

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное 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. 730-1
_____ К.В. Подойницын
«__» декабря 2023 г.

Проверил:
Руководитель проекта
Старший преподаватель каф.
КИБЭВС

_____ О.В. Пехов
(оценка) (подпись)
«__» декабря 2023 г.

Принял:
Ответственный за ГПО на кафедре
Доцент каф. БИС, к.т.н.
_____ И.А. Рахманенко
«__» декабря 2023 г.

Исполнители проекта ГПО КИБЭВС-1904:

Студент гр. <u>730-1</u>	_____ К.В. Подойницын
Студент гр. <u>730-1</u>	_____ Г.А. Астра
Студент гр. <u>731-2</u>	_____ Е.В. Демиденко
Студент гр. <u>731-2</u>	_____ А.Д. Коноваленко
Студент гр. <u>711-2</u>	_____ Д.А. Дудник

ф. ГПО-06 Дополненная

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное 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:
 - изучение технологий модуляции LoRa и сетевого протокола LoRaWAN;
 - создание IoT-сети, основанной на данных технологиях.
4. 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.

5. Научная новизна проекта: Нет.
6. Планируемый срок реализации: Получение результатов ожидается к июню 2024 г.
7. Целевая аудитория (потребители): IoT-разработчики.
8. Заинтересованные стороны: ТУСУР.
9. Источники финансирования и материального обеспечения: Нет
10. Ожидаемый результат (полученный товар, услуга): работоспособная IoT-сеть.
11. Руководитель проекта: Пехов О.В
12. Члены проектной группы:
Подойницын Кирилл Вадимович 730-1 (ответственный);
Астра Григорий Алексеевич 730-1;
Коноваленко Александр Дмитриевич 731-2;
Демиденко Егор Вадимович 731-2;
Дудник Дарья Андреевна 711-2.
13. Место выполнения проекта: ул. Красноармейская, д. 146, 4 этаж, ауд. 414.
14. Календарный план выполнения проекта:

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	

1	Тестирование платы с датчиком вне клиент-серверного приложения	Проведение тестирования платы и датчика вне клиент-серверного приложения	15.09.2023	26.10.2023	Получены логи тестирования датчиков и проверена корректность работы платы
2	Решение проблемы с некорректной передачей температуры от платы	Определение причины ошибки на основе тестирования и её исправление	26.10.2023	30.11.2023	Написана новая прошивка, корректно передающая значения температуры
3	Тестирование устройства с web-приложением после исправления ошибки	Отправка данных с устройства на сервер	30.11.2023	14.12.2023	На сервере были получены и отображены корректные значения температуры
4	Освоение моделирования в приложении Blender	Получение основных навыков моделирования в приложении Blender	15.09.2023	26.10.2023	Навыки 3d-моделирования в программном обеспечении Blender
5	Доработка конечного корпуса устройства	Исправить ошибки предыдущего корпуса устройства	30.11.2023	14.12.2023	Конечный вариант корпуса устройства

6	Решение проблемы отсутствия подключения платы к БС со стороны базовой станции	Решить проблему подключения базовой станции к коммутационной плате устройства	15.09.2023	26.10.2023	Решена проблема
7	Подбор выделенного сервера для переноса веб-приложения	Сравнение и анализ серверов для переноса веб-приложения	26.10.2023	30.11.2023	Выбран веб-сервер
8	Настройка серверной части базовой станции на выделенном сервере	Изучение и настройка параметров серверной части базовой станции на выделенном сервере	30.11.2023	14.12.2023	Рабочая серверная часть базовой станции
№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
9	Получение сертификата для реализации уведомлений	Создание SSL-сертификата и его последующая интеграция в код создания веб-ссылки приложения	15.09.2023	26.10.2023	Сформирован SSL-сертификат, и успешно интегрирован в код приложения

10	Реализация сброса пароля пользователя	Создание и описание на языке Python форм обработки данных, необходимых для получения ссылки на форму сброса пароля	26.10.2023	30.11.2023	Созданы формы обработки вводимых данных, создан почтовый сервер для отправки письма со ссылкой сброса пароля
11	Тестирование получения данных и корректного отображения графика	Проведение тестирования получения данных с устройства и отображение их на графике	15.09.2023	26.10.2023	Функционирующий график с новыми данными
12	Создание вкладки с уведомлениями о температуре и ее настройка	Реализация вкладки для уведомлений	26.10.2023	30.11.2023	Готовая вкладка с уведомлениями о температуре
13	Изучение API CHIRPSTACK	Ознакомление с API функциями ChirpStack	30.11.2023	14.12.2023	Список функций необходимых для перехода на ChirpStack
14	Отчётность	Написание отчёта по ГПО	14.12.2026	21.12.2023	Написан отчёт по проделанной работе.

«07» сентября 2023 г.

Руководитель проекта:

Старший преподаватель каф. КИБЭВС

(должность)

Пехов О.В.

(подпись)

(расшифровка)

Члены проектной группы:

Подойницын К.В.

(подпись)

(расшифровка)

Астра Г.А.

(подпись)

(расшифровка)

Демиденко Е.В.

(подпись)

(расшифровка)

Коноваленко А.Д.

(подпись)

Дудник Д.А.

(подпись)

Реферат

Отчет содержит 76 страниц, 59 рисунков, 19 источников.

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 технологии беспроводной передачи данных LPWAN, в частности технологии LoRa;

- Разработка системы сбора микроклиматических параметров в учебных аудиториях.

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

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

Оглавление

Введение	11
1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	13
2 ПРАКТИЧЕСКАЯ ЧАСТЬ	18
2.1 Тестирование датчика на корректность работы	18
2.2 Исправление ошибки с некорректной температурой	21
2.3 Тестирование устройства после исправления ошибки	25
2.4 Общие элементы настройки серверной части базовой станции	27
2.5 Настройка сетевого сервера IOT Vega Server для базовой станции	27
2.6 Настройка сетевого сервера ChirpStack для базовой станции	32
2.7 Сравнение сервисов веб-хостинга для переноса веб-приложения	39

2.8 Получение сертификата для реализации уведомлений	41
2.9 Реализация сброса пароля пользователя	44
2.10 Тестирование работы графика с новыми данными	49
2.11 Освоение моделирования в приложении Blender	52
2.12 Доработка конечного корпуса устройства	53
Заключение	56
Список источников	57
Приложение А	59
Приложение Б	60
Приложение В	61
Приложение Г	63
Приложение Д	65
Приложение Е	67
Приложение Ж	68
Приложение З	79
Приложение И	84

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. По оценкам IDC, в 2023 году объём IoT-рынка в Европе составит около 227 миллиардов долларов. В дальнейшем ожидается показатель CAGR (среднегодовой темп роста в сложных процентах) на уровне 11 %. В результате, к 2027-му затраты достигнут почти 345 миллиардов долларов.

Отмечается, что развитие сектора IoT в Европе отражает меняющиеся цели предприятий в области цифровой трансформации. Главными задачами являются снижение затрат, оптимизация процессов, внедрение средств автоматизации и повышение качества обслуживания клиентов.

В России же к 2030 году вложат 90 миллиардов в промышленных целях. Объем рынка отечественных решений на базе технологии IIoT к 2030 году должен достигнуть 147,25 миллиардов рублей против 79,6 миллиардов рублей в 2022 году. По итогам года рынок оценивался в 50–80 миллиардов рублей. При этом доля рынка отечественных промышленных решений к 2030 году должна составить 82,74%, в 2022-м она была 54,1%, указано в дорожной карте.

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. Облачные сервера будут еще долго оставаться в направлении развития и использования в сфере IoT, но уже и они перестают быть передовыми технологиями – сервисы известных компаний (например,

Amazon) позволяют разработчикам выполнять машинное обучение и вычислять задачи непосредственно на оконечных устройствах интернета вещей, дабы избежать нежелательных задержек передачи 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 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

На рисунках 1.1-1.6 представлены примеры использования API ChirpStack с официального сайта. Мы рассматриваем данную API так как в будущем планируется перенос с сервера Vega на ChirpStack из-за того, что с сайта Вега пропадали загрузки Vega Server. Переменная `server` отвечает за запуск сервера на определенном IP адресе. В `dev_eui` указывается уникальный идентификатор устройства. `Channel` создает канал связи с сервером. `Req` выполняет запрос на сервер, а `resp` получает ответ с сервера. Далее рассмотрим некоторый функционал библиотеки `grpc`.

```
import os
import sys

import grpc
from chirpstack_api import api

# Configuration.

# This must point to the API interface.
server = "localhost:8080"

# The DevEUI for which you want to enqueue the downlink.
dev_eui = "01010101010101"

# The API token (retrieved using the web-interface).
api_token = "...".

if __name__ == "__main__":
    # Connect without using TLS.
    channel = grpc.insecure_channel(server)

    # Device-queue API client.
    client = api.DeviceServiceStub(channel)

    # Define the API key meta-data.
    auth_token = [("authorization", "Bearer %s" % api_token)]

    # Construct request.
    req = api.EnqueueDeviceQueueItemRequest()
    req.queue_item.confirmed = False
    req.queue_item.data = bytes([0x01, 0x02, 0x03])
    req.queue_item.dev_eui = dev_eui
    req.queue_item.f_port = 10

    resp = client.Enqueue(req, metadata=auth_token)

    # Print the downlink id
    print(resp.id)
```

Рисунок 1.1 – Пример использования API ChirpStack

Создать клиента

ГРПК.**insecure_channel**(*target*, *options=None*, *compression=None*)

[исходник]

Создает небезопасный канал к серверу.

Возвращаемый канал является потокобезопасным.

Параметры:

- **target** – Адрес сервера
- **options** — необязательный список пар "ключ-значение" ([channel_arguments](#) в среде выполнения gRPC Core) для настройки канала.
- **compression** — необязательное значение, указывающее на метод сжатия. Используется в течение всего срока службы канала.

Возвращает: Канал.

Рисунок 1.2 – Создание канала связи с сервером

Создание учетных данных клиента

ГРПК.**ssl_channel_credentials**(*root_certificates=Нет*, *private_key=Нет*, *certificate_chain=Нет*) ¶

[исходник]

Создает ChannelCredentials для использования с каналом с поддержкой SSL.

Параметры:

- **root_certificates** – Корневые сертификаты в PEM-кодировке в виде байтовой строки, или Нет, чтобы извлечь их из расположения по умолчанию, выбранного gRPC Среда выполнения.
- **private_key** – закрытый ключ в PEM-кодировке в виде байтовой строки или None, если нет. Следует использовать закрытый ключ.
- **certificate_chain** – Цепочка сертификатов в кодировке PEM в виде байтовой строки или None, если цепочка сертификатов не должна использоваться.

Возвращает: Объект ChannelCredentials для использования с каналом с поддержкой SSL.

Рисунок 1.3 – Канал с поддержкой SSL

Создание учетных данных сервера

```
ГРПК.ssl_server_credentials(private_key_certificate_chain_pairs,  
root_certificates=Нем, require_client_auth=Ложь)
```

[исходник]

Создает `ServerCredentials` для использования с сервером с поддержкой SSL.

- Параметры:**
- **`private_key_certificate_chain_pairs`** – Список пар вида [Закрытый ключ в кодировке PEM, цепочка сертификатов в кодировке PEM].
 - **`root_certificates`** – Необязательная строка байтов корневого корня клиента в PEM-кодировке Сертификаты, которые сервер будет использовать для проверки подлинности клиента. Если `require_client_auth` опущен, он также должен иметь значение `False`.
 - **`require_client_auth`** – Логическое значение, указывающее, требовать или нет клиентов для аутентификации. Может быть истинным только в том случае, если `root_certificates` не является `Нет`.

Возвращает: `ServerCredentials` для использования с сервером с поддержкой SSL. Как правило, это объект является аргументом метода `add_secure_port()` при настройке сервера.

```
ГРПК.ssl_server_certificate_configuration(private_key_certificate_chain_pairs,  
root_certificates=Нем)
```

[исходник]

Создает `ServerCertificateConfiguration` для использования с сервером.

- Параметры:**
- **`private_key_certificate_chain_pairs`** – Коллекция пар вид [закрытый ключ в кодировке PEM, сертификат в кодировке PEM цепь].
 - **`root_certificates`** – Необязательная строка байтов корневого корня клиента в PEM-кодировке Сертификаты, которые сервер будет использовать для проверки подлинности клиента.

Возвращает: Объект `ServerCertificateConfiguration`, который может быть возвращен в сертификате
обратный вызов получения конфигурации.

Рисунок 1.4 – Создание учетных данных сервера с использованием протокола SSL

Класс `GRPC.AuthMetadataContext` [исходник]

Предоставляет сведения для вызова подключаемых модулей метаданных учетных данных.

service_url

Строковый URL-адрес вызываемой службы.

method_name

Строка с полным именем вызываемого метода.

Класс `GRPC.AuthMetadataPluginCallback` [исходник]

Объект обратного вызова, полученный плагином метаданных.

__call__(*метаданные, ошибка*) [исходник]

Передает метаданные проверки подлинности среды выполнения gRPC для RPC.

Параметры:

- **metadata** — [метаданные](#), используемые для создания CallCredentials.
- **error** — исключение для указания на ошибку или None для указания на успешное выполнение.

Рисунок 1.5 – Работа с метаданными

Класс `GRPC.Плагин AuthMetadataPlugin` [исходник]

Спецификация для пользовательской проверки подлинности.

__call__(*контекст, обратный вызов*) [исходник]

Реализует проверку подлинности путем передачи метаданных обратному вызову.

Этот метод будет вызываться асинхронно в отдельном потоке.

Параметры:

- **context** – `AuthMetadataContext`, предоставляющий информацию о RPC, который Подключаемый модуль вызывается для аутентификации.
- **callback** – `AuthMetadataPluginCallback`, который должен быть вызван либо синхронно или асинхронно.

Класс `GRPC.ServerCredentials`(*учетные данные*) [исходник]

Инкапсуляция данных, необходимых для открытия защищенного порта на сервере.

Этот класс не имеет поддерживаемого интерфейса - он существует для определения типа его класса `instances` и его экземпляры существуют для того, чтобы передаваться другим функциям.

Класс `GRPC.ServerCertificateConfiguration`(*certificate_configuration*) [исходник]

Конфигурация сертификата для использования с сервером с поддержкой SSL.

Экземпляры этого класса могут быть возвращены в конфигурации сертификата Получение обратного вызова.

Этот класс не имеет поддерживаемого интерфейса – он существует для того, чтобы определить его экземпляры и его экземпляры существуют для передачи в другие функции.

Рисунок 1.6 – Пользовательская проверка подлинности, инкапсуляция данных и конфигурация сертификата

Контекст на стороне клиента

Класс `grpc.aio.Звать`

[исходник]

Абстрактный базовый класс RPC на стороне клиента.

Абстрактный асинхронный `код()`

[исходник]

Обращается к коду состояния, отправленному сервером.

Возвращает: Значение `StatusCode` для RPC.

Тип возвращаемого значения: [СтатусКод](#)

Абстрактные асинхронные `детали()`

[исходник]

Обращается к сведениям, отправленным сервером.

Возвращает: Строка сведений RPC.

Тип возвращаемого значения: Ул

абстрактный асинхронный `initial_metadata()`

[исходник]

Обращается к исходным метаданным, отправленным сервером.

Возвращает: Исходные [метаданные](#).

Тип возвращаемого значения: [Метаданные](#)

абстрактный асинхронный `trailing_metadata()`

[исходник]

Обращается к конечным метаданным, отправленным сервером.

Возвращает: [Конечные метаданные](#).

Тип возвращаемого значения: [Метаданные](#)

Рисунок 1.7 – Контекст на стороне клиента

В результате изучения API мы пришли к выводу, что API описана не так подробно как у Веги и для переноса будет необходимо переписывать приличное количество кода. Поэтому мы считаем переход на другую API не целесообразным.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Тестирование датчика на корректность работы

Проблема заключалась в том, что на графике температур случайным образом отображалось некорректное значение температуры (рисунок 2.1).

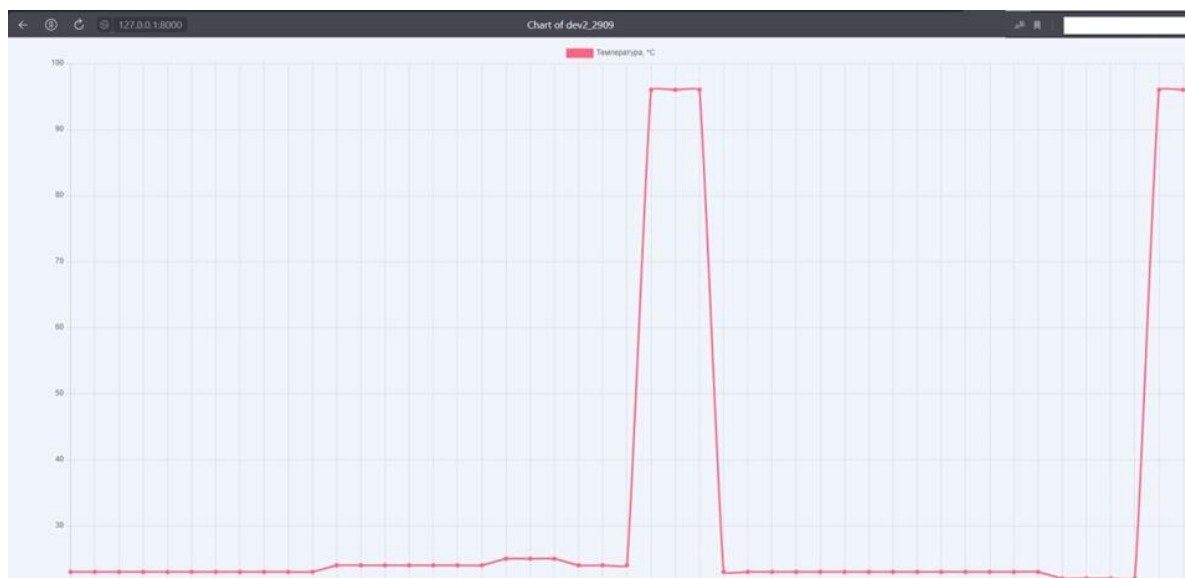


Рисунок 2.1 - График с некорректными значениями температуры

Для решения данной проблемы было установлено две возможные причины возникновения ошибки:

- Некорректная работа прошивки или измерительного датчика;
- Некорректная обработка полученных данных сервером.

Проверка прошивки и датчика температур проводилась в несколько этапов:

- Тестирование датчика температур отдельной прошивкой без отправки значений;
- Тестирование с использованием прошивки и отправкой данных.

Для исключения некорректного измерения температуры был выполнен тест нескольких модулей ВМЕ280, и произведено сравнение полученных логов (рисунок 2.2-2.3).

```
===== PuTTY log 2023.10.25 22:06:04 =====  
-----  
temp 30  
-----  
-----  
temp 30  
-----  
-----  
temp 30  
-----  
-----  
temp 29  
-----  
-----  
temp 29  
-----
```

Рисунок 2.2 - Лог корректно работающего модуля

```
===== PuTTY log 2023.10.25 22:28:03 =====  
-----  
temp -179  
-----  
-----  
temp -179  
-----  
-----  
temp -179  
-----  
-----  
temp -179  
-----  
-----  
temp -179  
-----  
-----  
temp -179  
-----
```

Рисунок 2.3 - Лог некорректно работающего модуля

По итогам тестирования был выявлен некорректно работающий модуль. Это проявляется в неправильном измерении температуры при одинаковом коде прошивки.

Код прошивки представлен в Приложении А.

Далее была выполнена попытка тестирования основной прошивки с исправным датчиком. Так как прошивка была написана на версии MbedOS5, то при сборе прошивки на новой версии, выдавалась ошибка, указывающая на драйвер RawSerial для работы с UART.

В документации к драйверу было указано, что он больше не поддерживается (рисунок 2.4).

RawSerial



Note: This API has been deprecated in favor of `UnbufferedSerial`.

Рисунок 2.4 - Информация о прекращении поддержки драйвера

Так как используемый драйвер в новой версии не поддерживался, то были рассмотрены доступные драйверы `BufferedSerial` и `UnbufferedSerial`.

Драйвер `BufferedSerial` для работы использует настраиваемый буфер, который служит для передачи значений от платы на UART интерфейс

`UnbufferedSerial` в работе не использует буферы для отправки и приема данных. Данный драйвер позволяет отправлять и получать данные только побайтово, за одну отправку или чтение работая только с одним байтом.

2.2 Исправление ошибки с некорректной температурой

Для исправления ошибки был выбран драйвер `BufferedSerial`, чтобы целиком отправлять команды для приемопередатчика RAK811, без деления на байты.

Поскольку в старой прошивке для записи данных использовалась функция `printf()`, то для работы с буфером и записи в него используется функция `snprintf()`. Данная функция имеет вид:

```
int snprintf (char * buffer, buf_size, const char * format,...);
```

Где в `char * buffer` указывается буфер для записи, в `buf_size` указывается размер буфера, а в `const char * format` указываются данные, которые будут записаны в буфер.

Для того, чтобы разобраться в работе драйвера была написана простая прошивка с его использованием и отправкой данных по UART (рисунок 2.5*).

```
#include "mbed.h"

// Объект для работы с UART
BufferedSerial dev (D8, D2);

int main()
{
    // Установка скорости передачи данных (бод)
    dev.set_baud(115200);

    // Основной цикл программы
    while (1) {
        int data = 42;

        // Отправка данных через UART
        char buffer[32];
        int len = snprintf(buffer, sizeof(buffer), "Data: %d\r\n", data);
        dev.write(buffer, len);

        // Задержка перед следующей отправкой данных
        ThisThread::sleep_for(1s);
    }
}
```

Рисунок 2.5 - Прошивка с драйвером BufferedSerial

В результате выполнения в терминал PuTTY при прослушивании UART преобразователя выводилась строка “Data: 42”.

Работа прошивки представлена на рисунке 2.6.

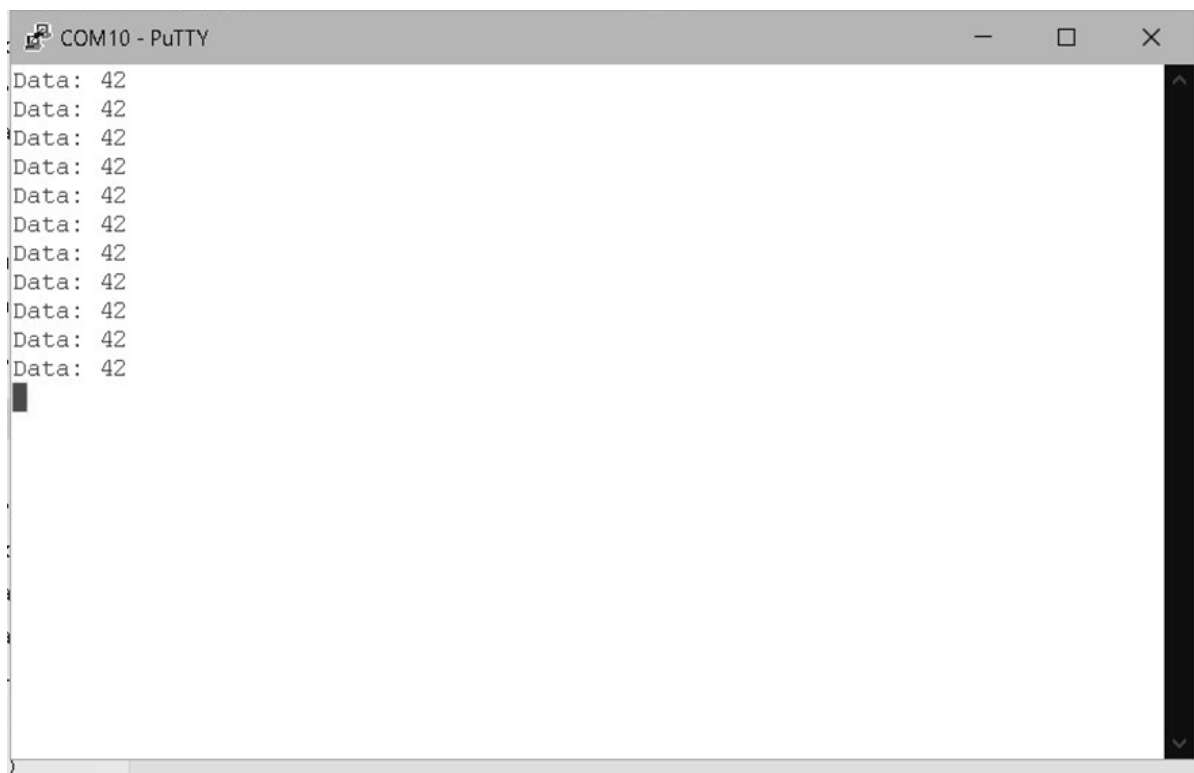


Рисунок 2.6 - Результат работы прошивки

Затем к прошивке были добавлены команды для взаимодействия с датчиком температур, а также была добавлена отправка команд для приемопередатчика на подключение и отправку данных.

После добавления команд код стал таким

```

#include "mbed.h"
#include "BME280.h"
#include <stdio>

BufferedSerial dev (D8, D2); // Объект для работы с UART
BME280 bme (I2C_SDA, I2C_SCL);

int main()
{
    dev.set_baud(115200); // Установка скорости передачи данных
    bme.initialize();
    char buffer[32];

    int len;
    len = snprintf(buffer, sizeof(buffer), "at+join\r\n");
    dev.write(buffer, len);
    ThisThread::sleep_for(5s);

    while (1)
    {
        int data = bme.getTemperature();
        char data2 = data;

        // Отправка данных через UART
        len = snprintf(buffer, sizeof(buffer), "at+send=lora:5:%d\r\n", data2);
        dev.write(buffer, len);

        // Задержка перед следующей отправкой данных
        ThisThread::sleep_for(10s);
    }
}

```

Рисунок 2.7 - Код новой прошивки

В данном коде сначала производится обозначение пинов для работы с UART интерфейсом и для работы с температурным датчиком BME280. Затем производится установка скорости передачи данных, инициализация датчика и задание размера буфера **buffer**. Далее задается переменная **len**, в которой передается длина строки, и выполняется отправка команды **at+join** на подключение приемопередатчика к станции.

Спустя 5 секунд запускается бесконечный цикл, в начале которого в переменную **data** записывается значение температуры с датчика с преобразованием в символьную переменную **data2**. Затем выполняется

отправка данных через UART с помощью команды **at+send=lora:5**, где 5 - номер порта, и выполняется ожидание в течение 10 секунд.

2.3 Тестирование устройства после исправления ошибки

После написания прошивки было выполнено её тестирование с подключением и отправкой данных на сервер. После отправки более 100 результатов измерений температур в различных условиях не было выявлено некорректных значений. Что означает, что ошибка была в некорректно написанной прошивке.

На рисунках 2.8-2.9 представлено сообщение при успешном подключении устройства и сообщение при получении данных с платы.

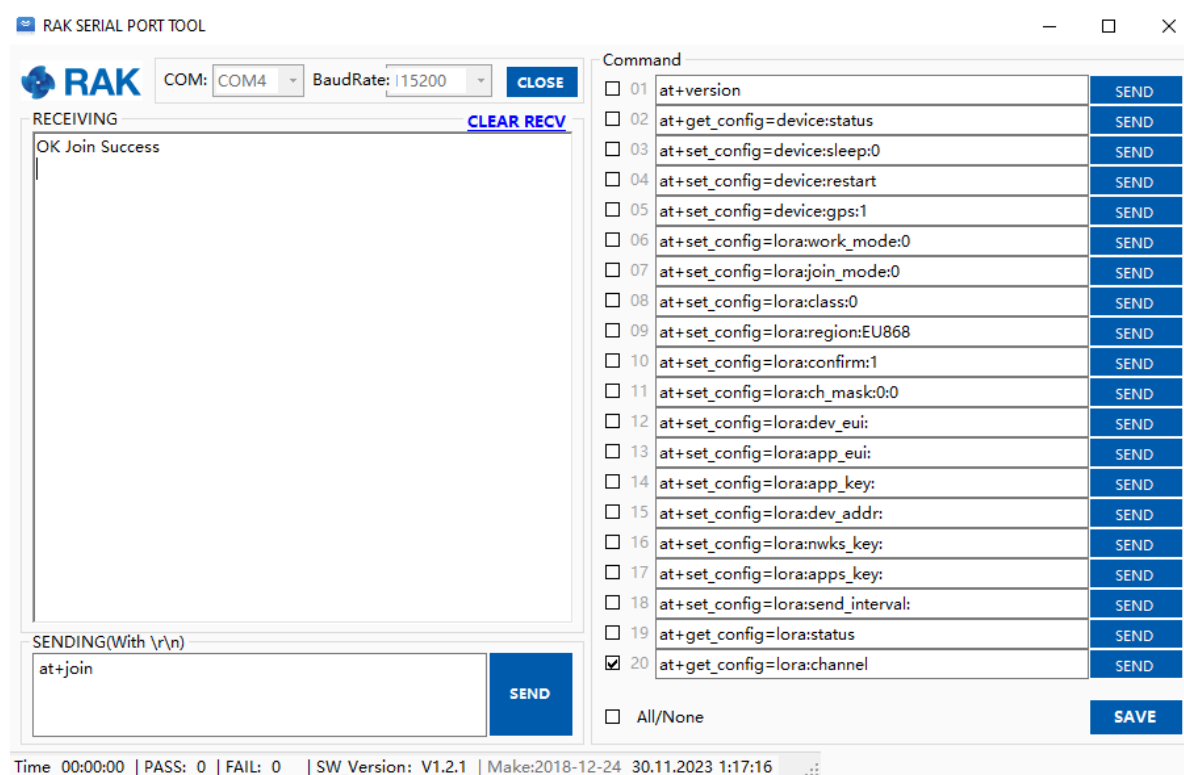


Рисунок 2.8 - Успешное подключение устройства

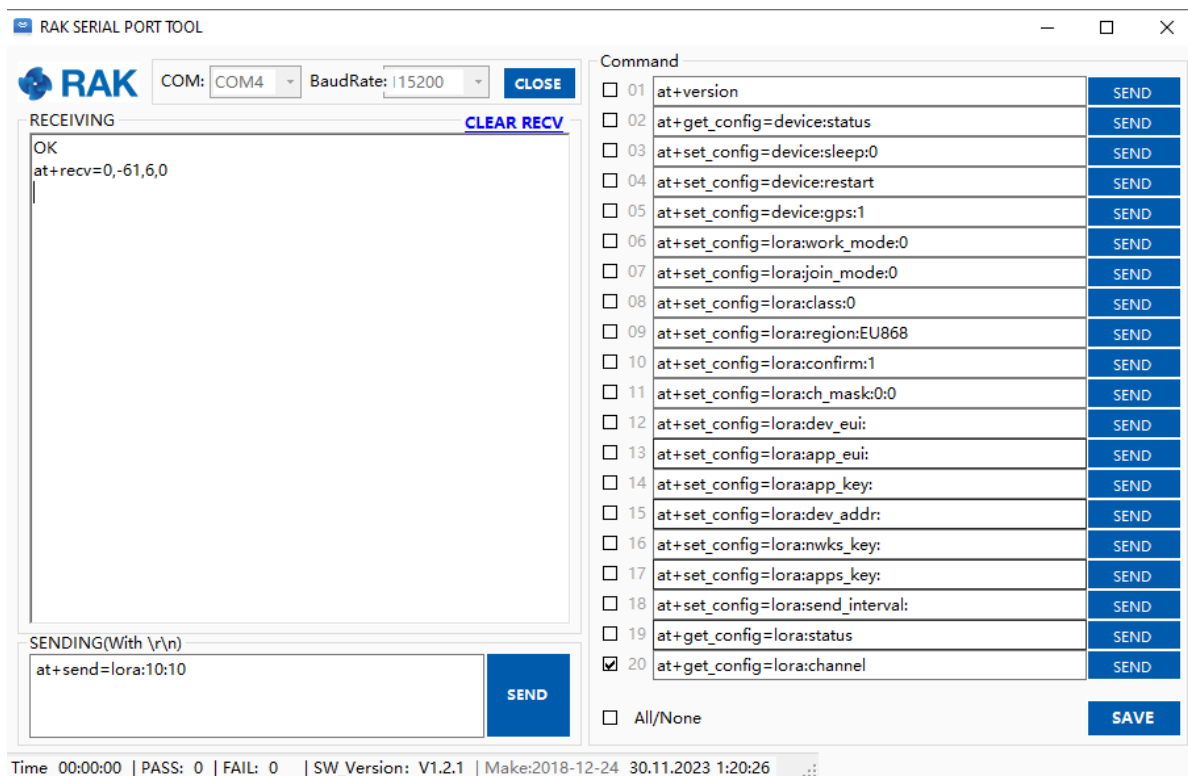


Рисунок 2.9 - Сообщение при получении данных с платы

На рисунке 2.10 представлены полученные данные на сервере

ack	data	dr	fcnt	freq	gatewayid
0	27	SF7 BW125 4/5	252	868500000	00006CC3743EE7C7
0	28	SF7 BW125 4/5	251	867300000	00006CC3743EE7C7
0	28	SF7 BW125 4/5	250	867500000	00006CC3743EE7C7
0	28	SF7 BW125 4/5	249	868100000	00006CC3743EE7C7
0	28	SF7 BW125 4/5	248	868500000	00006CC3743EE7C7
0	29	SF7 BW125 4/5	247	867300000	00006CC3743EE7C7
0	30	SF7 BW125 4/5	246	867500000	00006CC3743EE7C7
0	31	SF7 BW125 4/5	245	868100000	00006CC3743EE7C7
0	26	SF7 BW125 4/5	244	868500000	00006CC3743EE7C7
0	26	SF7 BW125 4/5	243	867100000	00006CC3743EE7C7
0	26	SF7 BW125 4/5	242	867700000	00006CC3743EE7C7
0	26	SF7 BW125 4/5	241	868100000	00006CC3743EE7C7

Рисунок 2.10 - Полученные данные на сервере

Код прошивки представлен в Приложении Б.

Историю версий прошивки можно посмотреть в репозитории GitHub [**].

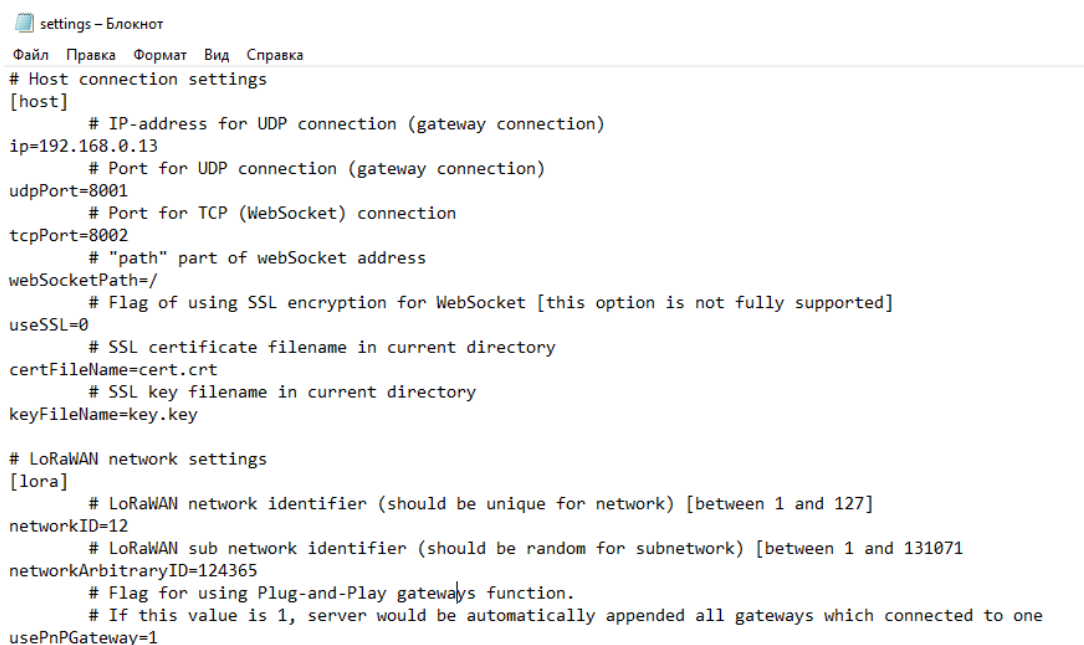
2.4 Общие элементы настройки серверной части базовой станции

Настройка серверной части базовой станции включает в себя ряд общих элементов, которые обеспечивают корректную работу сети. Основными из них являются:

1. Выбор LoRaWAN-сервера, такого как The Things Network (TTN), IOT Vega Server, ChirpStack и т.д.
2. Регистрация базовой станции
3. Задание частотных диапазонов и планов для базовой станции
4. Настройка сетевых параметров, таких как адрес сервера, порт и протокол связи
5. Добавление устройств передачи данных
6. Настройка параметров устройств передачи данных
7. Настройка секретных ключей для обеспечения безопасности передачи данных.

2.5 Настройка сетевого сервера IOT Vega Server для базовой станции

Для настройки сетевого сервера «IOT Vega Server» требуется установить загрузочную папку для своей операционной системы с официального сайта. В загрузочной папке необходимо изменить файл «settings.txt» вписав в него IP-адрес для сервера, а также необходимые порты подключения, что представлено на рисунке 2.11.



```


settings – Блокнот
Файл  Правка  Формат  Вид  Справка
# Host connection settings
[host]
# IP-address for UDP connection (gateway connection)
ip=192.168.0.13
# Port for UDP connection (gateway connection)
udpPort=8001
# Port for TCP (WebSocket) connection
tcpPort=8002
# "path" part of webSocket address
webSocketPath=/
# Flag of using SSL encryption for WebSocket [this option is not fully supported]
useSSL=0
# SSL certificate filename in current directory
certFileName=cert.crt
# SSL key filename in current directory
keyFileName=key.key

# LoRaWAN network settings
[lora]
# LoRaWAN network identifier (should be unique for network) [between 1 and 127]
networkID=12
# LoRaWAN sub network identifier (should be random for subnetwork) [between 1 and 131071]
networkArbitraryID=124365
# Flag for using Plug-and-Play gateways function.
# If this value is 1, server would be automatically appended all gateways which connected to one
usePnPGateway=1

```

Рисунок 2.11 - Файл конфигурации IOT Vega Server

Далее необходимо запустить файл `iot-vega-server.exe`, что представлено на рисунке 2.12.



```

C:\Users\konov\OneDrive\Рабочий стол\ГПО\IOT Vega Server (win) v1.2.1 (1)\IOT Vega Server (win) v1.2.1\iot-vega-server.exe
IOT Vega Server 1.2.1
LOG: current file for log messages [./history_1.log]
INFO: Table queueTransmit is cleaned
INFO: Table "bufMacDevParams" is cleared
INFO: DB-secure scanner started...
DEBUG [CDevicesCountInfo]: vega[0], totalNonVega[1000], usedNonVega[1]
INFO: DB-secure scanner successfully finished
UDP socket has opened. IP[192.168.0.13:8001]
WebSocketServer has opened. Port[8002]
DEBUG: UdpServer handler is started

```

Рисунок 2.12 - Запуск сервера

Для подключения базовой станции к серверу необходимо в веб-приложении базовой станции выставить IP-адрес сервера и указать значение верхнего и нижнего порта как значение параметра «Port for UDP connection» из файла конфигурации, что представлено на рисунке 2.13.



Настройки подключения к серверу LoRaWAN

Настройки частотного плана LoRa

Настройки GPS для LoRa

Настройки подключения к серверу LoRaWAN

Адрес сервера	Верхний порт	Нижний порт
192.168.0.13	8001	8001

Рисунок 2.13 - Подключения базовой станции к серверу

Для того чтобы зайти на сервер необходимо с официального сайта скачать инструмент «Admin Tool», в файле «config» которого указать адрес сервера с портом значения которого совпадает со значением параметра «Port for TCP» из файла конфигурации сервера, что представлено на рисунке 2.14.

config – Блокнот

Файл Правка Формат Вид Справка

```
const address_ws = 'ws://192.168.0.13:8002';
const demo_user = false;
const select_server = true;
const stock_address_ws = ['ws://192.168.0.13:8002', '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'
  }
}
// const map_tiles_leaflet = {
//   url: "https://{s}.tile.thunderforest.com/spinal-map/{z}/{x}/{y}.png",
//   options: {
//     attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
//   }
// }
```

Рисунок 2.14 - Файл конфигурации инструмента Admin Tool

После чего можно запустить веб-страницу «auth» из данной папки. Авторизовавшись в которой и перейдя в раздел «Gateway» можно наблюдать подключенную базовую станцию, что представлено на рисунке 2.15.

Home Devices Gateways Users Exit							
<input type="text"/>				CONNECTED GATEWAYS			+ Add new gateway
<input type="checkbox"/>	#	Name	Gateway ID	Active	Latency		
<input type="checkbox"/>	1	PnP_GW_00006CC3743EE7C7	00006CC3743EE7C7		0		

Рисунок 2.15 - Подключенная базовая станция

Для добавления устройства передачи данных, а именно коммутационной платы RAK811 необходимо перейти на вкладку «Devices» и нажать кнопку «Add new device». В всплывающем меню необходимо заполнить область «Main settings», в которой обязательно указываются DevEui – EUI идентификатор устройства и End-device class – класс устройства. Так как в данном проекте используется спецификация активации ОТАА, то необходимо заполнить область «Over-the-air activation (ОТАА)», в которой обязательно указываются AppEUI – EUI идентификатор приложения устройства и AppKey – ключ приложения устройства. После чего устройство будет добавлено на сервер. Настройки устройства представлены на рисунке 2.16

Activation by personalisation (ABP)

End-device address (devAddr)
DEVADDR

Application session key (AppSKey)
APPSKEY

Network session key (NwkSKey)
NWKSKEY

Over-the-air activation (OTAA)

Application identifier (AppEUI)
AC1F09FFF8680811

Application key (AppKey)
AC1F09FFFE015302AC1F09FFF8680811

Main settings

End-device name
device name

End-device identifier (DevEUI)
AC1F09FFFE015302

End-device class
Class A

End-device group
device group

Regional settings

Frequency plan
EU868

Nº	Frequency	Enabled
1	FIXED	✓
2	FIXED	✓
3	FIXED	✓
4	867100000	✓
5	867300000	✓
6	867500000	✓
7	867700000	✓
8	867900000	✓

RX2 Frequency, Hz
869525000

☐ Expert settings

Close

Save

Рисунок 2.16 - Настройки подключенного устройства

Для проверки устройства необходимо через официальное приложение «RAK Serial Port Tool» подключить устройство командой «at+join», после чего как на сервере, так и в приложении появиться сообщение об успешном подключении, что представлено на рисунках 2.17 - 2.18.

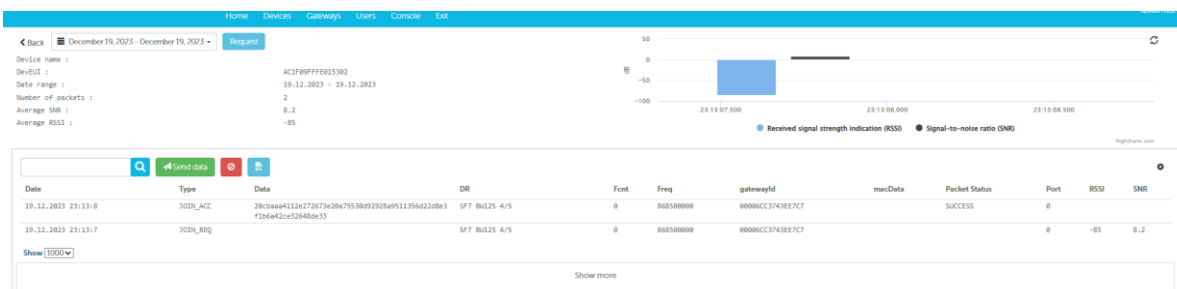


Рисунок 2.17 - Сообщение об успешном подключении на сервере

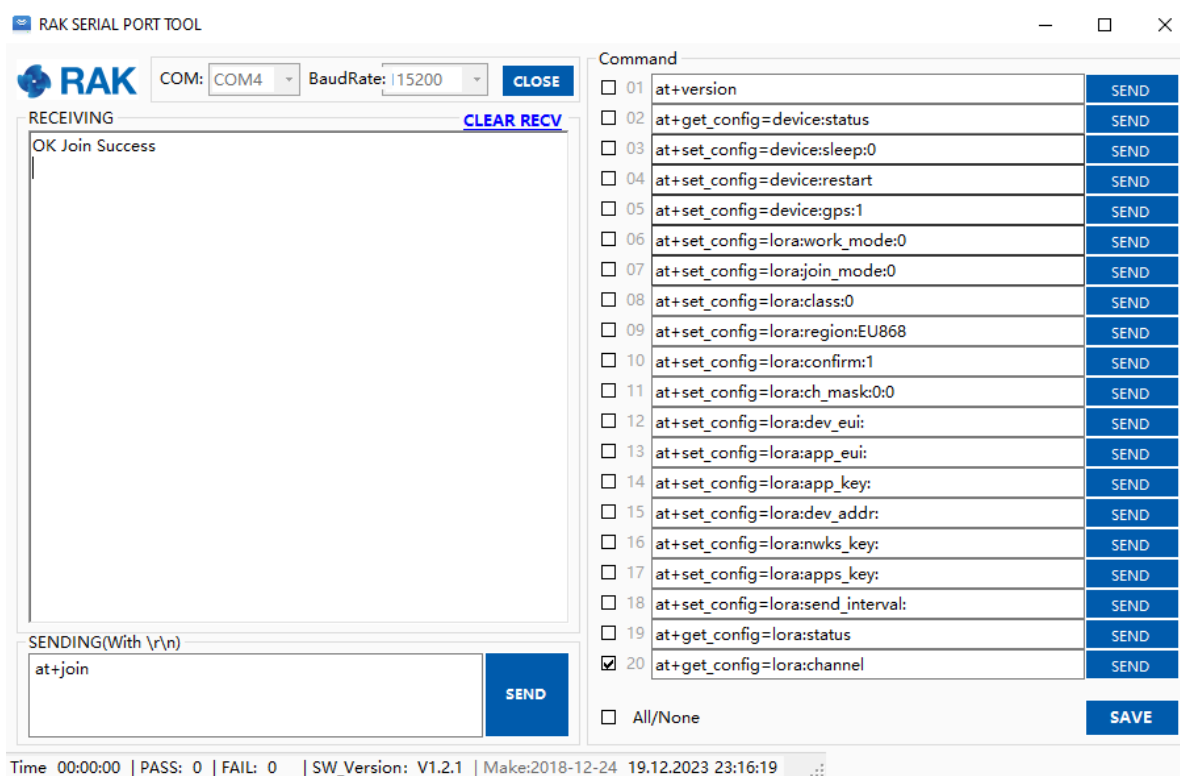


Рисунок 2.18 - Сообщение об успешном подключении в приложении

2.6 Настройка сетевого сервера ChirpStack для базовой станции

Для настройки сетевого сервера ChirpStack требуется установить на виртуальную машину с операционной системой Ubuntu дистрибутивы компонентов ChirpStack: Gateway Bridge — мост между программой Packet Forwarder установленной на базовой станции и структурой сервера LoRaWAN, Network Server — сетевой сервер, обрабатывающий

сообщения сетевого уровня, Application Server — сервер приложения обеспечивающий работу сети на пользовательском уровне.

Также необходимо установить сторонние программные модули: брокер MQTT mosquitto для внутреннего обмена сообщениями между составляющими сервера, СУБД PostgreSQL база данных для постоянного хранения данных и СУБД Redis промежуточная база данных для хранения скоротечных данных. После необходимо запустить компоненты сервера специальными терминальными программами. Итогом работы должна получиться возможность перехода в приложение по адресу виртуальной машины с портом под номером 8080, что представлено на рисунке 2.19.

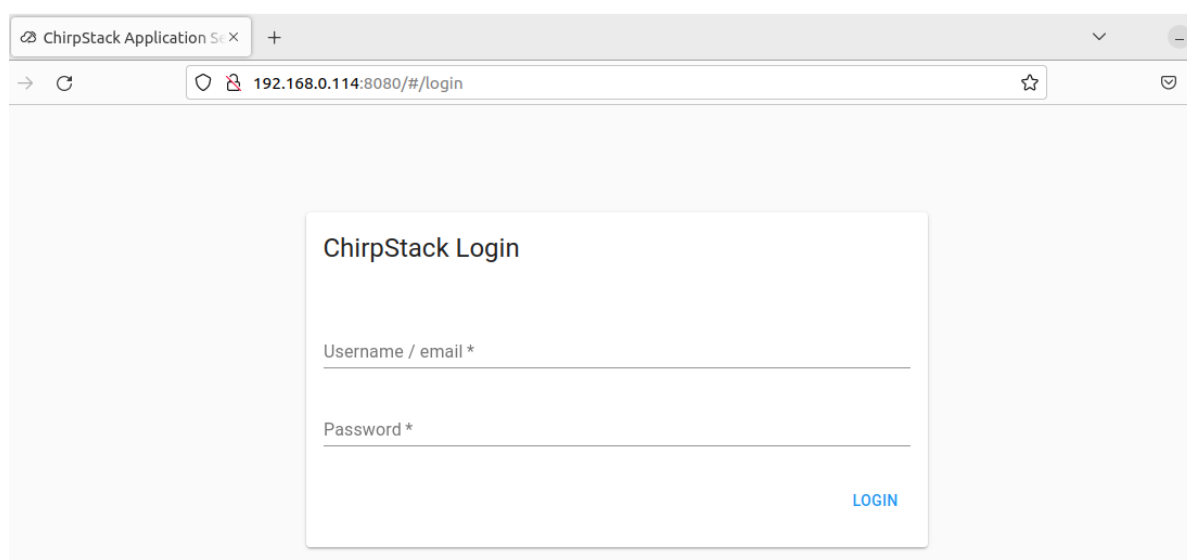


Рисунок 2.19 - Приложение ChirpStack

Далее настройка происходит в данном графическом интерфейсе. Первоочередно редактируется раздел «Network-servers», куда добавляется адрес сервера сети, что представлено на рисунке 2.20.

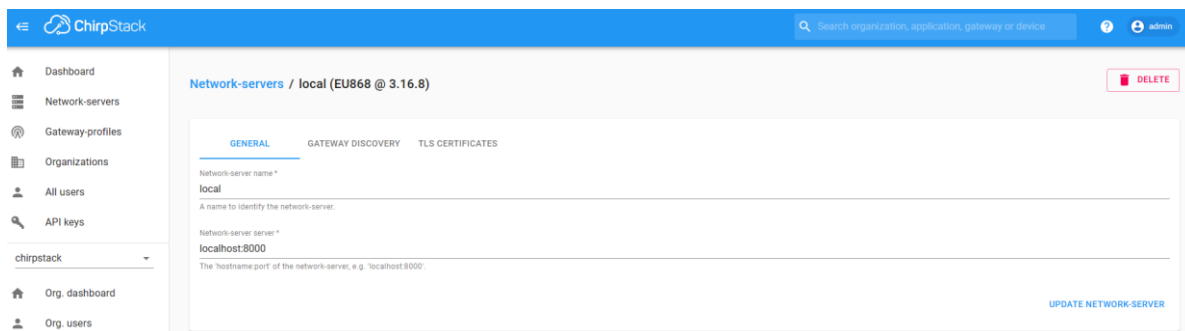


Рисунок 2.20 - Раздел Network-servers

Следом создается профиль для базовой станции в разделе «Gateway-profiles», что представлено на рисунке 2.21.

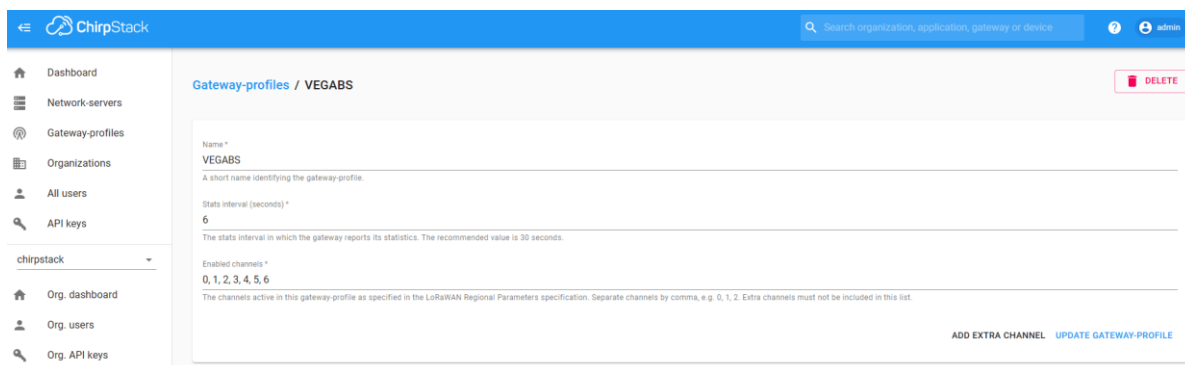


Рисунок 2.21 - Раздел Gateway-profiles

В разделе «Service-profiles» создается профиль взаимодействия, что представлено на рисунке 2.22.

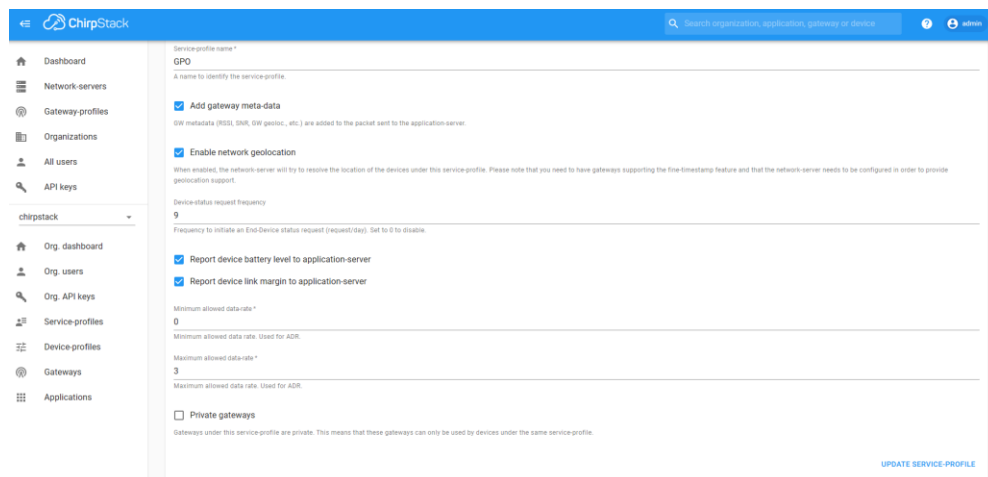


Рисунок 2.22 - Раздел Service-profiles

Создав профиль для подключаемых устройств в разделе «Device-profiles», что представлено на рисунке 2.23, необходимо перейти в раздел «Applications» для подключения устройства передачи данных, а именно коммутационной платы RAK 811, что представлено на рисунке 2.24.

Рисунок 2.23 - Раздел Device-profiles

Рисунок 2.24 - Раздел Service-profiles

После вышеописанных действий на сервере ChirpStack в разделе «Gateways» необходимо создать подключение к базовой станции и можно

будет наблюдать, что сервер видит базовую станцию, что представлено на рисунке 2.25.

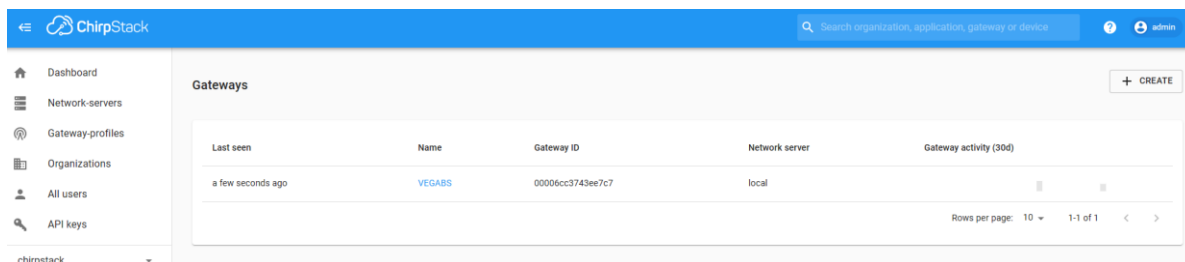


Рисунок 2.25 - Раздел Service-profiles

Для добавления устройства передачи данных, а именно коммутационной платы RAK811 необходимо перейти в раздел «Application» и нажать кнопку «Create». После чего необходимо заполнить поля Device name - имя устройства, Device Eui – EUI идентификатор устройства и выбрать профиль устройства, созданный ранее. Выставленные параметры представлены на рисунке 2.26.

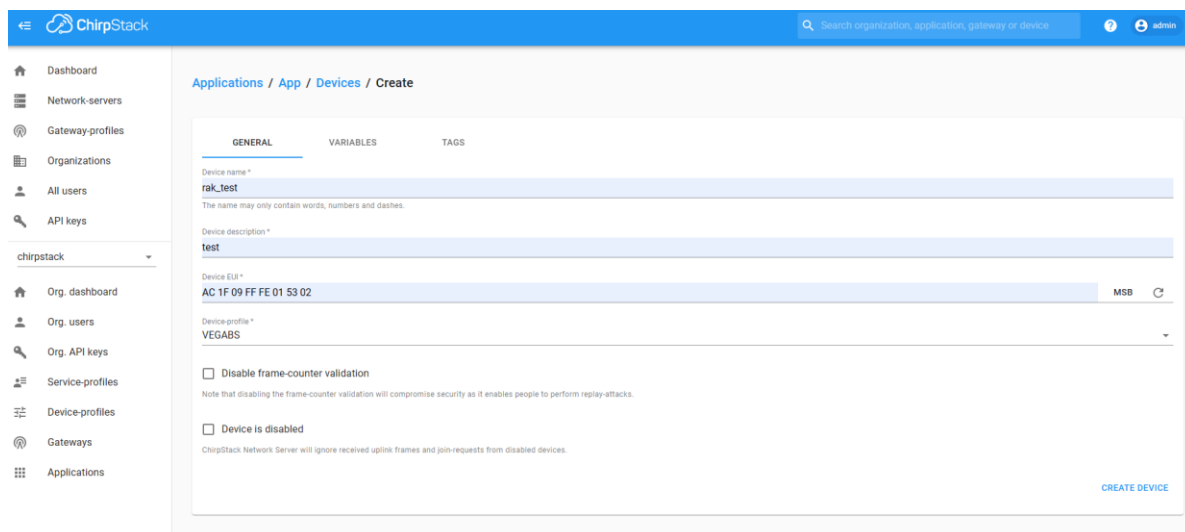


Рисунок 2.26 - Выставленные параметры устройства

После нажатия на кнопку «Create device» откроется окно в котором необходимо ввести Application Key – ключ приложения устройства, что представлено на рисунке 2.27.

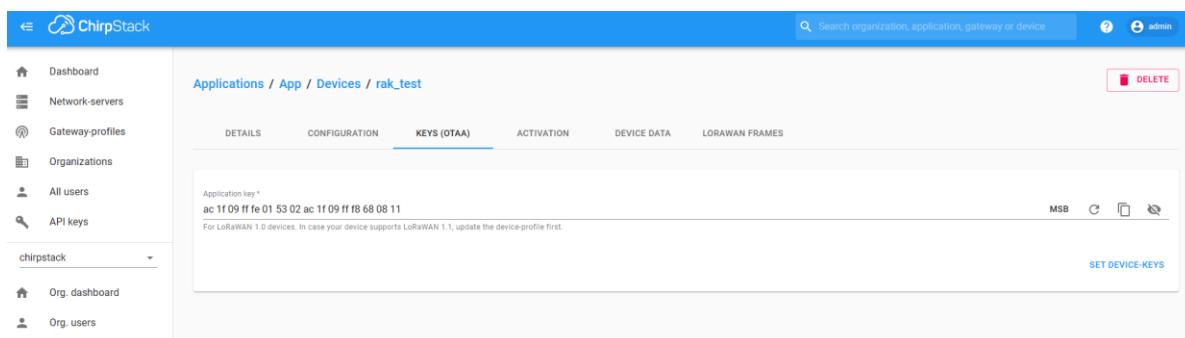


Рисунок 2.27 - Ввод Application Key

Application EUI не требуется для ChirpStack так как не используется join-server, поэтому в значение AppEUI необходимо записать значение DevEUI через официальное приложение «RAK Serial Port Tool», что представлено на рисунке 2.28.

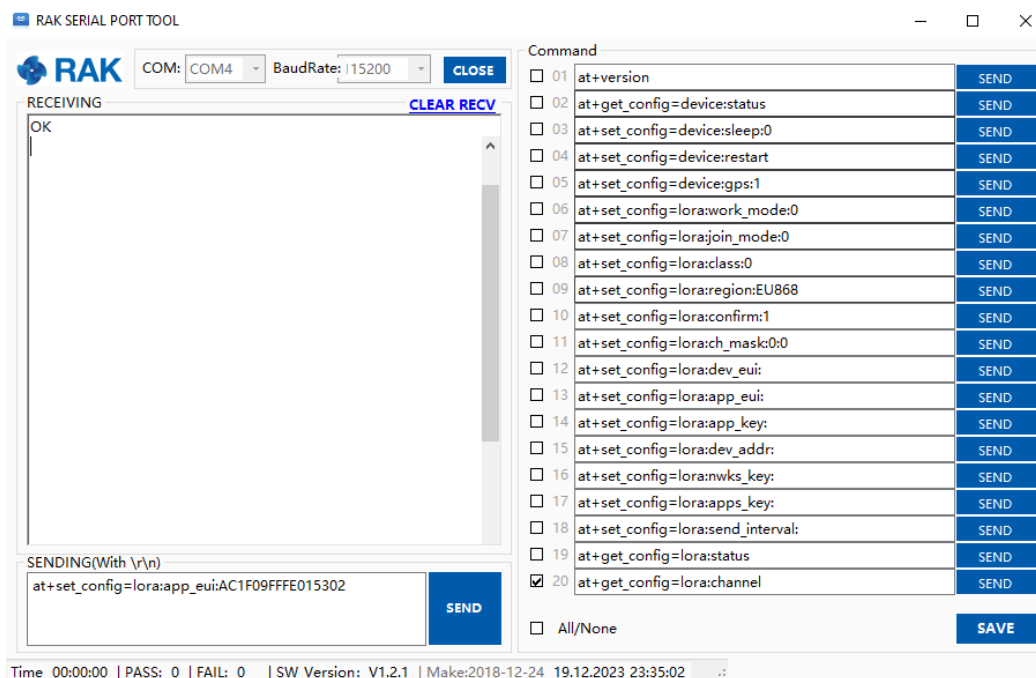


Рисунок 2.28 - Записанное значение AppEUI

Для проверки устройства необходимо через «RAK Serial Port Tool» подключить устройство командой «at+join», после чего как на сервере, так и в приложении появиться сообщение об успешном подключении, что представлено на рисунках 2.29 - 2.30.

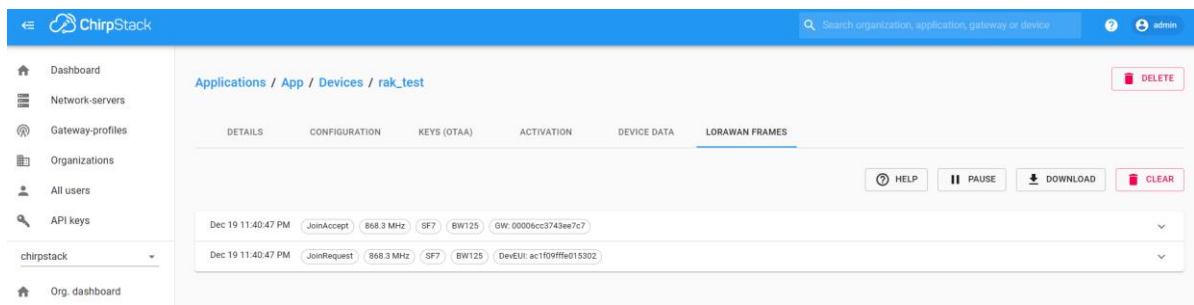


Рисунок 2.29 - Сообщение об успешном подключении на сервере

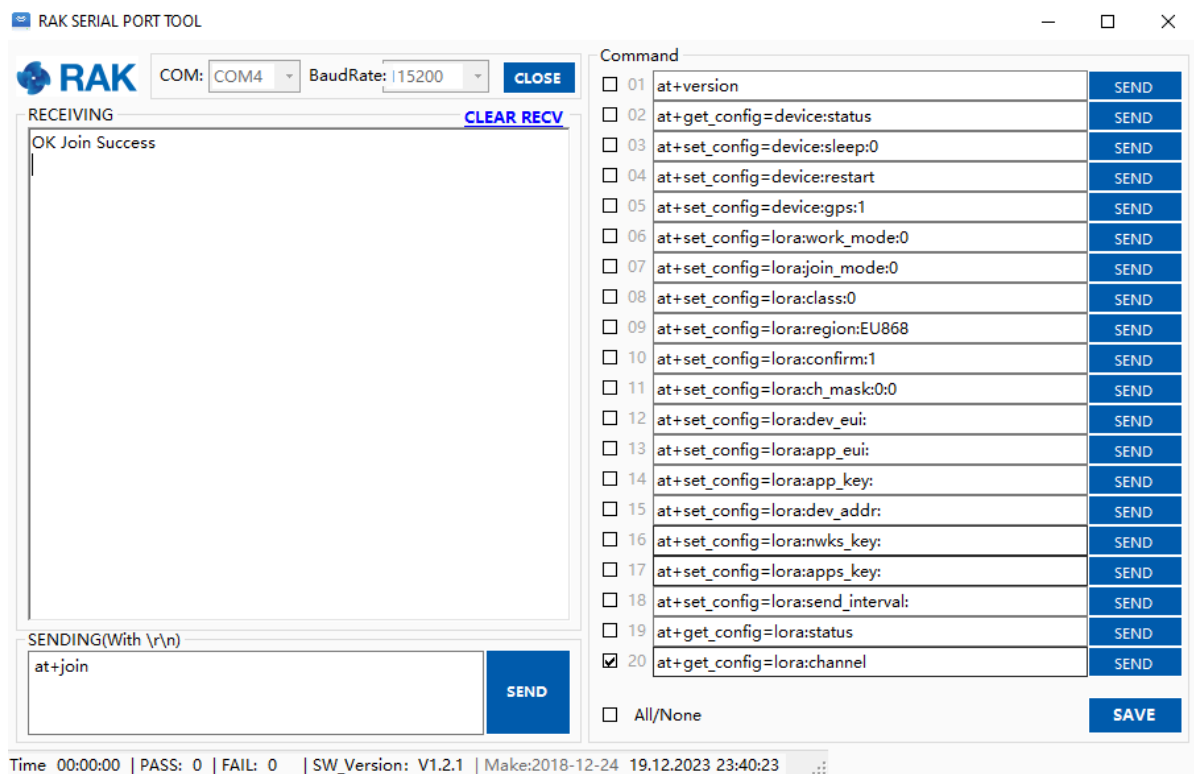


Рисунок 2.30 - Сообщение об успешном подключении в приложении

2.7 Сравнение сервисов веб-хостинга для переноса веб-приложения

Для возможности запуска веб-приложения не только на локальной станции, но и на других машинах необходимо было подобрать подходящий сервис веб-хостинга для переноса веб-приложения.

В данном проекте были рассмотрены три варианта веб-хостинга - SpaceWeb, pythonAnywhere и Heroku.

SpaceWeb предоставляет бесплатные тарифы с ограниченными ресурсами, что может быть подходящим вариантом для небольших проектов. Платформа поддерживает различные языки программирования, включая PHP, Python и Ruby, что обеспечивает гибкость в выборе технологии. Однако, бесплатные тарифы могут ограничивать количество доступных ресурсов, что может повлиять на производительность веб-приложения при росте активности пользователей.

PythonAnywhere специализирован на языке Python и предоставляет простой интерфейс для развертывания и управления веб-приложениями. Бесплатный тариф с ограниченными ресурсами может быть привлекательным выбором для проектов, основанных на Python. Однако, ограничение на ресурсы может оказаться недостаточным для более масштабируемых приложений.

Heroku, платформа, ориентированная на разработчиков, предоставляет бесплатные тарифы с некоторыми значительными преимуществами. Она поддерживает различные языки программирования и обеспечивает автоматическое масштабирование, что делает ее привлекательным вариантом для небольших и средних проектов. Однако, бесплатные тарифы имеют ограничение по времени активности, что может снизить доступность приложения в случае неактивности.

Сравнительный анализ вариантов веб-хостинга представлен в таблице 2.1.

Таблица 2.1 - Сравнение вариантов веб-хостинга

Хостинг	SpaceWeb	pythonAnywhere	Heroku
Критерии			
Поддержка языков программирования	PHP, Python, Ruby и другие.	Python	PHP, Python, Ruby и другие.
Ограничения минимального тарифа	1 ГБ дискового пространства Нагрузка 120 СР Одно веб-приложение в личном домене	1 ГБ дискового пространства Нагрузка 2 000 СР Одно веб-приложение в личном домене	1 ГБ дискового пространства Нагрузка 400 СР Одно веб-приложение в личном домене
Удобства пользования	Панель управления для удобства управления	Простой интерфейс и инструменты для Python-приложений	Автоматическое масштабирование
Поддержка	Через тикеты, электронную почту или чат	Через тикеты, электронную почту или чат	Через сообщество пользователей и документацию

Составив таблицу сравнения вариантов и учитывая, что веб-приложение было написано на языке Python, более подходящим вариантом сервиса для веб-хостинга приложения является pythonAnywhere.

2.8 Получение сертификата для реализации уведомлений

Для получения уведомлений от web-приложения необходимо защищенное подключение к нему по протоколу HTTPS, которое, в свою очередь, возможно при наличии в корневом каталоге приложения подписанного SSL-сертификата.

С помощью установленной криптографической библиотеки OpenSSL был сформирован закрытый ключ, а затем с его помощью и сам сертификат, указав в поле имени домена адрес хоста web-приложения (рисунок 2.31). Затем сформированные файлы сертификата и закрытого ключа были перенесены в корневой каталог приложения, чтобы интегрировать их в код создания веб-ссылки приложения (рисунок 2.32).

```
PS D:\OpenSSL-Win32> openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
Certificate request self-signature ok
subject=C=RU, ST=Tomsk, L=Tomsk, O=TUSUR, OU=GPO1904, CN=127.0.0.1, emailAddress=gpo1904@gpo.com
PS D:\OpenSSL-Win32>
```

Рисунок 2.31 – Процесс создания SSL-сертификата

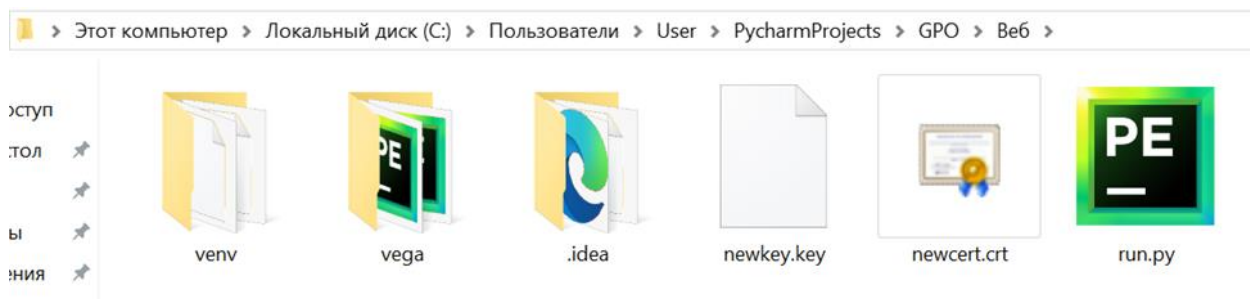
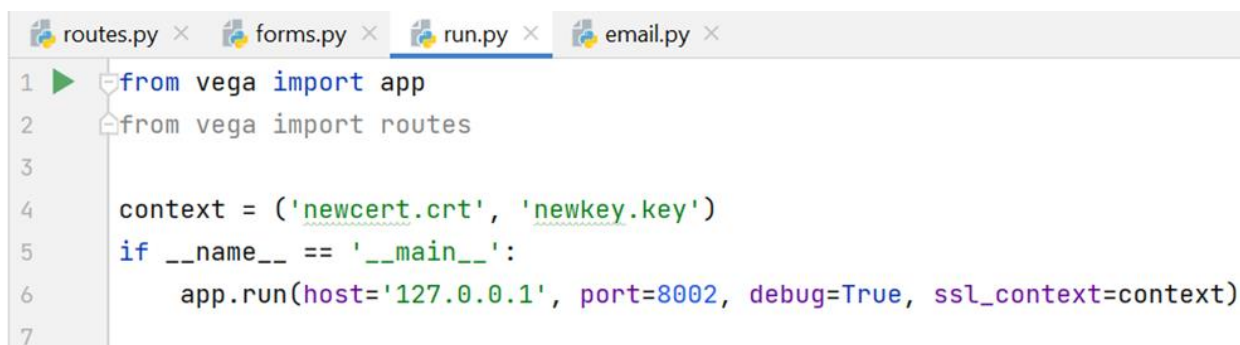


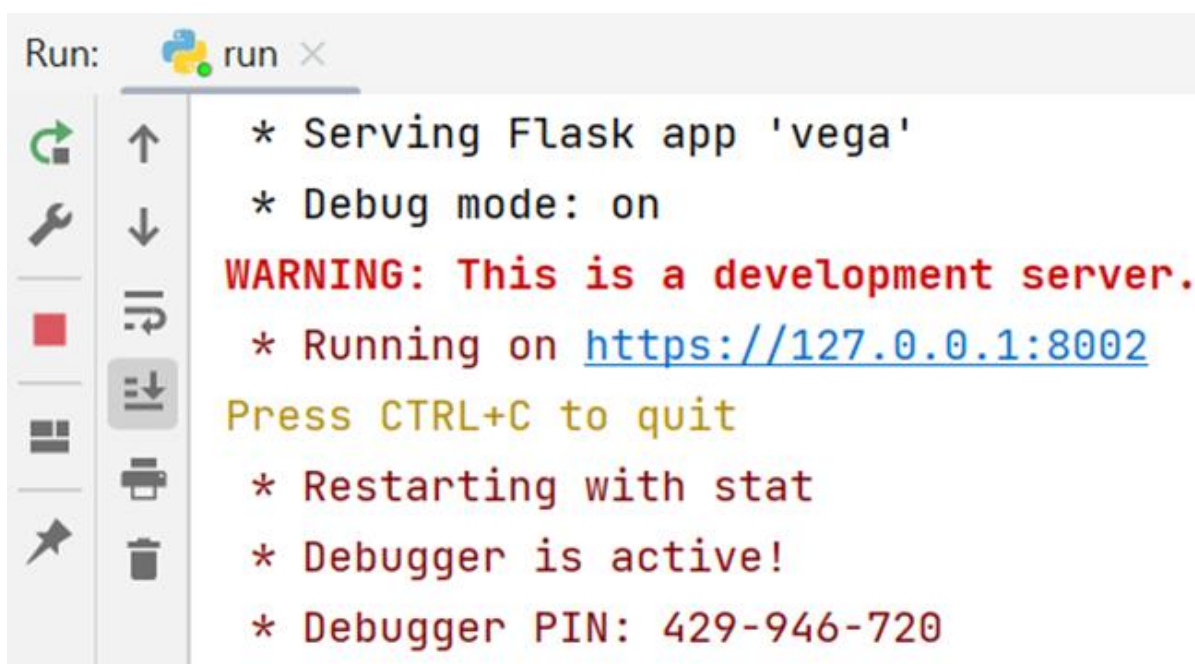
Рисунок 2.32 – Перенос сформированных файлов в корневой каталог приложения

Далее в файле запуска приложения «run.py» была прописана переменная context, подключающая в 6 строке ранее сформированные файлы сертификата и ключа для создания защищенной ссылки (рисунок 2.33 – 2.34).



```
1 from vega import app
2 from vega import routes
3
4 context = ('newcert.crt', 'newkey.key')
5 if __name__ == '__main__':
6     app.run(host='127.0.0.1', port=8002, debug=True, ssl_context=context)
7
```

Рисунок 2.33 – Подключение файлов сертификата и ключа к созданию защищенной ссылки



```
Run: run x
* Serving Flask app 'vega'
* Debug mode: on
WARNING: This is a development server.
* Running on https://127.0.0.1:8002
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 429-946-720
```

Рисунок 2.34 – Созданная ссылка по протоколу HTTPS

Однако, после создания защищенного соединения, браузер отказывается принимать сертификат как доверенный, даже после его добавления в соответствующий раздел доверенных сертификатов в оснастке консоли «Сертификаты» (рисунок 2.35 – 2.36).

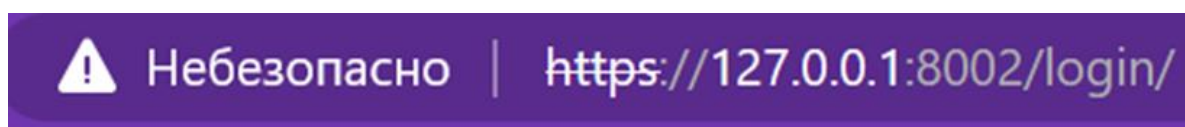


Рисунок 2.35 – Проблема доверия браузера сертификату

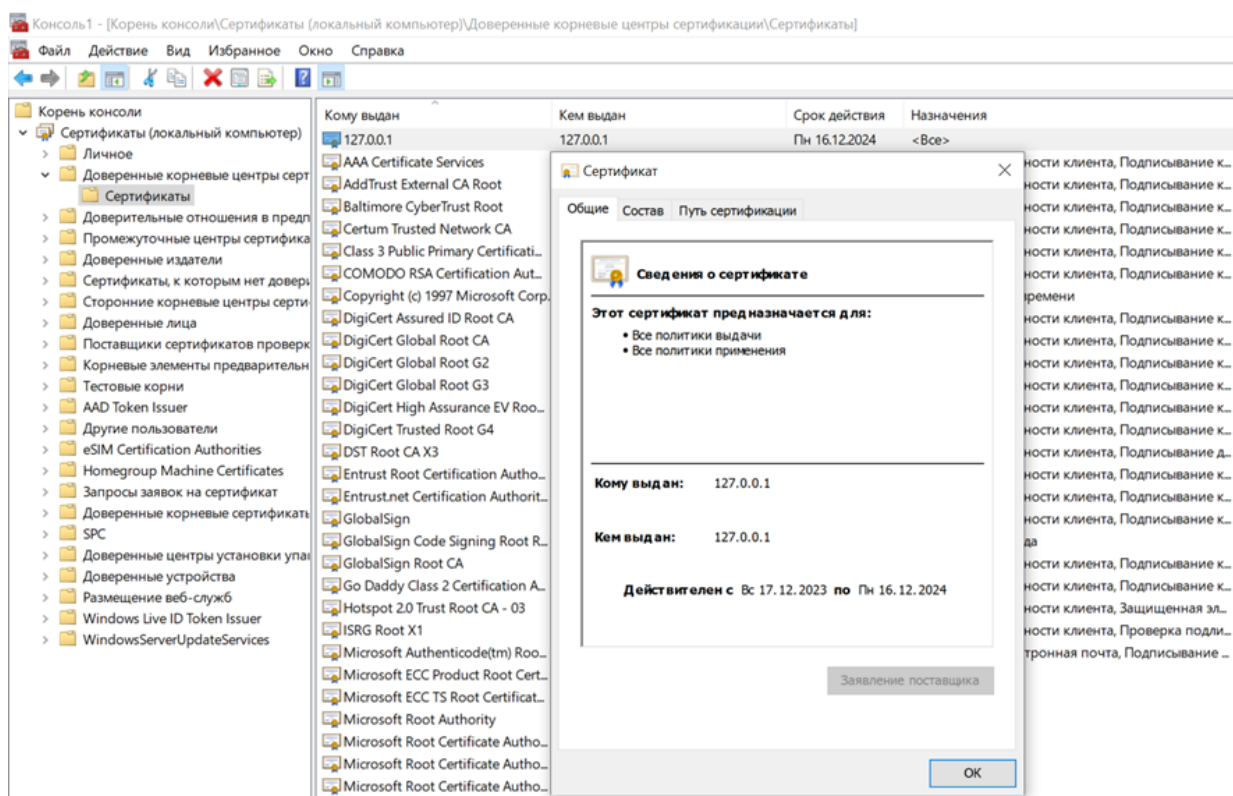


Рисунок 2.36 – Добавление сертификата в раздел доверенных

В результате выполнения поставленной задачи был сформирован самоподписанный SSL-сертификат, файл которого был интегрирован в код приложения, позволивший создавать защищенную веб-ссылку приложения через протокол HTTPS, но возникшая проблема доверия браузеру сертификату подразумевает доработку данного пункта на дальнейших стадиях разработки web-приложения.

2.9 Реализация сброса пароля пользователя

Для реализации сброса пароля пользователя был выбран метод сброса пароля с помощью URL-ссылки, основанный на генерации ссылки на форму сброса пароля, которая придет на указанную пользователем почту в сопровождении токена сброса пароля. Этот токен должен быть проверен, прежде чем разрешить изменение пароля, в качестве

доказательства того, что пользователь, который запросил электронное письмо со ссылкой, имеет доступ к адресу электронной почты в учетной записи.

Сначала была создана кнопка «Забыл пароль» на странице входа, которая при нажатии перенаправляет пользователя на форму, предлагающую ввести адрес электронной почты и отправить на него письмо со ссылкой на форму сброса пароля (рисунок 2.37 – 2.38).

Код измененной страницы входа, созданной формы с полем для электронной почты и формы сброса пароля представлены в Приложениях В–Д.

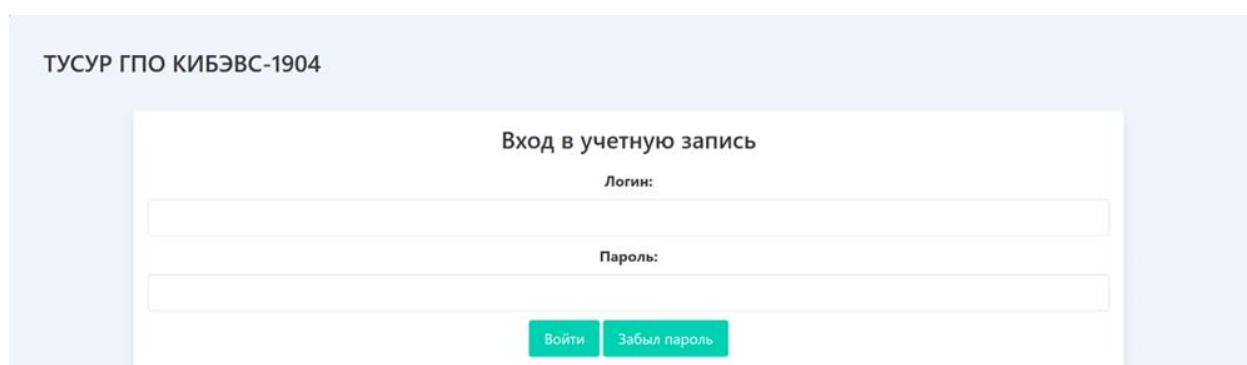


Рисунок 2.37 – Кнопка «Забыл пароль»

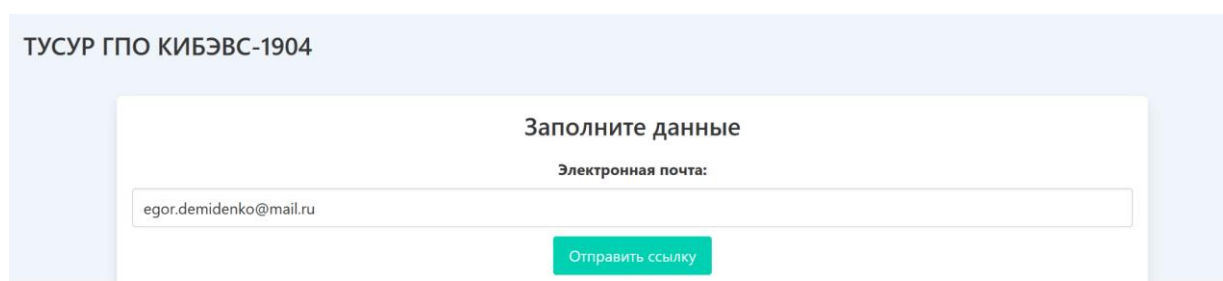
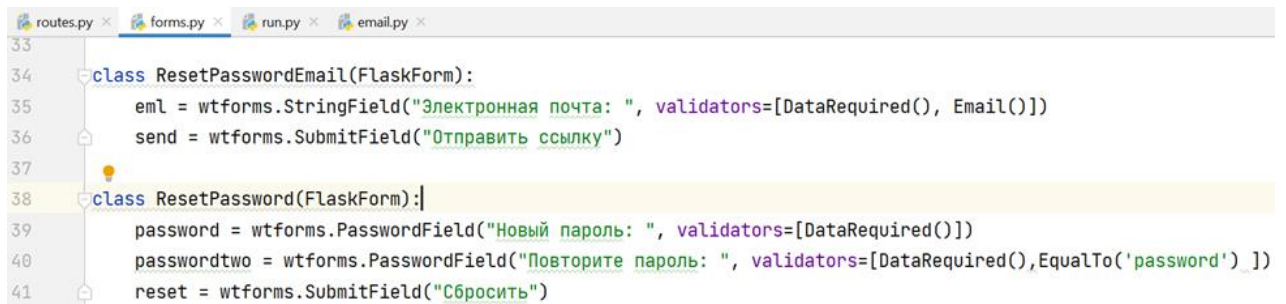


Рисунок 2.38 – Форма ввода адреса электронной почты

Для возможности реализации полей и кнопок новых форм были добавлены соответствующие классы в файл «forms.py» (рисунок 2.39).



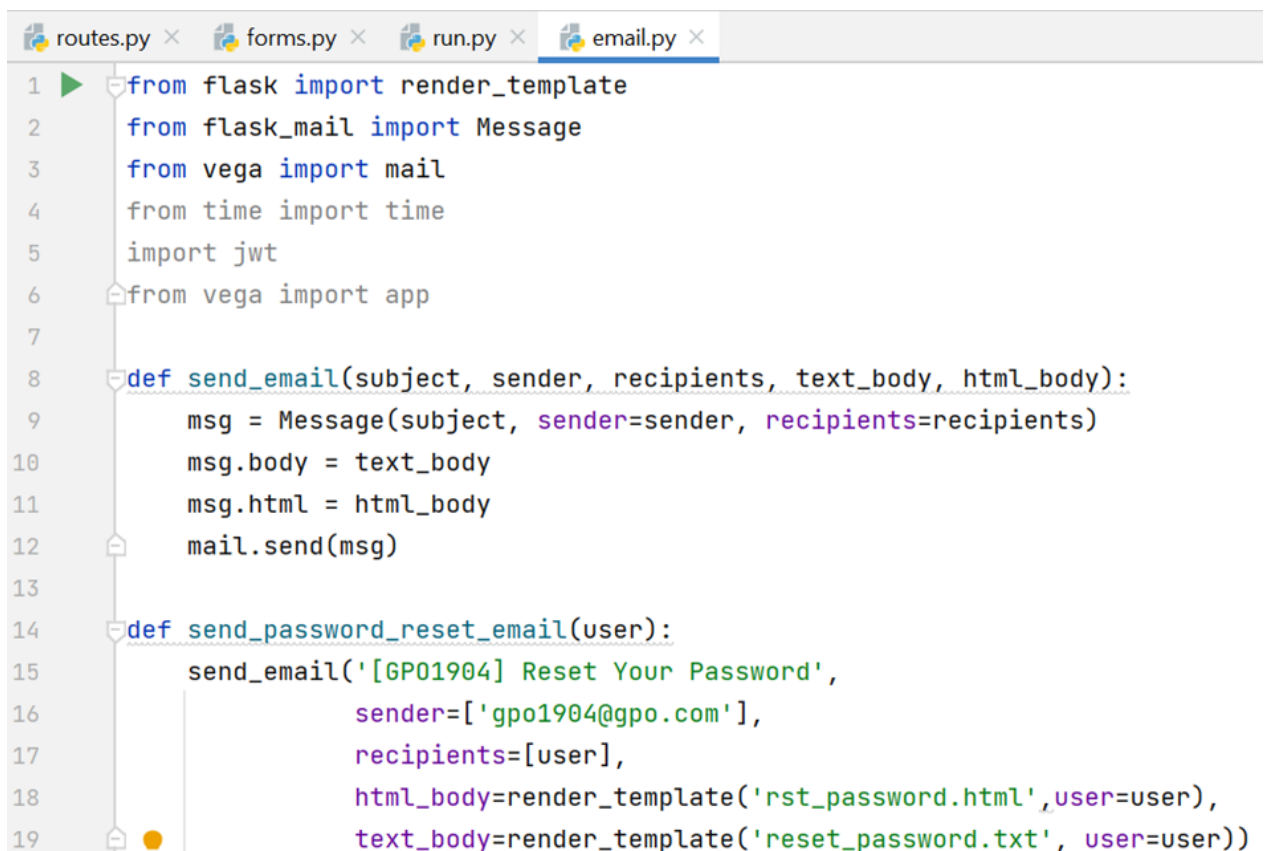
```

33
34 class ResetPasswordEmail(FlaskForm):
35     eml = wtforms.StringField("Электронная почта: ", validators=[DataRequired(), Email()])
36     send = wtforms.SubmitField("Отправить ссылку")
37
38 class ResetPassword(FlaskForm):
39     password = wtforms.PasswordField("Новый пароль: ", validators=[DataRequired()])
40     passwordtwo = wtforms.PasswordField("Повторите пароль: ", validators=[DataRequired(), EqualTo('password')])
41     reset = wtforms.SubmitField("Сбросить")

```

Рисунок 2.39 – Добавленные классы в файле «forms.py»

Также был создан отдельный файл «email.py», содержащий в себе методы генерации письма и его отправки на указанную пользователем электронную почту (рисунок 2.40). Код данного файла представлен в Приложении Е.



```

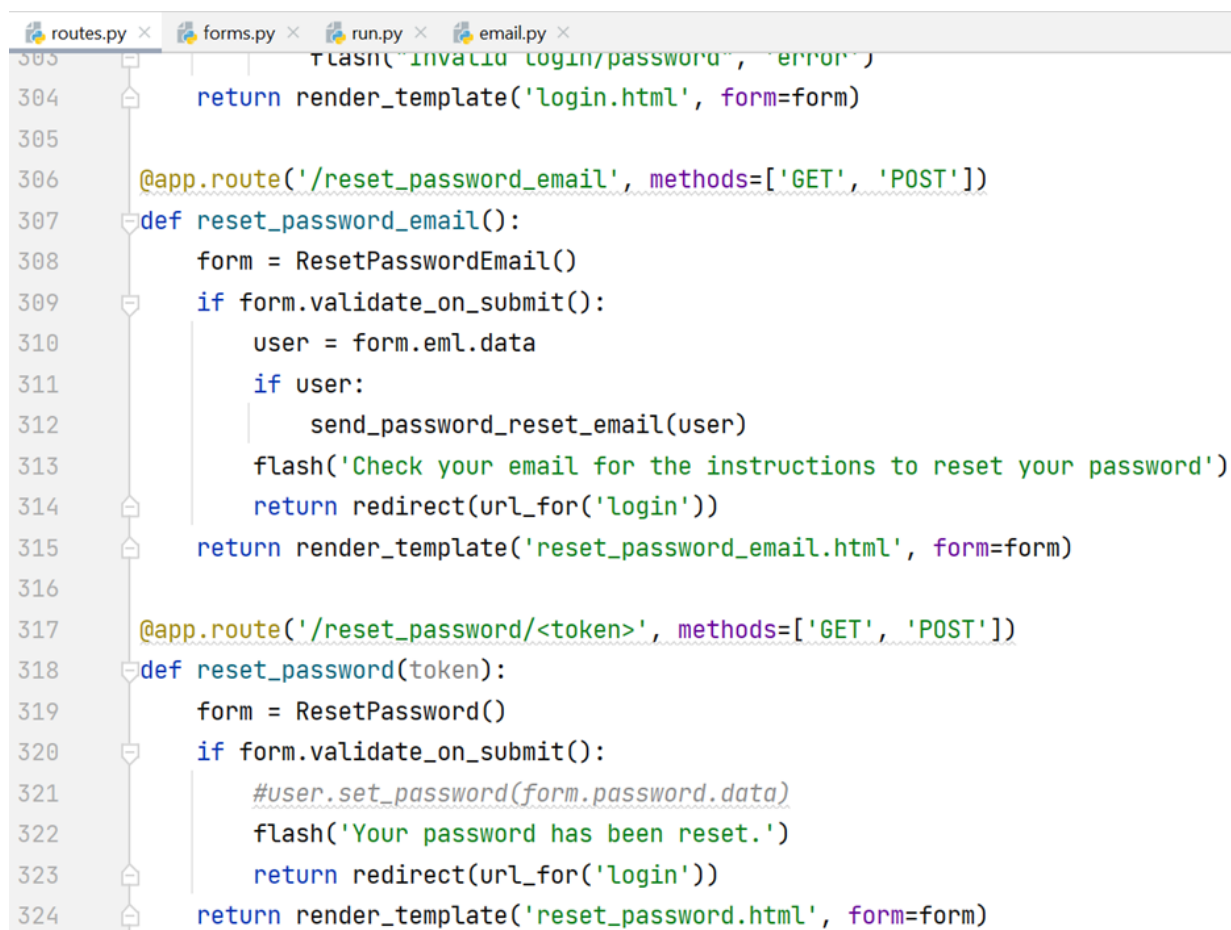
1 from flask import render_template
2 from flask_mail import Message
3 from vega import mail
4 from time import time
5 import jwt
6 from vega import app
7
8 def send_email(subject, sender, recipients, text_body, html_body):
9     msg = Message(subject, sender=sender, recipients=recipients)
10    msg.body = text_body
11    msg.html = html_body
12    mail.send(msg)
13
14 def send_password_reset_email(user):
15     send_email('[GP01904] Reset Your Password',
16               sender=['gpo1904@gpo.com'],
17               recipients=[user],
18               html_body=render_template('rst_password.html', user=user),
19               text_body=render_template('reset_password.txt', user=user))

```

Рисунок 2.40 – Файл «email.py»

Также был редактирован файл «routes.py», в который были добавлены методы реализации созданных форм ввода электронной почты

и сброса пароля. Код данного файла представлен в Приложении Ж. (рисунок 2.41).



```
303     flash('Invalid login/password', 'error')
304     return render_template('login.html', form=form)
305
306     @app.route('/reset_password_email', methods=['GET', 'POST'])
307     def reset_password_email():
308         form = ResetPasswordEmail()
309         if form.validate_on_submit():
310             user = form.eml.data
311             if user:
312                 send_password_reset_email(user)
313                 flash('Check your email for the instructions to reset your password')
314                 return redirect(url_for('login'))
315         return render_template('reset_password_email.html', form=form)
316
317     @app.route('/reset_password/<token>', methods=['GET', 'POST'])
318     def reset_password(token):
319         form = ResetPassword()
320         if form.validate_on_submit():
321             #user.set_password(form.password.data)
322             flash('Your password has been reset.')
323             return redirect(url_for('login'))
324         return render_template('reset_password.html', form=form)
```

Рисунок 2.41 – Добавленные методы в файле «routes.py»

Для возможности отправки письма на указанный пользователем адрес электронной почты также был создан новый почтовый сервер, адрес которого указан в методе «send_password_reset_email» файла «email.py» для его корректной работы (рисунок 2.42).

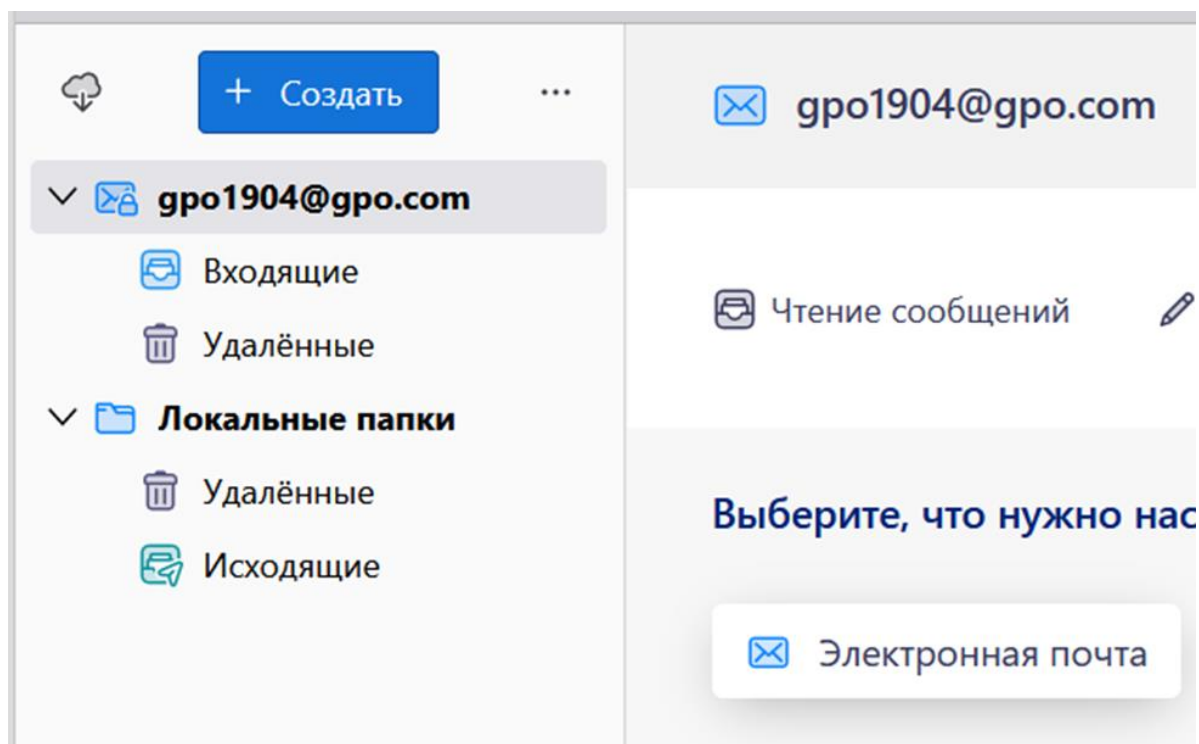


Рисунок 2.42 – Созданный почтовый сервер

При включении токена в процесс создания ссылки на форму сброса пароля в файле «email.py» произошел конфликт установленных пакетов Flask, требуемых для исправного функционирования web-приложения в целом, поэтому часть с генерацией ссылки, сопровождаемой токеном, реализовать не удалось.

В результате выполнения поставленной задачи были созданы необходимые файлы формата html, а также методы для их обработки в корневых файлах приложения, был создан отдельный файл «email.py» для реализации генерации электронного письма и отдельный почтовый сервер для наличия адреса-отправителя, необходимого для корректной работы файла «email.py», но не была реализована генерация токена, необходимого для создания ссылки на форму сброса пароля в отправленном письме, что планируется исправить в последующих стадиях разработки приложения.

2.10 Тестирование работы графика с новыми данными

В ходе работы над проектом была проблема с некорректным отображением температуры, из-за чего приходили слишком большие значения температур. График с некорректными данными с платы представлен на рисунке 2.43.



Рисунок 2.43 – График с большими значениями температур

Для того, чтобы перейти на страницу с графиком необходимо нажать на идентификатор устройства в списке устройств.

Под графиком на данной странице представлен список полученных пакетов. После решения проблем с отправкой ложных значений(путем доработки прошивки) было собрано устройство и были получены новые данные представленные на рисунках 2.44-2.45. Полученные значения температуры представлены в столбце data, также можно посмотреть частоту(freq), порт и дату получения пакетов(ts).

ack	data	dr	fcnt	freq	gatewayld
0	27	SF7 BW125 4/5	252	868500000	00006CC3743EE7C7
0	28	SF7 BW125 4/5	251	867300000	00006CC3743EE7C7
0	28	SF7 BW125 4/5	250	867500000	00006CC3743EE7C7
0	28	SF7 BW125 4/5	249	868100000	00006CC3743EE7C7
0	28	SF7 BW125 4/5	248	868500000	00006CC3743EE7C7
0	29	SF7 BW125 4/5	247	867300000	00006CC3743EE7C7
0	30	SF7 BW125 4/5	246	867500000	00006CC3743EE7C7
0	31	SF7 BW125 4/5	245	868100000	00006CC3743EE7C7
0	26	SF7 BW125 4/5	244	868500000	00006CC3743EE7C7
0	26	SF7 BW125 4/5	243	867100000	00006CC3743EE7C7
0	26	SF7 BW125 4/5	242	867700000	00006CC3743EE7C7
0	26	SF7 BW125 4/5	241	868100000	00006CC3743EE7C7

Рисунок 2.44 – Полученные данные с устройства

port	rssi	snr	ts	type
5	-75	9.8	2023-12-16 19:14:43.074000+00:00	CONF_UP
5	-71	9	2023-12-16 19:14:33.054000+00:00	CONF_UP
5	-77	6.5	2023-12-16 19:14:23.071000+00:00	CONF_UP
5	-72	9.2	2023-12-16 19:14:13.035000+00:00	CONF_UP
5	-75	9.5	2023-12-16 19:14:03.053000+00:00	CONF_UP
5	-67	8.8	2023-12-16 19:13:53.077000+00:00	CONF_UP
5	-69	7	2023-12-16 19:13:43.045000+00:00	CONF_UP
5	-71	9	2023-12-16 19:13:33.076000+00:00	CONF_UP
5	-75	8.5	2023-12-16 19:13:23.038000+00:00	CONF_UP
5	-70	8.5	2023-12-16 19:13:13.061000+00:00	CONF_UP
5	-71	10	2023-12-16 19:13:03.080000+00:00	CONF_UP
5	-69	9.2	2023-12-16 19:12:53.048000+00:00	CONF_UP

Рисунок 2.45 – Полученные данные с устройства

В результате по этим значениям был построен график представленный на рисунке 2.46. Из него видно, что значения находятся в пределах нормы.

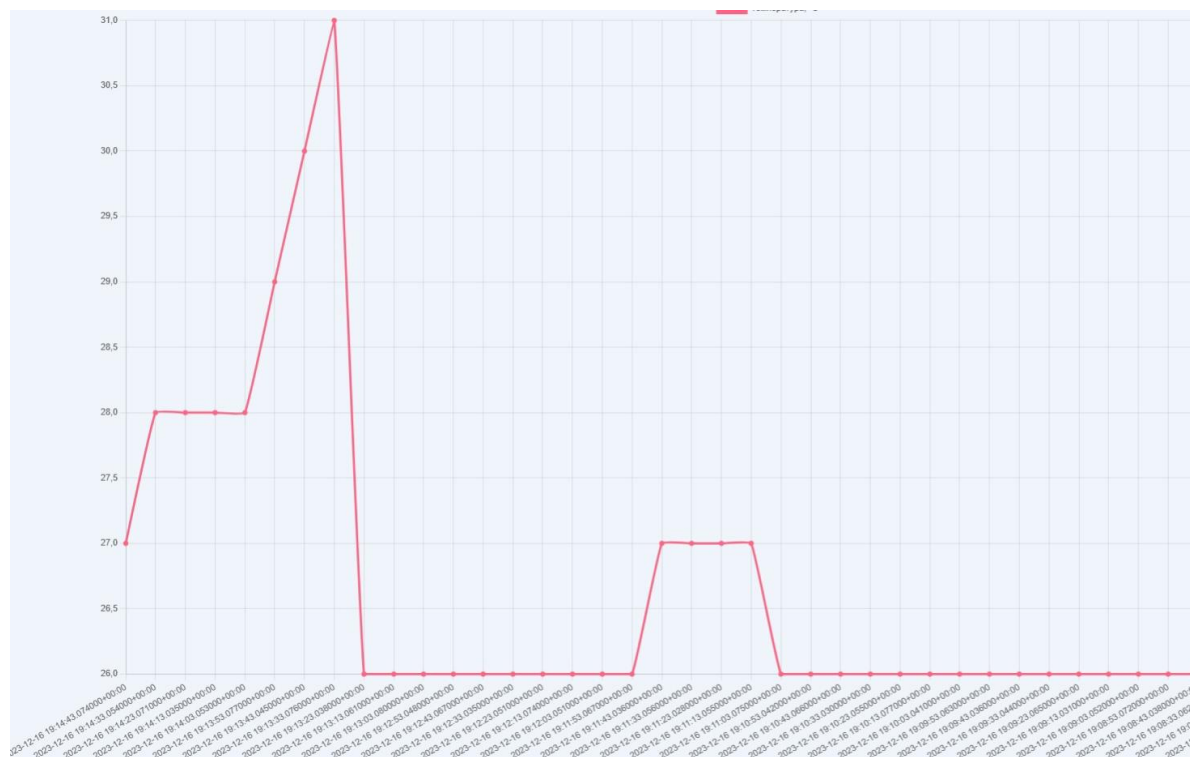


Рисунок 2.46 – График по полученным данным

2.11 Освоение моделирования в приложении Blender

Для реализации всех задуманных изменений в корпусе конечного устройства было принято решение перейти с программы моделирования Tinkercad на Blender.

Blender - профессиональное свободное и открытое программное обеспечение для создания трехмерной компьютерной графики. Преимуществом Blender является простота в управлении и навигации, позволяющие быстро освоиться в основном функционале программы. Данное приложение обладает широким набором инструментов и функций для моделирования, включая создание и редактирование трехмерных объектов.

Для освоения функционала приложения, дальнейшей простоты перехода с Tinkercad и успешного изменения корпуса устройства было построено несколько 3d моделей (рисунки 2.47 - 2.49).



Рисунок 2.47 - Модель снеговика

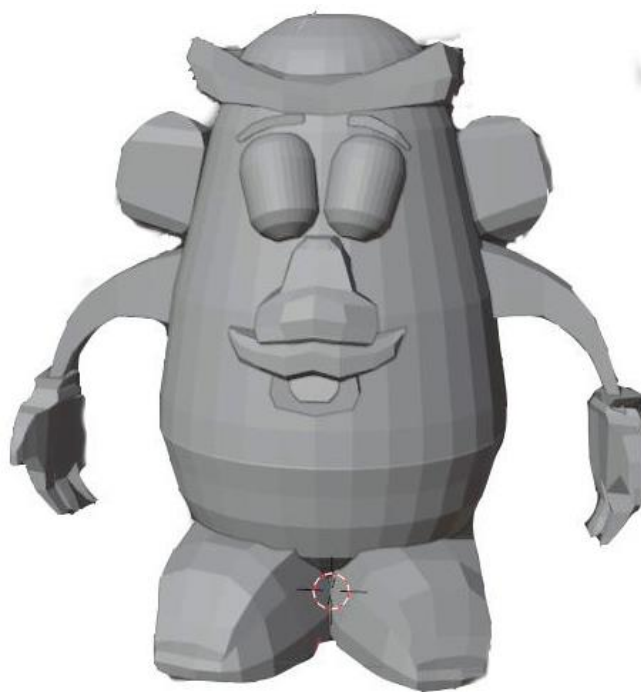


Рисунок 2.48 - Модель Мистер Картофельная голова

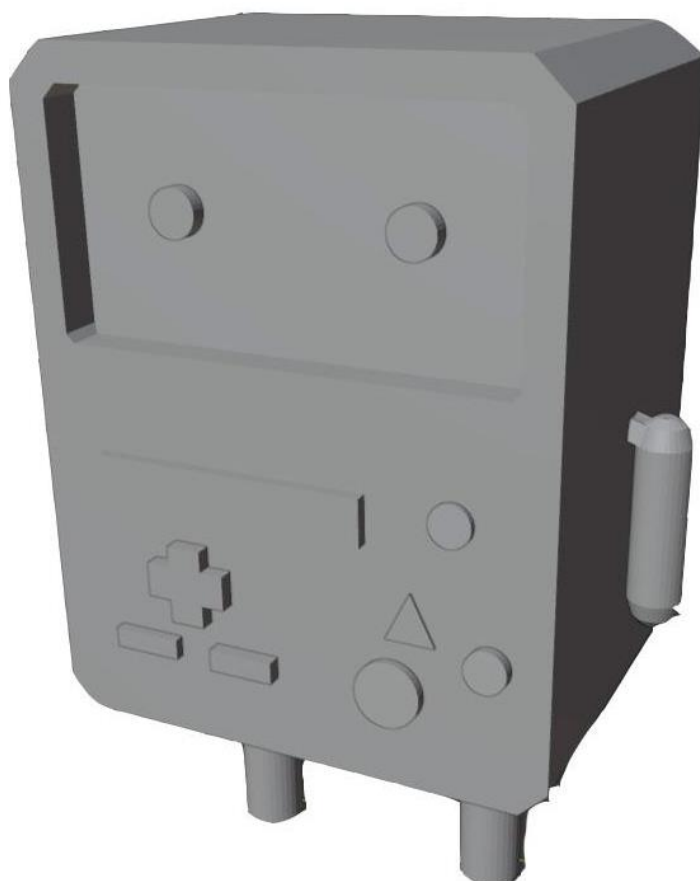


Рисунок 2.49 - Модель ВМО

2.12 Доработка конечного корпуса устройства

Перед тем, как перейти к редактированию корпуса устройства после получения навыков 3d моделирования в приложении Blender, был разработан эскиз всех вносимых поправок (рисунок 2.50). Во время создания эскиза во внимание были приняты ошибки предыдущего корпуса. Фотография предыдущего корпуса представлена на рисунке 2.51.

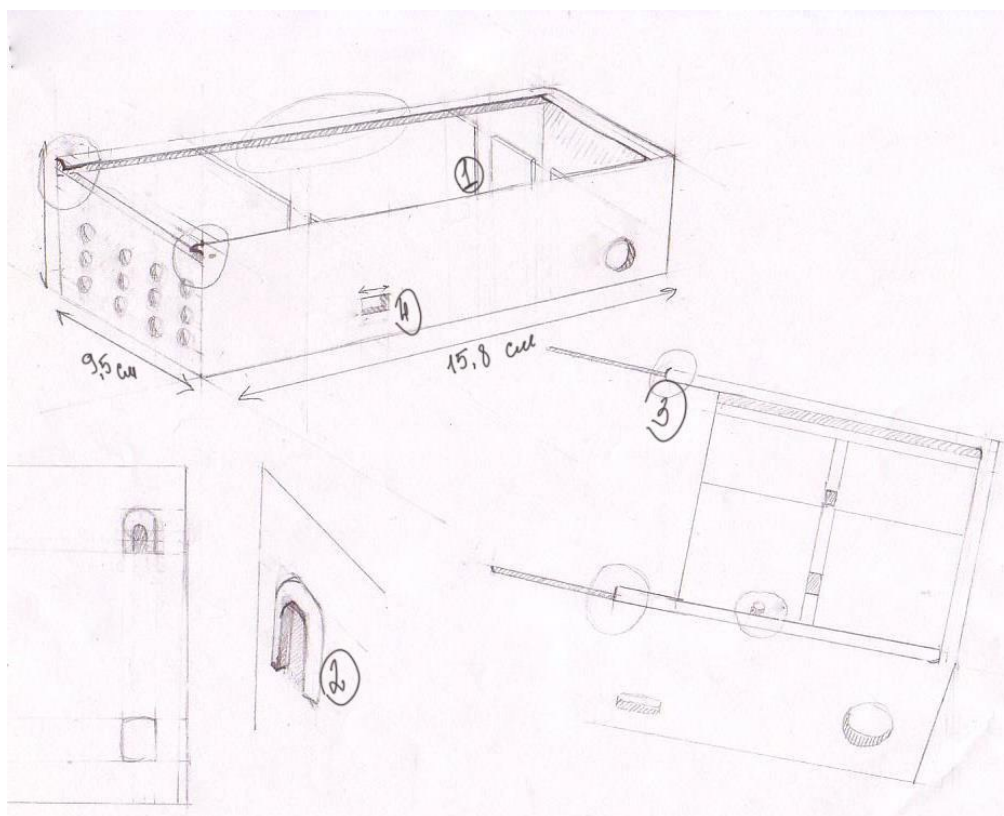


Рисунок 2.50 - Эскиз вносимых изменений

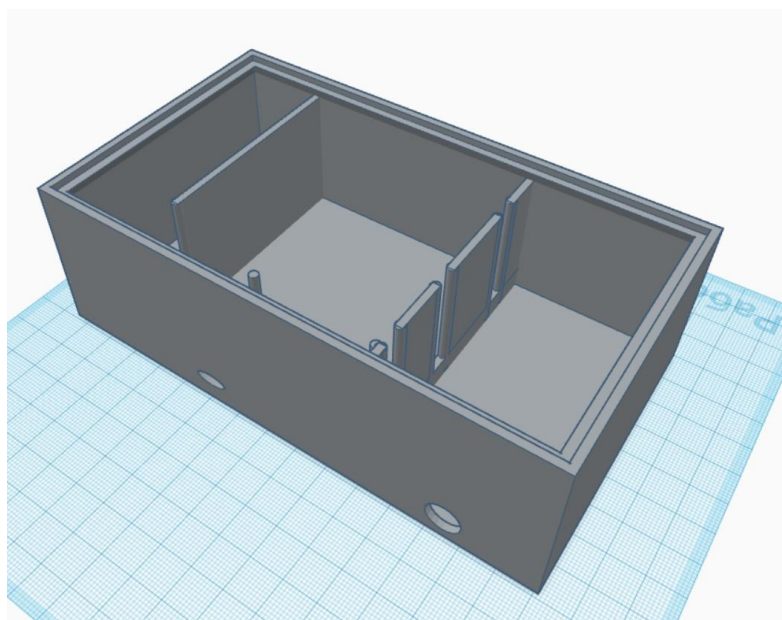


Рисунок 2.51 - Предыдущий контейнер

Так как некоторые изменения понесли бы за собой проблему в дальнейшей печати корпуса, некоторые элементы были изменены (таблица 2.2).

Таблица 2.2 - Внесенные изменения в эскиз

Номер элемента на эскизе	Внесенные изменения в эскиз
1	Отверстия для проводов были скруглены для упрощения при печати
2	Сделать отверстия для крепления контейнера не как дополнительные ножки, а отверстием
3	Оставить крышку как на оригинальной модели контейнера, так как она там надежнее
4	Разъем для провода было закруглено для упрощения печати

3d модель доработанного контейнера представлена на рисунке 2.52.
Фотография напечатанного контейнера представлена на рисунке 2.53.

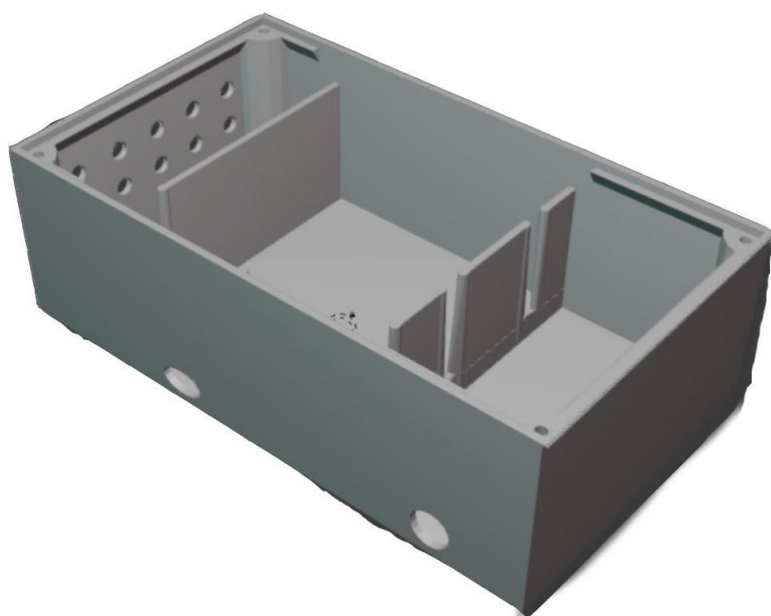


Рисунок 2.52 - 3d модель контейнера



Рисунок 2.53 - Напечатанный контейнер

Заключение

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

- в коде была исправлена ошибка отправки ложных данных на сервер;
- была произведена установка и настройка серверного обеспечения базовой станции;
 - были изучены API-функции ChirpStack;
 - было реализовано тестирование получения данных с платы и их корректное отображение на графике в веб-приложении;
 - был создан SSL-сертификат и интегрирован в код веб-приложения;
 - были созданы и описаны на языке программирования Python формы обработки данных для сброса пароля пользователя;
- были получены навыки 3d моделирования в приложении Blender;
- был доработан конечный корпус для 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 31.05.2023).
2. Mbed OS [электронный ресурс] – Режим доступа: <https://os.mbed.com/mbed-os/> (дата обращения 16.02.2023).
3. Руководство для IOT Vega Server рев23 [Электронный ресурс] – Режим доступа: <https://iotvega.com/soft/server> (дата обращения 24.03.2023).
4. Руководство для Vega БС-2.2 рев30 [Электронный ресурс] – Режим доступа: <https://iotvega.com/product/bs02-2> (дата обращения 16.02.2023).
5. API документация ChirpStack – Режим доступа: <https://www.chirpstack.io/docs/chirpstack/api/python-examples.html> (дата обращения 07.12.2023).
6. Документация для фреймворка GRPC – Режим доступа: <https://grpc.github.io/grpc/python/index.html> (дата обращения 07.12.2023).
7. Мега-Учебник Flask, Часть X: Поддержка электронной почты – Режим доступа: <https://habr.com/ru/articles/348566/> (дата обращения 20.09.2023).
8. Документация библиотеки websocket [Электронный ресурс] – Режим доступа: <https://websockets.readthedocs.io/en/9.0.1/pdf/> (дата обращения 30.10.2023).
9. Руководство по разворачиванию и настройки сети LoRaWAN [Электронный ресурс] - Режим доступа: <https://iotvega.com/product/bs02-2> (дата обращения 16.02.2023).
10. ChartJS – JavaScript-библиотека визуализации данных [Электронный ресурс]. – Режим доступа:

<https://habr.com/ru/companies/developersoft/articles/185210/> (дата обращения 14.04.2023).

11. Официальный сайт Tinkercad [Электронный ресурс] – Режим доступа: <https://www.tinkercad.com/> (дата обращения 07.04.2023).

12. Разница между strlen() и sizeof() [Электронный ресурс] – Режим доступа: <https://russianblogs.com/article/3022145414/> (дата обращения 13.04.2023).

13. Типы полей моделей [Электронный ресурс] – Режим доступа: <https://metanit.com/python/django/5.2.php> (дата обращения 27.04.2023).

14. Репозиторий GitHub [Электронный ресурс] – Режим доступа: <https://github.com/tsr-bloody/GPO-storage>

15. Уроки 3d моделирования в Blender [Электронный ресурс] – Режим доступа: <https://videoinfographica.com/blender-tutorials/> (дата обращения 14.09.23).

16. Пример использования bufferedSerial [Электронный ресурс] – Режим доступа: https://www.astro.gla.ac.uk/users/norman/distrib/qp/doc/class_buffered_serial.html (дата обращения 23.11.23).

17. Пример использования bufferedSerial [Электронный ресурс] – Режим доступа: <https://registry.platformio.org/libraries/mbed-sam-grove/BufferedSerial/examples/df-2014-heroku-thermostat-k64f> (дата обращения 23.11.23).

18. Функция форматного вывода printf() [Электронный ресурс] – Режим доступа: <https://count-zero.ru/2015/printf/> (дата обращения 23.11.23).

19. Руководство по развертыванию серверного обеспечения ChirpStack [Электронный ресурс] – Режим доступа: <https://www.chirpstack.io/docs/index.html> (дата обращения 07.12.2023).

Приложение А

(обязательное)

Код прошивки тестирования датчика

```
#include "mbed.h"
#include "BME280.h"

BME280 sensor_bme(I2C_SDA, I2C_SCL);

int temperature;

int main()
{
    sensor_bme.initialize();
    while(1)
    {
        printf("-----\n");
        printf("temp %i\n", sensor_bme.getTemperature());
        printf("-----\n");
        printf("-----\n");
        printf("\n");

        ThisThread::sleep_for(10s);
    }
}
```

Приложение Б

(обязательное)

Исправленная прошивка

```
#include "mbed.h"
#include "BME280.h"
#include <stdio>

// Объект для работы с UART
BufferedSerial dev (D8, D2);
BME280 bme (I2C_SDA, I2C_SCL);

int main()
{
    dev.set_baud(115200); // Установка скорости передачи данных (бод)
    bme.initialize();
    char buffer[32];

    int len;
    len = snprintf(buffer, sizeof(buffer), "at+join\r\n");
    dev.write(buffer, len);

    ThisThread::sleep_for(5s);

    while (1)
    {
        int data = bme.getTemperature();
        char data2 = data;

        // Отправка данных через UART
        len = snprintf(buffer, sizeof(buffer), "at+send=lora:5:%d\r\n", data2);
        dev.write(buffer, len);

        // Задержка перед следующей отправкой данных
        ThisThread::sleep_for(10s);
    }
}
```

Приложение В

(обязательное)

Файл login.html

```
{% extends 'base.html' % }
{% block title % }
<title>Sign-Up/Login Form</title>
{% endblock title % }
{% block links % }
{% endblock links % }
{% block content % }
<div class="container is-widescreen">
  <div class="form">
    <div class="box">
      <div id="login" align="center">
        {% set flash_msg = get_flashed_messages(with_categories=true) % }
        {% if flash_msg % }
        {% for category, message in flash_msg % }
        <span class="{{ category }}">{{ message }}</span>
        {% endfor % }
        {% else % }
        <h1 class="title">Вход в учетную запись</h1>
        {% endif % }

        <form method="post">
          {{ form.hidden_tag() }}
          <div class="field">
            {{ form.login.label(class="label is-medium") }}
            <div class="control">
              {{ form.login(class="input is-medium") }}
            </div>
          </div>

          <div class="field">
            {{ form.password.label(class="label is-medium") }}
            <div class="control">
              {{ form.password(class="input is-medium") }}
            </div>
          </div>
          {{ form.submit(class="button is-medium is-primary") }}
          <button id="Забыл пароль" type="button"
onclick="window.location.href = '{{ url_for('reset_password_email') }}'; "
```

```
        class="button is-medium is-primary">Забыл пароль</button>
    </form>
</div>
</div>
</div>
</div>
{% endblock content %}
{% block scripts %}
{% endblock scripts %}
```

Приложение Г

(обязательное)

Файл reset_password_email.html

```
{% extends 'base.html' % }
{% block title % }
<title>RST/ResetPasswordEmail</title>
{% endblock title % }
{% block links % }
{% endblock links % }
{% block content % }
<div class="container is-widescreen">
  <div class="form">
    <div class="box">
      <div id="reset_password_email" align="center">
        {% set flash_msg = get_flashed_messages(with_categories=true) % }
        {% if flash_msg % }
          {% for category, message in flash_msg % }
            <span class="{{ category }}">{{ message }}</span>
          {% endfor % }
        {% else % }
          <h1 class="title">Заполните данные</h1>
        {% endif % }
        <form method="post">
          {{ form.hidden_tag() }}
          <div class="field">
            {{ form.eml.label(class="label is-medium") }}
            <div class="control">
              {{ form.eml(class="input is-medium") }}
            </div>
          </div>
          {{ form.send(class="button is-medium is-primary") }}
        </form>
      </div>
    </div>
  </div>
{% endblock content % }
```

```
{% block scripts %}  
{% endblock scripts %}
```


Приложение Д

(обязательное)

Файл reset_password.html

```
{% extends 'base.html' % }
{% block title % }
<title>RST/ResetPassword</title>
{% endblock title % }
{% block links % }
{% endblock links % }
{% block content % }
<div class="container is-widescreen">
  <div class="form">
    <div class="box">
      <div id="login" align="center">
        {% set flash_msg = get_flashed_messages(with_categories=true) % }
        {% if flash_msg % }
        {% for category, message in flash_msg % }
        <span class="{{ category }}">{{ message }}</span>
        {% endfor % }
        {% else % }
        <h1 class="title">Новый пароль</h1>
        {% endif % }
        <form method="post">
          {{ form.hidden_tag() }}
          <div class="field">
            {{ form.password.label(class="label is-medium") }}
            <div class="control">
              {{ form.password(class="input is-medium") }}
            </div>
```

```

</div>
<div class="field">
  {{ form.passwordtwo.label(class="label is-medium") }}
  <div class="control">
    {{ form.passwordtwo(class="input is-medium") }}
  </div>
</div>
{{ form.reset(class="button is-medium is-primary") }}
</form>
</div>
</div>
</div>
</div>
{% endblock content %}
{% block scripts %}
{% endblock scripts %}

```

Приложение Е

(обязательное)

Файл email.py

```
from flask import render_template
from flask_mail import Message
from vega import mail
from time import time
import jwt
from vega import app

def send_email(subject, sender, recipients, text_body, html_body):
    msg = Message(subject, sender=sender, recipients=recipients)
    msg.body = text_body
    msg.html = html_body
    mail.send(msg)

def send_password_reset_email(user):
    send_email('[GPO1904] Reset Your Password',
               sender=['gpo1904@gpo.com'],
               recipients=[user],
               html_body=render_template('rst_password.html', user=user),
               text_body=render_template('reset_password.txt', user=user))
```

Приложение Ж

(обязательное)

Файл routes.py

```
from idlelib import query
from flask import render_template, request, redirect, url_for, flash, session,
make_response
import json
import websocket
from websocket import create_connection
from vega import app
from datetime import datetime, timezone
import tzlocal
from .forms import LoginForm, CreateUserForm, AddDeviceForm,
ResetPasswordEmail, ResetPassword
from .email import send_password_reset_email

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')
    print("=====")
    print(f"|command - {req.get('cmd')}|")
    print(resp_json)
    print("=====")
print("sended")
ws.send(json.dumps(req)) # Get command
print("recieved")
resp_json = json.loads(ws.recv())
if resp_json.get('cmd') == "console":
    ws.recv()
    return send_req(req)
print("=====")
print(f"|command - {req.get('cmd')}|")
print(resp_json)
print("=====")
ws.close()
return resp_json

```

```

@app.route("/delete_device/<string:dev_eui>", methods=["GET"])

```

```

def delete_device(dev_eui):
    if session.get('token') is None:
        redirect(url_for('login'))
    device_list = list()
    device_list.append(dev_eui)

```

```

query = {
    "cmd": "delete_devices_req",
    "devices_list": device_list
}
print(f"query - {query}")
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("/delete_user/<string:login>", methods=["GET"])
def delete_user(login):
    if session.get('token') is None:
        redirect(url_for('login'))
    user_list = list()
    user_list.append(login)
    query = {
        "cmd": "delete_users_req",
        "user_list": user_list
    }
    print(f"query - {query}")
    resp = send_req(query)
    if resp.get("status") and resp.get("delete_user_list")[0].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('index'))

@app.route('/dev_graph/<string:dev_eui>/<int:limit>', methods=["GET"])
def dev_chart(dev_eui, limit):
    if session.get('token') is None:
        redirect(url_for('login'))
    context = dict()
    title = f"Chart of {dev_eui}"
    context["legend"] = dev_eui
    query_req = {
        "cmd": "get_data_req",
        "devEui": dev_eui,
        "select":
            {
                "limit": limit
            }
    }
    resp = send_req(query_req)
    if not resp.get("status"):
        flash(resp.get("err_string"), 'error')
        return render_template("index.html")
    if resp.get("cmd") == "get_data_resp":
        data_list = resp.get("data_list")
        labels = list()
        data = list()
        for every_data in data_list:
            # labels.append(datetime.fromtimestamp(every_data.get('ts'),
            # timezone.utc))

```

```

        every_data['ts'] = str(datetime.fromtimestamp(every_data.get('ts')/1000,
timezone.utc))

        labels.append(every_data.get('ts'))
        data.append(every_data.get('data'))
        context["data"] = data
        context["labels"] = labels
        context["raw_data_list"] = data_list
        context["raw_data_list_keys"] = data_list[0].keys()
        return render_template("chart.html", context=context, title=title)

```

```

@app.route('/create_user/', methods=['post', 'get'])
def create_user():
    context = dict()
    form = CreateUserForm()
    if request.method == "POST":
        form.devEui_list.choices = form.devEui_list.data
    if form.validate_on_submit():
        login = form.login.data # запрос к данным формы
        password = form.password.data
        login = form.login.data
        password = form.password.data
        device_access = form.device_access.data
        console_enable = form.console_enable.data
        devEui_list = form.devEui_list.data
        command_list = form.command_list.data
        unsolicited = form.unsolicited.data
        direction = form.direction.data
        with_MAC_Commands = form.with_MAC_Commands.data

```



```

set_data = {
    "login": login,
    "password": password,
    "device_access": device_access,
    "consoleEnable": console_enable,
    "devEui_list": devEui_list,
    "command_list": command_list,
    "rx_settings": {
        "unsolicited": unsolicited,
        "direction": direction,
        "withMacCommands": with_MAC_Commands
    }
}

user_list = list()
user_list.append(set_data)

query = {
    "cmd": "manage_users_req",
    "user_list": user_list
}

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]
dev_names = [dev.get("devName") for dev in devices_list]
form.command_list.choices = session.get('command_list')
form.devEui_list.choices = dev_Euis
return render_template('create_user.html', form=form, context=context)

@app.route('/add_device/', methods=['post', 'get'])
def add_device():
    context = dict()
    form = AddDeviceForm()
    if request.method == "POST":
        if form.is_submitted():
            dev_eui = form.dev_eui.data # запрос к данным формы
            dev_name = form.dev_name.data
            dev_address = form.dev_address.data
            apps_key = form.apps_key.data
            nwks_key = form.nwks_key.data
            app_eui = form.app_eui.data
            app_key = form.app_key.data
            set_data = {
                "devEui": dev_eui,
            }
            if dev_address is not None and apps_key != "" and nwks_key is not
None:
                abp = {
                    "devAddress": dev_address,
                    "appsKey": apps_key,

```

```

        "mwksKey": nwks_key
    }
    set_data["ABP"] = abp
if app_key != "":
    otaa = {
        "appKey": app_key,
    }
    if app_eui != "":
        otaa["appEui"] = app_eui
    set_data["OTAA"] = otaa
if dev_name is not None:
    set_data["devName"] = dev_name
device_list = list()
device_list.append(set_data)
query = {
    "cmd": "manage_devices_req",
    "devices_list": device_list
}
print(f'query add dev - {query}')
resp = send_req(query)
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('add_device.html', form=form, context=context)
return render_template('add_device.html', form=form, context=context)

```

```

@app.route('/')
def index():
    context = dict()
    if 'token' not in session:
        return redirect('login')
    # Get information from connected server
    srvinfo = {"cmd": "server_info_req"} # Don't change!
    # Get device list w/attributes
    devalist = {"cmd": "get_device_appdata_req"} # Don't change!
    # Get reg users
    reguser = {"cmd": "get_users_req"} # Don't change!
    infresp_dict = send_req(srvinfo)
    if infresp_dict.get("err_string") == "unknown_auth":
        return redirect(url_for('login'))
    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
    time_serv_now = infresp_dict.get("time").get("utc") / 1000
    local_timezone = tzlocal.get_localzone()
    serv_time = 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')
    # Get dev list w/attributes
    devalistresp = send_req(devalist)

```

```

context["devices_list"] = devalistresp.get("devices_list")
# Get registered users
reguserresponse = send_req(reguser)
context["user_list"] = reguserresponse.get("user_list")
return render_template('index.html', context=context)

@app.route('/logout/')
def logout():
    if 'token' in session:
        log_out = {"cmd": "close_auth_req", # Don't change!
                    "token": session.get("token")}
        out = send_req(log_out)
        if out.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():
        login = form.login.data # запрос к данным формы
        password = form.password.data
        # Autorization on VEGA server
        autreq = {"cmd": "auth_req", # Don't change!
                  "login": str(login), # Login name
                  "password": str(password) # password

```

```

        }
    autresp = send_req(autreq)
    if autresp.get("err_string") is None:
        session["command_list"] = autresp.get("command_list")
        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():
    form = ResetPasswordEmail()
    if form.validate_on_submit():
        user = form.eml.data
        if user:
            send_password_reset_email(user)
            flash('Check your email for the instructions to reset your password')
            return redirect(url_for('login'))
    return render_template('reset_password_email.html', form=form)

@app.route('/reset_password/<token>', methods=['GET', 'POST'])
def reset_password(token):
    form = ResetPassword()
    if form.validate_on_submit():
        #user.set_password(form.password.data)
        flash('Your password has been reset.')
        return redirect(url_for('login'))
    return render_template('reset_password.html', form=form)

```

Приложение 3

(обязательное)

Файл main.cpp

```
#include "mbed.h"
#include "BME280.h"
#include <arm_acl.h>
#include <cstring>
#include <string>
#include <iostream>
#include <cstdlib>
#include <charconv>

#define MAX_DIGITS 10
RawSerial pc(USBTX, USBRX);
RawSerial dev(D8, D2);
BME280 sensor(I2C_SDA, I2C_SCL);
void dev_recv()
{
    while(dev.readable()) {
        pc.putc(dev.getc());
    }
}

void pc_recv()
{
    while(pc.readable()) {
        dev.putc(pc.getc());
    }
}
```

```

void print_f(const char* temperature, const char* pressure, const char*
humidity)
{
    pc.printf("%i\n", sensor.getTemperature());
    pc.printf("-----\n");
    pc.printf("%s\n", temperature);
    pc.printf("-----\n");
    pc.printf("-----\n");

    pc.printf("%i\n", sensor.getPressure());
    pc.printf("-----\n");
    pc.printf("%s\n", pressure);
    pc.printf("-----\n");
    pc.printf("-----\n");

    pc.printf("%i\n", sensor.getHumidity());
    pc.printf("-----\n");
    pc.printf("%s\n", humidity);
    pc.printf("-----\n");
    pc.printf("-----\n");
}

int main()
{
    int q = 0;
    while(1) {
        char command_WAKE_UP[2] = {'a', 't'};
        for(int i = 0; i < sizeof(command_WAKE_UP); i++)

```



```

{
    dev.putc(command_WAKE_UP[i]);
    pc.printf("%c", command_WAKE_UP[i]);
}
dev.putc('\n');
dev.putc('\r');
wait(3);
pc.baud(115200);
dev.baud(115200);
char command_JOIN[9] = {'a', 't', '+', 'j', 'o', 'i', 'n', '\n', '\r'};
if(q == 3)
{
    for(int i = 0; i < sizeof(command_JOIN); i++)
    {
        dev.putc(command_JOIN[i]);
        pc.printf("%c", command_JOIN[i]);
    }
    q = 0;
    wait(10);
}
q++;
char command_SEND[23] = {'a', 't', '+', 's', 'e', 'n', 'd', '=', 'l', 'o', 'r', 'a', ':',
'1', ':'};
//типа метод по преобразованию инта температуры в символы

```

```

char temp[MAX_DIGITS];
int k = sensor.getTemperature();
pc.printf("\n%d\n", sensor.getTemperature());
int v = 0; //количество цифр в числе n

```

```

//разбиваем на отдельные символы число n
while (k > 9)
{
    temp[v] = (k % 10) + '0';
    k = k / 10;
    v++;
}
temp[v] = k + '0';
temp[v] = '\0';
char t;
//инвертируем массив символов
for (int i = 0; i < v / 2; i++)
{
    t = temp[i];
    temp[i] = temp[v - 1 - i];
    temp[v - 1 - i] = t;
}
v = 0;
for(int i = 0; i < strlen(temp); i++)
{
    command_SEND[i+15] += temp[i];
}
pc.baud(115200);
dev.baud(115200);
for(int i = 0; i < sizeof(command_SEND); i++)
{
    dev.putc(command_SEND[i]);
    pc.printf("%c", command_SEND[i]);
}

```

```

dev.putc('\n');
dev.putc('\r');
wait(7);
pc.attach(&pc_recv, Serial::RxIrq);
dev.attach(&dev_recv, Serial::RxIrq);
char command_SLEEP[28] = {'a', 't', '+', 's', 'e', 't', '_', 'c', 'o', 'n', 'f', 'i', 'g',
'=', 'd', 'e', 'v', 'i', 'c', 'e', ':', 's', 'l', 'e', 'e', 'p', ':', '1'};
for(int i = 0; i < sizeof(command_SLEEP); i++)
{
    dev.putc(command_SLEEP[i]);
    pc.printf("%c", command_SLEEP[i]);
}
dev.putc('\n');
dev.putc('\r');
ThisThread::sleep_for(5000);
}
}

```

Приложение И

(Обязательное)

Чертеж контейнера (в сантиметрах)

