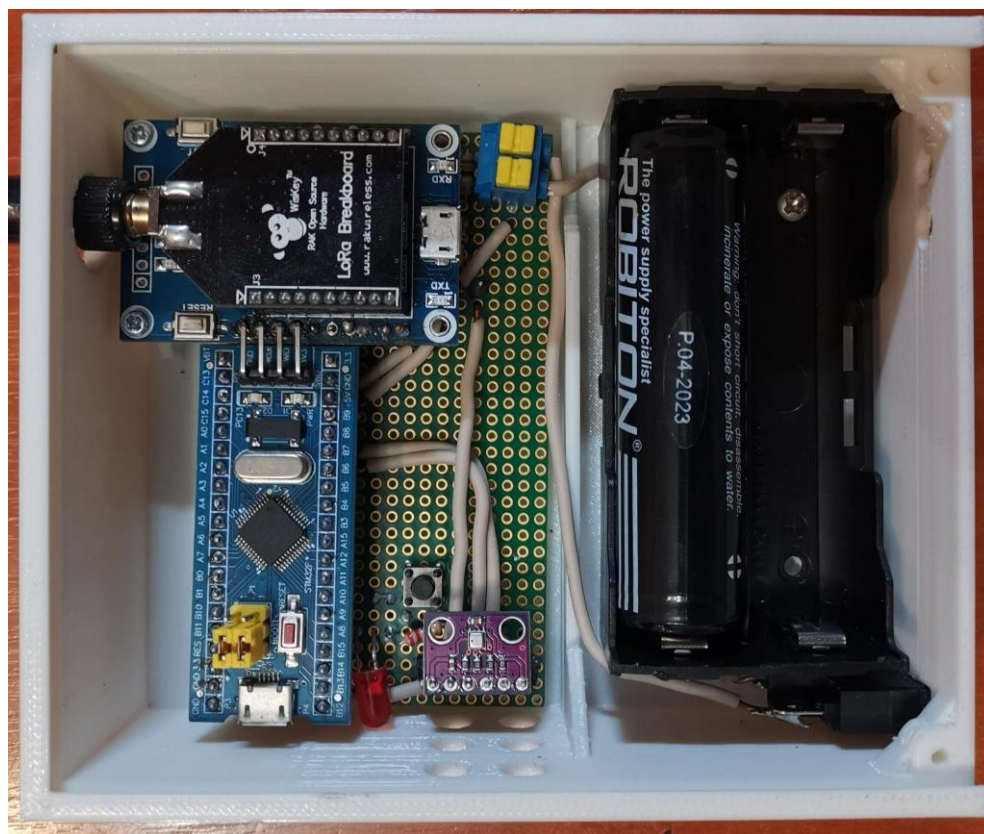


Рисунок 1.37 – Распечатанный корпус, часть 1

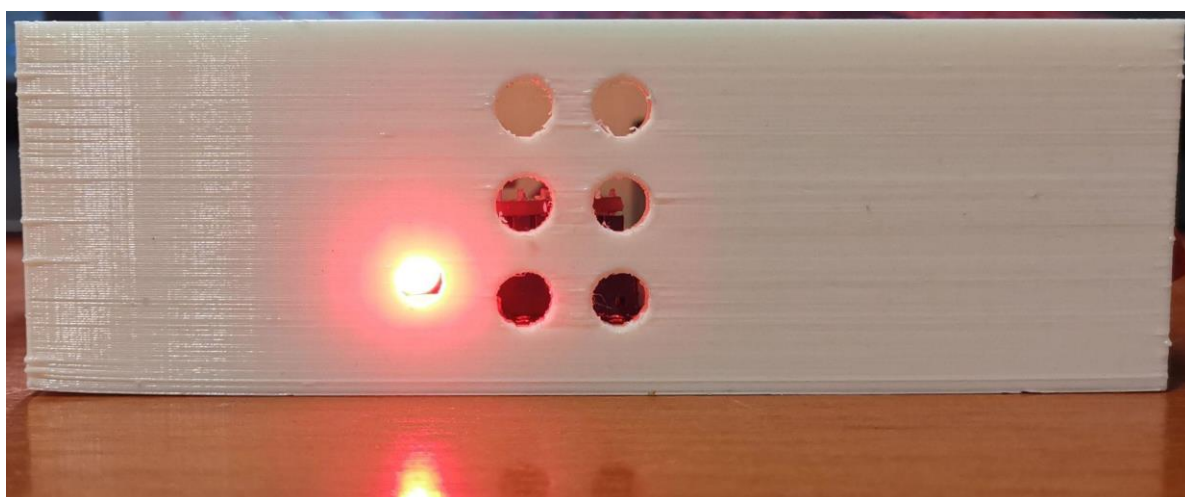


Рисунок 1.38 – Распечатанный корпус, часть 2

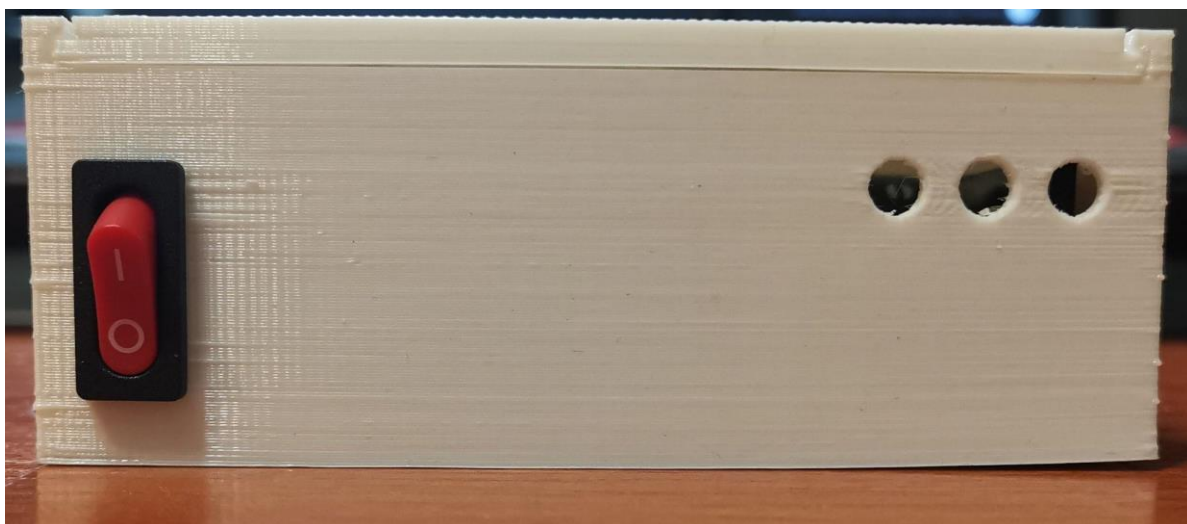


Рисунок 1.39 – Распечатанный корпус, часть 3

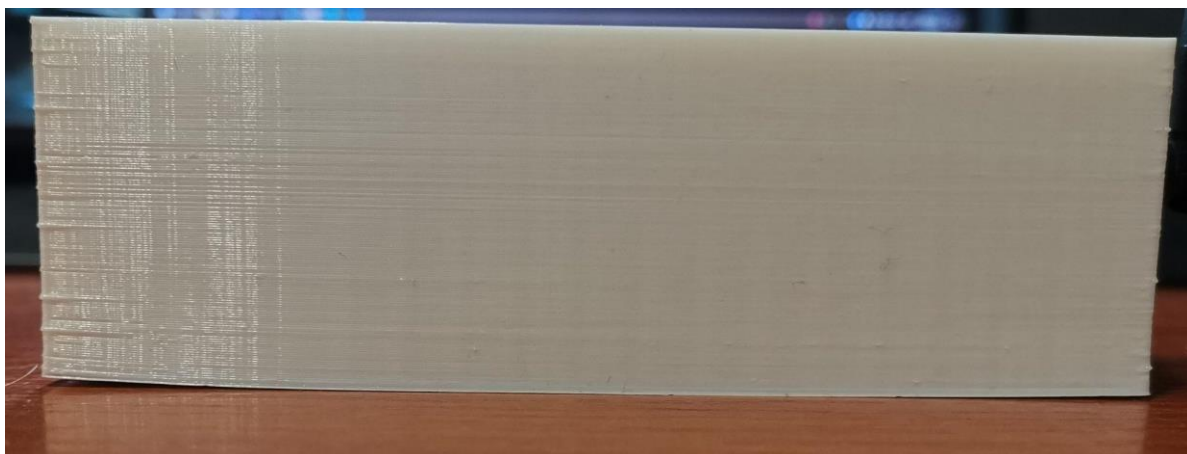


Рисунок 1.40 – Распечатанный корпус, часть 4

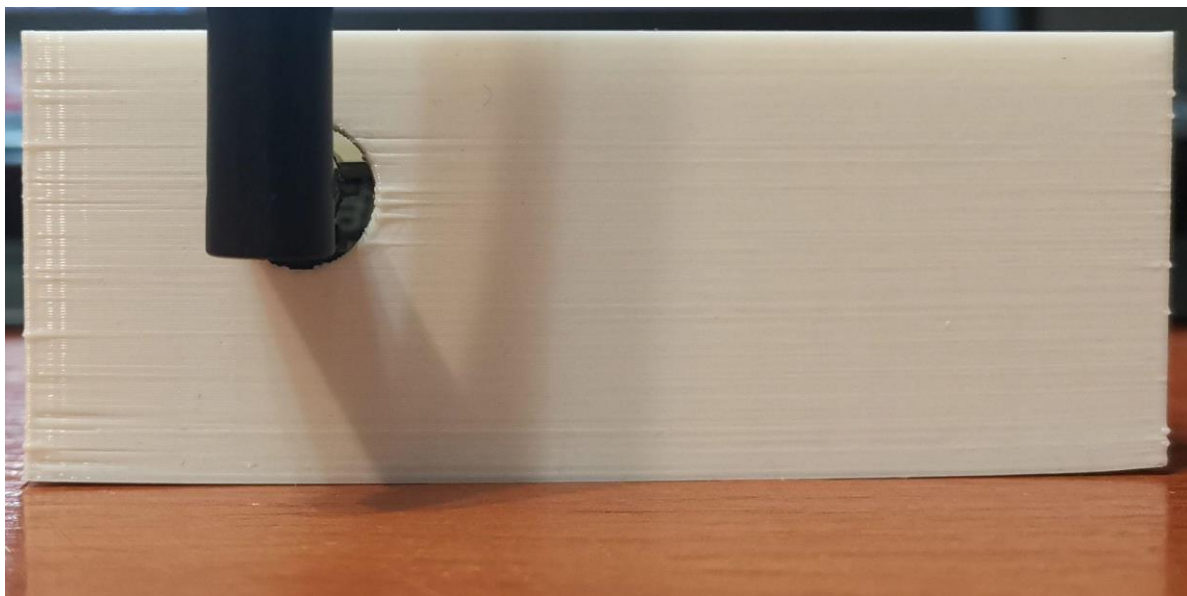


Рисунок 1.41 – Распечатанный корпус, часть 5

Заключение

В ходе выполнения проекта «Исследование и создание IoT–сети, основанной на технологиях модуляции LoRa и сетевого протокола LoRaWAN»:

- Выполнена настройка автономной работы виртуального сервера;
- Разработан алгоритм добавления устройств на сервер через IoT Vega Server;
- Составлена документация по настройке системы виртуального сервера;
- Переработан интерфейс на уровне кода;
- Реализован ролевое разграничения доступа на сайте;
- Реализован масштабируемый интерфейс;
- Был изучен язык Python для дальнейшей работы с веб–приложением;
- Добавлена возможность фильтрации данных;
- Реализован выбор нескольких устройств для отображения на графике;
- Был расширен функционал таблицы с данными;
- Была решена проблема быстрого закрытия сессии пользователя;
- Была улучшена общая визуальная часть веб–приложения;
- Добавлен новый функционал устройства (добавлена настройка конфигурации устройства, запись конфигурации во флэш–память);
- Разработана плата расширения для контроллера Nucleo F103RB;

- Адаптирована прошивка под новый контроллер BluePill;
- Оптимизирована прошивка устройства, данные отправляются корректно;
- Обеспечена автономность работы устройства (сбор данных о заряде аккумулятора);
- Спроектирован корпус для конечного устройства;
- Составлены чертежи корпуса и крышки устройства;
- Выполнена 3D–печать корпуса устройства.

Список источников

1. Образовательный стандарт ВУЗа ОС ТУСУР 01–2021. [Электронный ресурс]: Сайт ТУСУР. URL: https://storage.tusur.ru/files/40668/rules_tech_01–2021.pdf (дата обращения 11.12.2024).
2. Российский рынок интернета вещей, прогнозы роста рынка интернета вещей [Электронный ресурс] Режим доступа: <https://www.ferra.ru/news/techlife/v-rosatome-sprognozirovali-rost-rynka-interneta-veshei-na-60-k-2030-godu-19-11-2024.htm> (дата обращения 11.12.2024)
3. Руководство по настройке виртуального сервера: <https://github.com/1astheaven/GPO–storage/blob/main/Отчетность/Руководство.docx>
4. Самоучитель Python [Электронный ресурс]. Режим доступа: <https://pythonworld.ru/samouchitel–python> (дата обращения 17.03.2024).
5. Основы CSS – Руководство для самых маленьких [Электронный ресурс] – Режим доступа: <https://webdesign–master.ru/blog/html–css/css–rukovodstvo.html> (дата обращения 13.11.2024).
6. ChartJS JavaScript–библиотека визуализации данных [Электронный ресурс]. Режим доступа: <https://habr.com/ru/companies/developersoft/articles/185210/> (дата обращения 14.10.2024).
7. WTForms – гибкая библиотека для проверки и рендеринга форм в веб–разработке на Python [Электронный ресурс] Режим доступа: <https://wtforms.readthedocs.io/en/3.2.x/> (Дата обращения 20.11.2024)
8. Руководство для IOT Vega Server рев23 [Электронный ресурс] Режим доступа: <https://iotvega.com/soft/server> (дата обращения 10.11.2024).

9. Редактор плат EasyEDA [Электронный ресурс] Режим доступа: <https://easyeda.com> (дата обращения 13.11.2024).

10. Библиотека для контроллера BluePill [Электронный ресурс] Режим доступа: https://os.mbed.com/users/hudakz/code/STM32F103C8T6_Hello/ (дата обращения 08.10.2024)

11. Solidworks для самых начинающих. Часть 1 [Электронный ресурс] Режим доступа <http://www.youtube.com/watch?v=m85RlY-Обус&t> (дата обращения 19.10.2024)

12. Solidworks для самых начинающих. Часть 2 [Электронный ресурс] Режим доступа: <https://www.youtube.com/watch?v=dgo-qyD0mqg> (дата обращения 14.10.2024)

13. Книга Д. Зиновьев “Основы Моделирования в Solidworks”. [Электронный ресурс] Режим доступа: <https://monster-book.com/osnovy-modelirovaniya-v-solidworks> (дата обращения 19.11.2024)

Приложение А
(обязательное)
Скрипт запуска сервера IoT Vega Server

```
#!/bin/bash

# Каталог, в котором находится скрипт
DIR="/opt/iot-vega-server"

# Имя скрипта
SCRIPT="iot-vega-server.sh"

# Переходим в каталог
cd "$DIR" || { echo "Каталог $DIR не найден!"; exit 1; }

# Запускаем скрипт
./"$SCRIPT" &

echo "Скрипт $SCRIPT был успешно запущен."
#!/bin/bash

cd /opt/iot-vega-server

./iot-vega-server.sh
```

Приложении Б

(обязательное)

Скрипт ручного запуска веб-сервера

```
#!/bin/bash

# Переходим в каталог
cd veb2/venv/Lib/site-packages || { echo "Не удалось
перейти в каталог"; exit 1; }

# Ищем запущенный процесс run.py
pid=$(pgrep -f run.py)

# Если процесс найден, убиваем его
if [ -n "$pid" ]; then
    echo "Найден процесс run.py с PID $pid, завершаем
его..."
    sudo kill -9 $pid
    echo "Процесс завершен."
else
    echo "Процесс run.py не найден."
fi

# Запускаем run.py в фоновом режиме
echo "Запускаем run.py..."
sudo nohup python3 run.py > output.log 2>&1 &
echo "Скрипт run.py запущен в фоновом режиме."
```

Приложение В

(обязательное)

Скрипт запуска файла вывода логов

```
cd veb2/venv/Lib/site-packages  
sudo nano output.log
```

Приложение Г

(обязательное)

Файл routes.py

```
from flask import render_template, request, redirect,
url_for, flash, session
import json, datetime, tzlocal
from websocket import create_connection
from vega import app
from .forms import LoginForm, CreateUserForm,
ResetPasswordEmail, ResetPassword, AddDeviceOTAAForm,
AddDeviceABPForm, DevGraphForm, roles
from .email import send_password_reset_email
from hashlib import sha256
from .topsecret import root_password

URL_WS = "ws://localhost:8002/"

success_result_add_device_list = [
    "added",
    "updated",
    "nothingToUpdate",
    "updateViaMacBuffer"
]

def send_req(req: dict) -> dict:
    ws = create_connection(url=URL_WS)
    if req.get("cmd") != "auth_req":
        recovery_session = {"cmd": "token_auth_req",
'token': session.get('token')}
        ws.send(json.dumps(recovery_session))
        resp_json = json.loads(ws.recv())
        if resp_json.get('cmd') == "console":
            ws.recv()
        if resp_json.get("status"):
            session['token'] = resp_json.get('token')
    ws.send(json.dumps(req)) # Get command
    aaa = ws.recv()
    if aaa == "":
        return send_req(req)
    resp_json = json.loads(aaa)
    if resp_json.get('cmd') == "console":
        resp_json = json.loads(ws.recv())
        if resp_json == "" or resp_json.get("cmd") ==
"console":
            return send_req(req)
    ws.close()
    return resp_json
```



```

def add_device(set_data):
    query = {
        "cmd": "manage_devices_req",
        "devices_list": [set_data]
    }
    return send_req(query)

def getDataForIndexChart(devices, datefrom, dateto,
limit):
    data_chart = dict()
    query_request = {
        "cmd": "get_data_req",
        "select": {}
    }
    if datefrom != 0:
        query_request["select"]['date_from'] = datefrom
    if dateto != 0:
        query_request["select"]['date_to'] = dateto
    if limit != 0:
        query_request["select"]['limit'] = limit

    data_list = dict()
    lables = []
    devalistresp = send_req({"cmd":
"get_device_appdata_req"}).get("devices_list")

    for device in devices:
        query_request["devEui"] = device
        data_list[device] =
send_req(query_request).get('data_list')

    for device in devices:
        data_chart[device] = {"device":
[dev.get("devName") for dev in devalistresp if
dev.get("devEui") == device][0], "climate": [], "pressure": [],
"humidity": []}

        for data_item in data_list[device]:
            data_item['ts'] =
datetime.datetime.fromtimestamp(data_item.get('ts')
1000).strftime("%d-%m-%Y %H:%M")
            lables.append(data_item.get('ts'))

            raw_data = str(data_item.get('data'))
            if raw_data[-1] == '1':
                raw_data = raw_data[:-1]
            elif raw_data[-2] == '1':
                raw_data = raw_data[:-3]
            elif raw_data[-2] == '0':

```

```

raw_data = raw_data[:-2]

data_chart[device]["climate"].append(int(raw_data[0:2]))

data_chart[device]["pressure"].append((int(raw_data[2:5])*0.7
5))

data_chart[device]["humidity"].append(int(raw_data[5:7]))
    data_chart["lables"] = lables
    raw_data_list = []
    data_chart["devices"] = dict()
    for device in devices:
        data_chart["devices"][device] = data_chart[device]
        data_chart.pop(device)

    for device in data_chart["devices"]:
        for x in
range(len(data_chart["devices"][device]["climate"])):
            element = {
                'Устройство': [dev.get("devName") for dev
in devalistresp if dev.get("devEui") == device][0],
                'Температура':
data_chart["devices"][device]["climate"][x],
                'Давление':
data_chart["devices"][device]["pressure"][x],
                'Влажность':
data_chart["devices"][device]["humidity"][x],
                'Дата': data_chart["lables"][x]
            }
            raw_data_list.append(element)
    data_chart["raw_data_list"] = raw_data_list
    data_chart["raw_data_list_keys"] =
raw_data_list[0].keys() if raw_data_list else []
    return data_chart

def navbar_footer_data(context: dict) -> dict:
    # Get information from connected server
    infresp_dict = send_req({"cmd": "server_info_req"})
    if infresp_dict.get("err_string") == "unknown_auth":
        context["error"] = True
        return context
    if "manage_users" in session.get("command_list"):
        context['is_can_create_user'] = True
    if "manage_devices" in session.get("command_list"):
        context['is_can_add_device'] = True
    if "delete_users" in session.get("command_list"):
        context['is_can_delete_user'] = True
    if "delete_devices" in session.get("command_list"):
        context['is_can_delete_device'] = True

```

```

        if "get_data" in session.get("command_list"):
            context['is_can_get_data'] = True
        if "get_devices" in session.get("command_list"):
            context['is_can_view_devices'] = True
        if 'get_users' in session.get("command_list"):
            context['is_can_view_users'] = True
        time_serv_now = infresp_dict.get("time").get("utc") /
1000
        local_timezone = tzlocal.get_localzone()
        serv_time =
datetime.datetime.fromtimestamp(time_serv_now, local_timezone)
        context['time'] = serv_time.strftime("%Y-%m-%d
%H:%M:%S")
        context['city'] =
infresp_dict.get("time").get("time_zone", 'None')
        return context

@app.route("/device_list/delete_device/<string:dev_eui>"
, methods=["GET"])
def delete_device(dev_eui):
    query = {
        "cmd": "delete_devices_req",
        "devices_list": [dev_eui]
    }
    resp = send_req(query)
    if resp.get("status") and
resp.get("device_delete_status")[0].get('status') ==
'deleted':
        flash("Device was deleted", "info")
    else:
        flash(f"Device was not deleted, because
'{resp.get('err_string')}'", "error")
        return redirect(url_for('index'))

@app.route("/user_list/delete_user/<string:user>",
methods=["GET"])
def delete_user(user):
    query = {
        "cmd": "delete_users_req",
        "user_list": [user]
    }
    resp = send_req(query)

    if resp.get("status") and resp.get("err_string") is
None:
        flash("user was deleted", "info")
    else:
        flash(f"User was not deleted, because
'{resp.get('err_string')}'", "error")

```

```

        return redirect(url_for('user_list'))

@app.route('/create_user/', methods=['post', 'get'])
def create_user():
    context = navbar_footer_data(dict())
    if "error" in context:
        return redirect(url_for("login"))
    form = CreateUserForm()
    if request.method == "POST":
        form.devEui_list.choices = form.devEui_list.data
    if form.validate_on_submit():
        command_list = list()
        for r in roles:
            if r['name'] == form.role.data:
                command_list = r['command_list']
        set_data = {
            "login": form.login.data,
            "password": form.password.data,
            "device_access": form.device_access.data,
            "consoleEnable": form.console_enable.data,
            "devEui_list": form.devEui_list.data,
            "command_list": command_list,
            "rx_settings": {
                "unsolicited": form.unsolicited.data,
                "direction": form.direction.data,
                "withMacCommands":
form.with_MAC_Commands.data
            }
        }
        query = {
            "cmd": "manage_users_req",
            "user_list": [set_data]
        }
        resp = send_req(query)
        if resp.get("err_string") is None:
            return redirect(url_for('index'))
        else:
            flash(resp.get("err_string"), 'error')
        return render_template('create_user.html',
form=form, context=context)
        query = {
            "cmd": "get_devices_req"
        }
        resp = send_req(query)
        devices_list = resp.get("devices_list")
        form.devEui_list.data = [dev.get("devName") for dev in
devices_list]
        dev_Euis = [dev.get("devEui") for dev in devices_list]
        form.devEui_list.choices = dev_Euis

```

```

        return render_template('create_user.html', form=form,
context=context)

```

```

@app.route('/add_device/OTAA', methods=['post', 'get'])
def add_device_otaa():
    context = navbar_footer_data(dict())
    if "error" in context:
        return redirect(url_for("login"))
    form = AddDeviceOTAAForm()
    if form.validate_on_submit():
        dev_eui = form.dev_eui.data
        dev_name = form.dev_name.data
        app_eui = form.app_eui.data
        app_key = form.app_key.data
        class_user = form.class_user.data
        set_data = {
            "devEui": dev_eui,
        }
        if dev_name is not None:
            set_data["devName"] = dev_name
        otaa = {
            "appKey": app_key,
        }
        if app_eui != "":
            otaa["appEui"] = app_eui
        set_data["OTAA"] = otaa

        if class_user is not None:
            set_data["class"] = class_user

        resp = add_device(set_data)

        if resp.get("err_string") is None and
resp.get('device_add_status')[0].get(
            "status") in
success_result_add_device_list:
            return redirect(url_for('index'))
        else:

flash(resp.get('device_add_status')[0].get("status"), 'error')
        return
render_template('Устройства/add_device_OTAA.html', form=form,
context=context)
        return
render_template('Устройства/add_device_OTAA.html', form=form,
context=context)

```

```

@app.route('/add_device/ABP', methods=['post', 'get'])
def add_device_abp():

```

```

context = navbar_footer_data(dict())
if "error" in context:
    return redirect(url_for("login"))
form = AddDeviceABPForm()
if form.validate_on_submit():
    dev_eui = form.dev_eui.data
    dev_name = form.dev_name.data
    dev_address = form.dev_address.data
    apps_key = form.apps_key.data
    nwks_key = form.nwks_key.data
    class_user = form.class_user.data
    set_data = {
        "devEui": dev_eui,
        "ABP": {
            "devAddress": dev_address,
            "appsKey": apps_key,
            "mwksKey": nwks_key
        }
    }
    if dev_name is not None:
        set_data["devName"] = dev_name
    if class_user is not None:
        set_data["class"] = class_user
    resp = add_device(set_data)

    if resp.get("err_string") is None and
resp.get('device_add_status')[0].get(
    "status") in
success_result_add_device_list:
        return redirect(url_for('index'))
    else:

flash(resp.get('device_add_status')[0].get("status"), 'error')
return
render_template('Устройства/add_device_ABP.html', form=form,
context=context)
return
render_template('Устройства/add_device_ABP.html', form=form,
context=context)

@app.route('/')
def index():
    if session.get('role') == 'Гость':
        return redirect('dev_graph')
    else:
        return redirect('device_list')

@app.route('/device_list/')
def device_list():

```

```

        context = navbar_footer_data(dict())
        if "error" in context:
            return redirect(url_for("login"))
        devalistresp = send_req({"cmd":
"get_device_appdata_req"})
        context["devices_list"] =
devalistresp.get("devices_list")
        for index in range(len(context["devices_list"])):
            data = send_req({"cmd": "get_data_req", "devEui":
context["devices_list"][index]["devEui"], "select": { "limit":
1 }})
            raw_data = str(data.get("data_list")[0]["data"])
            if len(data.get("data_list")) != 0 else ""
                if len(raw_data) > 0:
                    if raw_data[-1] == '1':
                        raw_data = raw_data[:-1]
                    elif raw_data[-2] == '1':
                        raw_data = raw_data[:-3]
                    elif raw_data[-2] == '0':
                        raw_data = raw_data[:-2]
                    context["devices_list"][index]["charge"] = "null"
            if len(data.get("data_list")) == 0 or len(raw_data) <= 8 else
str(raw_data[-3:-2])+str(raw_data[-2:])
            return render_template('Устройства/DevicesRoute.html',
context=context)

```

```

@app.route('/user_list/')
def user_list():
    context = navbar_footer_data(dict())
    if "error" in context:
        return redirect(url_for("login"))
    reguserresponse = send_req({"cmd": "get_users_req"})
    context["user_list"] =
reguserresponse.get("user_list")
    return render_template('UsersRoute.html',
context=context)

```

```

@app.route('/dev_graph/', methods=['post', 'get'])
def dev_graph():
    form = DevGraphForm()
    context = navbar_footer_data(dict())
    if "error" in context:
        return redirect(url_for("login"))
    context["role"] = session["role"]

    devalistresp = send_req({"cmd":
"get_device_appdata_req"})
    context["devices_list"] =
devalistresp.get("devices_list")

```

```

        form.devEui_list.choices = [dev.get("devName") for dev
in context["devices_list"]]
        datachart = getDataForIndexChart(["AC1F09FFFE015302"],
0,0, 10)

```

```

        if form.is_submitted():
            datefrom = 0 if form.dateFrom.data is None else
int(form.dateFrom.data.timestamp())*1000
            dateto = 0 if form.dateTo.data is None else
int(form.dateTo.data.timestamp())*1000
            ChLimit = 0 if form.ChLimit.data is None else
int(form.ChLimit.data)
            devices = [device.get("devEui") for device in
context["devices_list"] if device.get("devName") in
form.devEui_list.data]
            if context["role"] == "Гость":
                datefrom = 0
                dateto = 0
                ChLimit = 10
            datachart = getDataForIndexChart(devices,
datefrom, dateto, ChLimit)
            return render_template("devgraphroute.html",
context=context, datachart=datachart, form=form)

```

```

@app.route('/logout/')
def logout():
    if 'token' in session:
        close_auth = send_req({"cmd": "close_auth_req", #
Don't change!
                                "token": session.get("token")
                                })
        if close_auth.get("err_string") is None:
            flash("You have been logged out")
            session.pop('token', None)
            return redirect(url_for('login'))
    return redirect(url_for('login'))

```

```

@app.route('/login/', methods=['post', 'get'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        autreq = {
            "cmd": "auth_req",
            "login": form.login.data,
            "password": form.password.data
        }
        autresp = send_req(autreq)
        if autresp.get("err_string") is None:

```



```

        session["command_list"] =
autresp.get("command_list")
        if form.login.data == 'root':
            session['role'] = 'root'
        elif 'create_user' in
autresp.get("command_list"):
            session['role'] = 'Администратор'
        elif 'get_devices' in
autresp.get("command_list"):
            session['role'] = 'Оператор системы'
        else:
            session['role'] = 'Гость'
            session['token'] = autresp.get("token")
            return redirect(url_for('index'))
        else:
            flash("Invalid login/password", 'error')
            return render_template('login.html', form=form)

@app.route('/reset_password_email', methods=['GET',
'POST'])
def reset_password_email():
    if "user" in session:
        session.pop("user", None)
    form = ResetPasswordEmail()
    if form.validate_on_submit():
        dateform = {"login": form.login.data, "email":
form.eml.data}

        #Вход пята
        autresp = send_req({"cmd": "auth_req", "login":
"root", "password": root_password})
        session["token"] = autresp.get("token")

        #Поиск юзера в списке юзеров
        allusers = send_req({"cmd":
"get_users_req"}).get("user_list")
        for onethe in allusers:
            if onethe.get('login') == dateform['login']:
                session["user"] = onethe
                break

        if "user" in session:
            #генерация токена алгоритмом sha256 на основе
логина
            token =
sha256(session["user"].get("login").encode('utf-
8')).hexdigest()
            # Выход пята
            out = send_req({"cmd": "close_auth_req",
"token": session.get("token")})

```

```

        if out.get("err_string") is None:
            session.pop('token', None)

send_password_reset_email(dateform.get("email"), token)
    flash('Check your email for the instructions to
reset your password')
    return redirect(url_for('login'))
    else:
        flash("The user does not exist")
        return redirect(url_for('login'))
    return
render_template('Пароль/reset_password_email.html', form=form)

@app.route('/reset_password/<token>', methods=['GET',
'POST'])
def reset_password(token):
    form = ResetPassword()
    if token !=
sha256(session["user"].get("login").encode('utf-
8')).hexdigest():
        flash("token is not true")
        return redirect(url_for('login'))

    user = session.get('user')

    # Вход pyta
    autresp = send_req({"cmd": "auth_req", "login": "root",
"password": root_password})
    session["token"] = autresp.get("token")

    if form.validate_on_submit():
        session.pop('user', None)

        delus = send_req({"cmd": "delete_users_req",
"user_list": [user.get("login")]])

        if delus.get("status") and delus.get("err_string")
is None:
            resp = send_req({
                "cmd": "manage_users_req",
                "user_list": [
                    {
                        "login": user.get("login"),
                        "password": form.password.data,
                        "device_access":
user.get("device_access"),
                        "devEui_list":
user.get("devEui_list"),

```

```

        "command_list":
user.get("command_list"),
        "rx_settings":
user.get("rx_settings")
    }
    ]
    })

    # Выход пята
    out = send_req({"cmd": "close_auth_req",
"token": session.get("token")})
    if out.get("err_string") is None:
        session.pop('token', None)

    if resp.get("status") and
resp.get("err_string") is None:
        flash('Your password has been reset.')
        return redirect(url_for('login'))
    else:
        flash("Error, try again")
        return redirect(url_for('login'))
else:
    # Выход пята
    out = send_req({"cmd": "close_auth_req",
"token": session.get("token")})
    if out.get("err_string") is None:
        session.pop('token', None)

        flash("Error, try again")
        return redirect(url_for('login'))
    return render_template('Пароль/reset_password.html',
form=form)

```

Приложение Д

(обязательное)

Файл forms.py

```
import wtforms
from flask_wtf import FlaskForm
from wtforms.validators import Length, InputRequired,
NumberRange, ValidationError, DataRequired, Email, EqualTo
```

```
all_command_list = [
    "get_users",
    "manage_users",
    "delete_users",
    "get_device_appdata",
    "get_data",
    "send_data",
    "manage_device_appdata",
    "delete_device_appdata",
    "get_gateways",
    "manage_gateways",
    "delete_gateways",
    "get_devices",
    "manage_devices",
    "delete_devices",
    "get_coverage_map",
    "get_device_downlink_queue",
    "manage_device_downlink_queue",
    "server_info",
    "send_email",
    "tx"
]
```

```
roles = [
    {
        'name': 'Администратор',
        'command_list': all_command_list,
    },
    {
        'name': 'Оператор системы',
        'command_list': [
            'get_devices',
            'get_device_appdata',
            'get_data',
            'server_info'
        ]
    },
    {
        'name': 'Гость',
        'command_list': [
```

```

        'get_device_appdata',
        'get_data',
        'server_info'
    ]
}

]

class LoginForm(FlaskForm):
    login = wtforms.StringField("Логин: ",
validators=[InputRequired()])
    password = wtforms.PasswordField("Пароль: ",
validators=[InputRequired(), Length(min=1, max=100)])
    submit = wtforms.SubmitField("Войти")

class ResetPasswordEmail(FlaskForm):
    login = wtforms.StringField("Логин: ",
validators=[InputRequired()])
    eml = wtforms.StringField("Электронная почта: ",
validators=[DataRequired(), Email()])
    send = wtforms.SubmitField("Отправить ссылку")

class ResetPassword(FlaskForm):
    password = wtforms.PasswordField("Новый пароль: ",
validators=[DataRequired(),
Length(min=8)])
    passwordtwo = wtforms.PasswordField("Повторите пароль: ",
validators=[DataRequired(message='*Required'),
EqualTo('password',
message='Пароли должны совпадать!')])
    reset = wtforms.SubmitField("Сбросить")

class CreateUserForm(FlaskForm):
    login = wtforms.StringField("Логин: ",
validators=[InputRequired()])
    password = wtforms.PasswordField("Пароль: ",
validators=[DataRequired(),
Length(min=8)])
    device_access = wtforms.SelectField("Доступ к устройству: ",
choices=["FULL", "SELECTED"],
default="SELECTED")
    console_enable = wtforms.BooleanField("Работа с консолью:")
    devEui_list = wtforms.SelectMultipleField("Список устройств:")

```

```

        role = wtforms.SelectField("Список ролей:",
choices=[role['name'] for role in roles])
        unsolicited = wtforms.BooleanField("Unsolicited:")
        direction = wtforms.SelectField("Направление:",
choices=["UPLINK", "DOWNLINK", "ALL"])
        with_MAC_Commands = wtforms.BooleanField("С MAC
командами:")
        submit = wtforms.SubmitField("Внести")

def is_string_is_hex(form, field):
    for ch in field.data:
        if ((ch < '0' or ch > '9') and
            (ch < 'A' or ch > 'F')):
            raise ValidationError('Field must be in HEX')

class AddDeviceForm(FlaskForm):
    dev_eui = wtforms.StringField("EUI устройства*: ",
validators=[InputRequired(), is_string_is_hex,
                                                    Length(16,
16, message="Must be 16 length")])
    dev_name = wtforms.StringField("Имя устройства: ")
    class_user = wtforms.SelectField("Класс пользователя",
choices=['CLASS_A', 'CLASS_C'])
    submit = wtforms.SubmitField("Добавить")

class AddDeviceABPForm(AddDeviceForm):
    dev_address = wtforms.IntegerField(
        "Адрес устройства: ",
        validators=[NumberRange(min=0x00000001,
max=0xFFFFFFFF, message="0x00000001 and 0xFFFFFFFF desired")])
    apps_key = wtforms.StringField("Application session
key: ",
validators=[Length(32,
32, message="Must be 32 length"), is_string_is_hex],
default="")
    nwks_key = wtforms.StringField("Network session key: ",
validators=[Length(32,
32, message="Must be 32 length"), is_string_is_hex],
default="")

class AddDeviceOTAAForm(AddDeviceForm):
    app_eui = wtforms.StringField("Application EUI: ",
validators=[Length(16,
16, message="Must be 16 length"), is_string_is_hex],
default="")
    app_key = wtforms.StringField("Application key: ",

```

```

                                validators=[Length(32,
32, message="Must be 32 length"), is_string_is_hex],
                                default="")

class DevGraphForm(FlaskForm):
    devEui_list = wtforms.SelectMultipleField("Выберите
устройство")
    dateFrom = wtforms.DateTimeLocalField("С", format='%Y-
%m-%dT%H:%M')
    dateTo = wtforms.DateTimeLocalField("До", format='%Y-
%m-%dT%H:%M')
    ChLimit = wtforms.IntegerField("Количество данных",
validators=[NumberRange(min=1)])
    submit = wtforms.SubmitField("Показать")

```

Приложение Е

(обязательное)

Файл main.cpp

```
#include "mbed.h"
#include "stm32f103c8t6.h"
#include "PinNames.h"
#include "BME280.h"
#include "FlashIAP.h"
#include <cstring>
#include <cstdio>
FlashIAP flash;
#define MAX_DIGITS 4
#define CONFIG_MODE_HOLD_TIME 2s
DigitalOut LED(PB_15);
InterruptIn mybutton(PA_8);
AnalogIn ain(PA_0);
BufferedSerial pc(PA_2, PA_3);
BufferedSerial dev(PB_10, PB_11);
BME280 bme(PB_7, PB_6);
Thread configThread;
volatile bool enterConfigModeFlag = false;
bool startDevice = false;
// Логи управления
bool logEnabled = true;
#define LOG(fmt, ...) \
    if (logEnabled && !enterConfigModeFlag) \
    { \
        printf(fmt, ##__VA_ARGS__); \
    }

// Настройки устройства
struct DeviceConfig
{
    char DevUI[17] = "0";
    char AppUI[17] = "0";
    char AppKey[33] = "0";
    uint16_t sendPeriod = 10; // в секундах
```



```

        char floor[5] = "";
        char room[5] = "";
        bool confWrite = false;
    } config;
    uint8_t data_read[1];
    void writeFlash(const DeviceConfig &cfg);
    void readFlash(DeviceConfig &cfg);
    class LoraRAK
    {
    public:
        void restart()
        {
            dev.write("at+set_config=device:restart\r\n",
32);

            Read_Lora();
        }

        bool Connect()
        {
            LOG("Connecting...\n");
            /*
            SetDevUI();
            SetAppUI();
            SetAppKey();
            */
            dev.write("at+join\r\n", 11);
            ThisThread::sleep_for(2s);
            return Read_Lora() == 5;
        }
        void SetDevUI()
        {
            snprintf(command, sizeof(command),
"at+set_config=lora:dev_eui:%s\r\n", config.DevUI);
            dev.write(command, strlen(command));
            Read_Lora();
        }
        void SetAppUI()
        {

```

```

        snprintf(command, sizeof(command),
"at+set_config=lora:app_eui:%s\r\n", config.AppUI);
        dev.write(command, strlen(command));
        Read_Lora();
    }
    void SetAppKey()
    {
        snprintf(command, sizeof(command),
"at+set_config=lora:app_key:%s\r\n", config.AppKey);
        dev.write(command, strlen(command));
        Read_Lora();
    }

    bool Send_Lora(const char *data)
    {
        char Serialbuffer[50];
        snprintf(Serialbuffer, sizeof(Serialbuffer),
"at+send=lora:5:%s\n\r", data);
        dev.write(Serialbuffer, strlen(Serialbuffer));
        return Read_Lora() > 0;
    }

    void Sleep()
    {
        dev.write("at+set_config=device:sleep:1\r\n",
31);

        Read_Lora();
    }

    void Wake_up()
    {
        dev.write("at+set_config=device:sleep:0\r\n",
31);

        Read_Lora();
    }

private:
    char command[128];

```

```

int tryConnect = 0;
int Read_Lora()
{
    char buffer[300] = {};
    for (int retries = 0; retries < 50 &&
!dev.readable(); ++retries)
    {
        ThisThread::sleep_for(20ms);
    }
    ThisThread::sleep_for(150ms);

    if (dev.read(buffer, sizeof(buffer)) > 0)
    {
        removeNewlines(buffer);
        LOG("%s\n", buffer);
    }

    int numErrorr = 0; // номер ошибки
    const char *errors[6] = {"No Errors", "ERROR: 2",
"ERROR: 99", "ERROR: 86", "ERROR: 80", "OK Join Success"};
    for (int i = 0; i < 6; ++i)
    {
        if (strstr(buffer, errors[i]) != nullptr)
        {
            numErrorr = i;
        }
    }

    switch (numErrorr)
    {
    case 1:
        LOG("Недопустимый параметр в команде AT.\n");
        break;
    case 2:
        LOG("Устройство не подключено к сети
LoRa.\n");

        tryReconnect();
        break;

```

```

        case 3:
            LOG("Не удалось подключиться к сети LoRa.\n");
            tryReconnect();
            break;
        case 4:
            LOG("Приемопередатчик LoRa занят, не удалось
обработать новую команду.\n");
            break;
        case 5:
            LOG("Устройство подключено к сети LoRa.\n");
            tryConnect = 0;
            break;
        default:
            break;
    }

    ThisThread::sleep_for(200ms);
    return numErrorr;
}

void tryReconnect()
{
    if (tryConnect < 8)
    {
        ++tryConnect;
        Connect();
    }
    else
    {
        LOG("Количества попыток достигло %d,
перезагрузка устройства...\n", tryConnect);
        restart();
        tryConnect = 0;
    }
}

void removeNewlines(char *str)
{
    int i = 0, j = 0;

```

```

        while (str[i] != '\0')
        {
            if (str[i] != '\n')
            {
                str[j++] = str[i];
            }
            ++i;
        }
        str[j] = '\0';
    }
};

class Climat
{
private:
    char Temperature[MAX_DIGITS + 1];
    char Pressure[MAX_DIGITS + 1];
    char Humidity[MAX_DIGITS + 1];
    char Voltage[MAX_DIGITS + 1];

    float R1 = 100.0;
    float R2 = 10.0;
    float K = R2 / (R1 + R2);

    float readV = 0;
    float batteryVoltage = 0;

    void getClimat()
    {
        int temperature = bme.getTemperature();
        int pressure = bme.getPressure();
        int humidity = bme.getHumidity();
        int voltage =
static_cast<int>(getBatteryVoltage() * 100);

        LOG("Temp:  %d,  Pressure:  %d,  Humidity:  %d,
Voltage: %d\n", temperature, pressure, humidity, voltage);

```

```

        snprintf(Temperature, sizeof(Temperature), "%d",
temperature);
        snprintf(Pressure, sizeof(Pressure), "%04d",
pressure);
        snprintf(Humidity, sizeof(Humidity), "%d",
humidity);
        snprintf(Voltage, sizeof(Voltage), "%d",
voltage);
    }
    public:
        float getBatteryVoltage()
        {
            readV = ain.read_voltage();
            batteryVoltage = readV / K;
            return batteryVoltage;
        }

        const char *getStringClimat()
        {
            getClimat();
            static char mergedArray[4 * MAX_DIGITS + 3];
            snprintf(mergedArray, sizeof(mergedArray),
"%s%s%s%s", Temperature, Pressure, Humidity, Voltage);

            size_t len = strlen(mergedArray);
            mergedArray[len] = (len % 2 == 0) ? '0' : '1';
            mergedArray[len + 1] = '\0';
            return mergedArray;
        }
    };

    void enterConfigMode() {
        logEnabled = false;
        pc.set_baud(9600);
        printf("Введите команду для настройки
устройства.\n");
        printf("Команды:\n");
        printf("SET_DEVUI:<16-символов DevUI>\n");
    }

```

```

printf("SET_APPUI:<16-символов AppUI>\n");
printf("SET_APPKEY:<32-символа AppKey>\n");
printf("SET_PERIOD:<в секундах>\n");
printf("SET_FLOOR:<этаж>\n");
printf("SET_ROOM:<кабинет>\n");
printf("ENABLE_LOGS\n");
printf("DISABLE_LOGS\n");
printf("EXIT <выход>\n");

char buffer[64];
size_t buffer_index = 0;

while (true) {
    if (pc.readable()) {
        char c;
        pc.read(&c, 1); // Считываем по одному символу

        // Проверка конца строки
        if (c == '\n' || c == '\r') {
            buffer[buffer_index] = '\0'; // Завершаем
строку

            buffer_index = 0; // Сбрасываем индекс для
новой строки

            // Обработка команды
            if (strcmp(buffer, "EXIT") == 0) {
                LOG("Exiting configuration
mode...\n");

                break;
            } else if (strncmp(buffer, "SET_DEVUI:",
10) == 0) {

                strncpy(config.DevUI, buffer + 10,
16);

                config.DevUI[16] = '\0';
                LOG("DevUI set to: %s\n",
config.DevUI);

            } else if (strncmp(buffer, "SET_APPUI:",
10) == 0) {

```

```

        strncpy(config.AppUI, buffer + 10,
16);

        config.AppUI[16] = '\0';
        LOG("AppUI      set      to:      %s\n",
config.AppUI);
    } else if (strncmp(buffer, "SET_APPKEY:",
11) == 0) {
        strncpy(config.AppKey, buffer + 11,
32);

        config.AppKey[32] = '\0';
        LOG("AppKey      set      to:      %s\n",
config.AppKey);
    } else if (strncmp(buffer, "SET_PERIOD:",
11) == 0) {
        config.sendPeriod = atoi(buffer +
11);

        LOG("Send      period      set      to:      %d
seconds\n", config.sendPeriod);
    } else if (strncmp(buffer, "SET_FLOOR:",
10) == 0) {
        strncpy(config.floor, buffer + 10,
4);

        config.floor[4] = '\0';
        LOG("Floor      set      to:      %s\n",
config.floor);
    } else if (strncmp(buffer, "SET_ROOM:", 9)
== 0) {
        strncpy(config.room, buffer + 9, 4);
        config.room[4] = '\0';
        LOG("Room      set      to:      %s\n",
config.room);
    } else if (strcmp(buffer, "ENABLE_LOGS")
== 0) {
        logEnabled = true;
        LOG("Logs enabled.\n");
    } else if (strcmp(buffer, "DISABLE_LOGS")
== 0) {
        logEnabled = false;

```



```

        LOG("Logs disabled.\n");
    } else {
        printf("Unknown command: %s\n",
buffer);
    }
} else if (buffer_index < sizeof(buffer) - 1)
{
    // Добавляем символ в буфер, если он не
переполнен

    buffer[buffer_index++] = c;
}
}

// Сохраняем конфигурацию при выходе из режима
config.confWrite = true;
LOG("Saving configuration to flash...\n");
writeFlash(config);
}
void writeFlash(const DeviceConfig &cfg)
{
    flash.init();

    const uint32_t flash_start = flash.get_flash_start();
    const uint32_t flash_size = flash.get_flash_size();
    const uint32_t flash_end = flash_start + flash_size;
    uint32_t sector_size =
flash.get_sector_size(flash_end - 1);
    uint32_t addr = flash_end - sector_size;
    uint8_t data_write[1] = {1};
    flash.erase(addr, sector_size);
    //flash.program(reinterpret_cast<const void *>(&cfg),
addr, sizeof(DeviceConfig));
    flash.program(data_write, addr, 1);

    flash.deinit();
    LOG("Configuration written to flash.\n");
}

```

```

void readFlash(DeviceConfig &cfg)
{
    flash.init();

    const uint32_t flash_start = flash.get_flash_start();
    const uint32_t flash_size = flash.get_flash_size();
    const uint32_t flash_end = flash_start + flash_size;
    uint32_t          sector_size          =
flash.get_sector_size(flash_end - 1);
    uint32_t addr = flash_end - sector_size;

    //flash.read(reinterpret_cast<void  *>(&cfg),  addr,
sizeof(DeviceConfig));
    flash.read(data_read, addr, 1);

    flash.deinit();
    LOG("Configuration read from flash.\n");
}
/*
static Timer buttonTimer;

void buttonHandlerDown()
{
    if (!mybutton.read()) {
        buttonTimer.start();
    }
}

void buttonHandlerUP()
{
    if (mybutton.read()) {
        buttonTimer.stop();
        auto duration = buttonTimer.elapsed_time();
        buttonTimer.reset();
        if (duration >= CONFIG_MODE_HOLD_TIME) {
            enterConfigModeFlag = true;

```

```

        if (!configThread.get_state()) {
            configThread.start(enterConfigMode);
        }
    }
}
*/
void checkConfigOnStart()
{
    if (data_read[0] == 1)
    {
        LOG("Configuration found: DevUI=%s, AppUI=%s,
AppKey=%s\n", config.DevUI, config.AppUI, config.AppKey);
    }
    else
    {
        LOG("Configuration missing or incomplete.
Entering config mode...\n");
        enterConfigMode(); // Вход в режим конфигурации,
если конфигурация пустая
    }
}

int main()
{
    //confSysClock();
    pc.set_baud(9600);
    float RefV = 3.3;
    ain.set_reference_voltage(RefV);
    bme.initialize();
    Climat climat;
    LOG("Напряжәне: %f\n", climat.getBatteryVoltage());
    if (climat.getBatteryVoltage() > 3.4)
    {
        startdevise = true;
    }
    ThisThread::sleep_for(500ms);
    mybutton.mode(PullUp);
}

```

```

//mybutton.fall(&buttonHandlerDown);
//mybutton.rise(&buttonHandlerUP);

readFlash(config);
checkConfigOnStart();

dev.set_baud(9600);
LoraRAK Lora;
Lora.restart();

LED = 1;
ThisThread::sleep_for(2s);
LED = 0;
Lora.Connect();
int resetCounter = 0;
while (startdevisе)
{
    LED = 1;
    const char *climatData =
climat.getStringClimat();
    Lora.Send_Lora(climatData);
    Lora.Sleep();
    LED = 0;
    ThisThread::sleep_for(config.sendPeriod * 1s);
    Lora.Wake_up();

    resetCounter++;
    if (resetCounter >= 8600) // Перезагрузка через 24
часа (10 секунд цикла)
    {
        printf("Ежедневная перезагрузка...\n");
        NVIC_SystemReset();
    }
}

LOG("Низкое напряжение");

```

```
while (!startdevice)
{
    LED = !LED;
    ThisThread::sleep_for(500ms);
}
return 0;

}
```

Приложение Ж (обязательное)

Блок схема программы устройства

