

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
«__» июня 2023 г.

Проверил:
Руководитель проекта
Старший преподаватель каф.
КИБЭВС

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

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

Демиденко Егор Вадимович 731-2;

Коноваленко Александр Дмитриевич 731-2;

Дудник Дарья Андреевна 711-2.

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

14. Календарный план выполнения проекта:

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
1	Ознакомление с проектом		09.02.2023	16.02.2023	
№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
3	Исправление ошибки с отправкой ложных значений	Оптимизация кода и исправление ошибки отправки ложных данных	07.04.2023	30.05.2023	Исправление ошибки в коде

4	Ознакомление с базовой станцией	Изучение основных теоретических источников	16.02.2023	07.04.2023	Получение знаний о принципе работы базовой станции
5	Разработка корпуса для конечного устройства	Моделирование и печать корпуса для конечного устройства	07.04.2023	30.05.2023	Готовый корпус для конечного устройства
6	Изучение API-функций сервера	Ознакомление с базовыми функциями в API-документации сервера ВЕГА	16.02.2023	16.03.2023	Выделенные базовые функции из API-документации для их реализации
7	Настройка динамического обновления времени в веб-приложении	Доработка обновления часов в реальном времени в веб-приложении	29.04.2023	18.05.2023	Синхронизация времени на сервере и клиенте веб-приложения, динамическое обновление в клиентской части
№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
8	Изучение Python, реализация проектов на языке Python	Изучение основ языка программирования Python и реализация проектов на нем	17.02.2023	27.04.2023	Получение знаний об основах программирования на языке Python и несколько

					реализованных проектов на этом языке
9	Настройка динамического обновления графика и реализация уведомлений	Просмотр вариантов реализации уведомлений и реализация обновления графика в режиме реального времени	07.04.2023	18.05.2023	График обновляемый в режиме реального времени
10	Формирование API требований для Web - приложения	Определение необходимого функционала для Web - приложения	16.02.2023	16.03.2023	Сформированные API требования для Web - приложения
№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
11	Работа с пользователями	Изучение функций сервера для настройки пользователей и их ролей.	09.02.2023	13.04.2023	Полученные знания о работе с пользователями на сервере.

12	Настройка прав пользователей	Настройка, создание пользователей и ролей на основной версии проекта	13.04.2023	11.05.2023	Созданные пользователи с определенными ролями, проверенные на корректность работы.
13	Реализация сброса пароля пользователя	Рассмотрение возможных реализаций сброса пароля для пользователя	11.05.2023	25.05.2023	Определение возможных вариантов реализации сброса пароля.
14	Отчётность	Написание отчёта по ГПО	25.05.2026	31.05.2023	Отчёт по проделанной работе.

«09» февраля 2023 г.

Руководитель проекта:

Старший преподаватель каф. КИБЭВС

(должность)

(подпись) Пехов О.В.
(расшифровка)

Члены проектной группы:

(подпись) Подойницын К.В.
(расшифровка)

(подпись) Астра Г.А.
(расшифровка)

(подпись) Демиденко Е.В.
(расшифровка)

(подпись) Коноваленко А.Д.

(подпись) Дудник Д.А.

Реферат

Отчет содержит 95 страниц, 42 рисунков, 10 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 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	11
1 ПРАКТИЧЕСКАЯ ЧАСТЬ	13
1.1 Ознакомление с проектом	13
1.1.1 Ознакомление со средой разработки mbed	13
1.1.2 Изучение API-функций сервера	15
1.2 Ознакомление с базовой станцией	22
1.3 Реализация проектов на языке программирования Python	23
1.4 Назначение прав пользователей	26
1.5 Реализация сброса пароля	30
1.6 Настройка динамического обновления времени в веб-приложении	31
1.7 Настройка динамического обновления графика	33
1.8 Уведомления о повышенных значениях температуры	35
1.9 Настройка базовой станции	37
1.10 Разработка корпуса для конечного устройства	44
1.10 Исправление ошибки с отправкой ложных значений	49
Заключение	51
Список источников	52
Приложение А	53
Приложение Б	55
Приложение В	68
Приложение Г	69
Приложение Д	70
Приложение Е	72
Приложение Ж	75
Приложение З	78
Приложение И	80
Приложение К	84
Приложение Л	85
Приложение М	86
Приложение Н	88
Приложение О	92

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. Согласно прогнозам индийской аналитической компании Fortune Business Insights, глобальный рынок IoT вырастет с 478,36 миллиарда долларов в 2023 году до 2465,26 миллиарда долларов к 2029 году, при среднем росте на 26,4% за прогнозируемый период. Пандемия COVID-19 для интернета вещей была и есть серьезной проблемой: спрос на решения и услуги IoT во всех регионах оказался ниже ожидаемого по сравнению с периодом до пандемии. Согласно анализу, мировой рынок продемонстрировал падение на 23,4% в 2020 году по сравнению с 2019.

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 конечных 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 ПРАКТИЧЕСКАЯ ЧАСТЬ

1.1 Ознакомление с проектом

1.1.1 Ознакомление со средой разработки mbed

MBED OS (также известная как ARM mbed OS) — это операционная система для встраиваемых систем, разработанная компанией ARM для упрощения разработки и управления устройствами Интернета вещей (IoT). Она предоставляет набор инструментов, библиотек и функций, которые облегчают создание и развертывание программного обеспечения на микроконтроллерах и других встраиваемых платформах. [2]

MBED OS предоставляет разработчикам удобный интерфейс для работы с различными аппаратными платформами, периферийными устройствами и сетевыми протоколами.

Для выполнения лабораторных работ использовалась плата STM32Nucleo f103rb и выполнены базовые работы на C++, такие как: мигание светодиодом, работа с сенсором BME280, вычисление криптографического хэша сообщения и вывода его в консоль. Все написанные программы находятся в приложениях Л, М и Н.

Работа в самом начале была затруднена, так как при подключении устройства к программе MBED Studio оно не определялось автоматически. При решении этой проблемы выяснилось, что плата оказалась не рабочей и её пришлось заменить.

После замены устройства при запуске кода появлялась ошибка (рисунок 1.1).

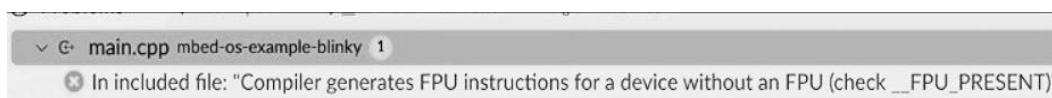


Рисунок 1.1 – Ошибка в работе программы

Проблема заключалась в том, что программа думала, что подключалась плата Floating Point Unit.

Во время решения проблемы в файле «core_cm3.h» была найдена сама строка, которая выводит ошибку (рисунок 1.2) и далее в файле «macros-armclang.cfg» была закомментирована строка «#define __ARM_FP 0x6» (рисунок 1.3). После этого ошибка исчезла.

```
82
83 #elif defined (__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
84     #if defined __ARM_FP
85         #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
86     #endif
```

Рисунок 1.2 - Строка, которая выводит текст ошибки

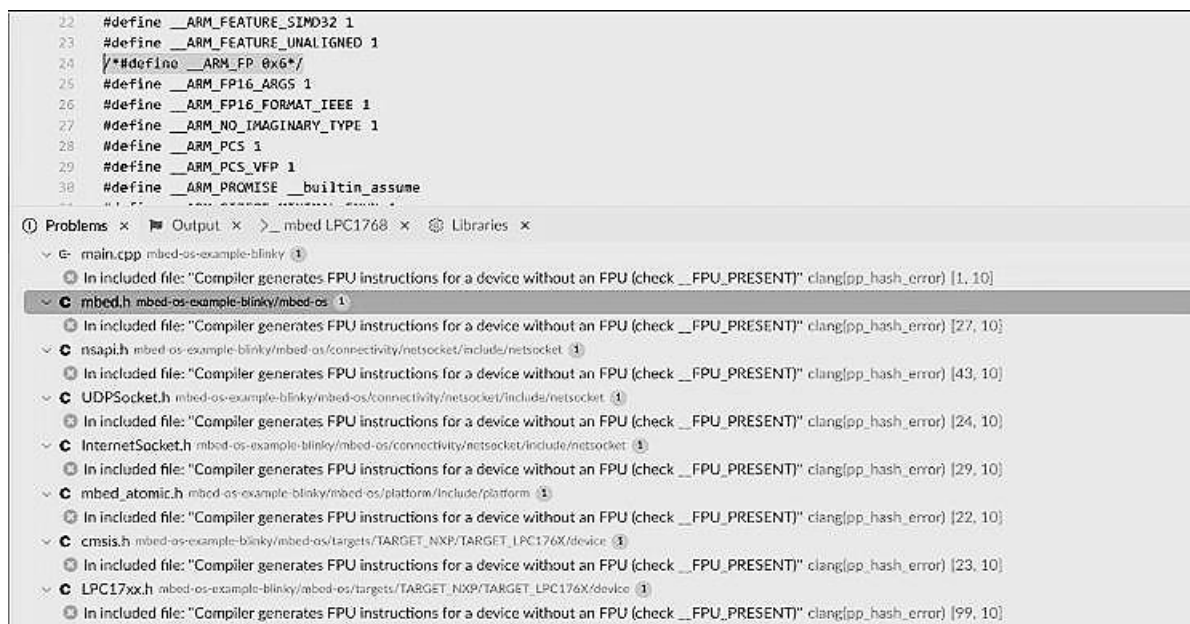


Рисунок 1.3 - Закомментированная строка

1.1.2 Изучение API-функций сервера

Перед продолжением работы над проектом было необходимо изучить документацию на него, в особенности API-документацию сервера Vega для ознакомления с функциями, понимание которых необходимо для дальнейшего возможного совершенствования функционала и логики приложения. [3]

К данным функциям относятся: авторизация пользователя (Рисунок 2.4), получение списка зарегистрированных пользователей (Рисунок 1.5), получение списка подключенных устройств (Рисунок 1.6), возврат сохранённых данных, полученных с устройств (Рисунок 1.7), добавление учетной записи пользователя (Рисунок 1.8), удаление учетной записи пользователя (Рисунок 1.9).

Сообщение с запросом:

```
{
  "cmd": "auth_req",
  "login": string,           // строка без учета регистра
  "password": string        // исходная строка пароля без какой-либо кодировки
}
```

Ответное сообщение:

```
{
  "cmd": "auth_resp",
  "status": boolean,
  "err_string"? : string    //[необязательно существует, если "статус" равен false] – строковый код ошибки
  "token"? : string        //[необязательно существует, если "статус" равен true] – строковый токен
                             сеанса (32 шестнадцатеричных символа)
  "device_access"? : string, //[необязательно существует, если "статус" равен true] – уровень доступа к
                             устройствам
  "consoleEnable": bool,    //[необязательно существует, если "статус" равен true] – разрешить
                             подключение к подсхеме консоли
  "withdebug information"
  "command_list"? :        //[необязательно существует, если "статус" равен true] – доступные
                             команды
  [
    "command_1",
    ...,
    "command_n"
  ],
  "rx_settings"? :         //[необязательно существует, если "статус" равен true] – настройка онлайн-
                             приема сообщений
  {
    "unsolicited": boolean, //онлайн-сообщение отправляется [true] или не отправляется [false]
    "direction": string,    //[необязательно отсутствует, если значение "незапрошено" равно false] –
                             направление возможных онлайн-сообщений
    "withMacCommands":boolean //[необязательно] – онлайн-сообщение содержит команды MAC
  }
}
```

Рисунок 1.4 – Функция для авторизации пользователя

Сообщение с запросом:

```
{
  "cmd": "get_users_req",
  "keyword"? :    //[необязательно] Смотрите описание ниже
  [
    string, ...
  ]
}
```

Возможные строковые значения "ключевого слова":

- "no_command_and_devEui" – возвращает список пользователей без "devEui_list" и "command_list".

Ответное сообщение:

```
{
  "cmd": "get_users_resp",
  "status": boolean,           // Основной статус выполнения
  "err_string"? : string,      //[необязательно существует, если "статус" равен
                               false] – строковый код ошибки

  "user_list":
  [
    {
      "login": string,
      "device_access": string,  //Уровень доступа к устройствам. Если "FULL", то "devEui_list" будет
                               проигнорирован
      "consoleEnable": bool,    //Разрешить подключение к подсхеме консоли с отладочной
                               информацией
      "devEui_list"? :          //[необязательно отсутствует, если существует
                               "no_command_and_devEui"] Список DevEUI, доступный для пользователя
      [
        "devEui_1",
        ...,
        "devEui_n"
      ],
      "command_list"? :         //[необязательно отсутствует, если существует
                               ["NO_COMMAND_AND_DEVEUI"] Список команд, доступных пользователю
      [
        "command_1",
        ...,
        "command_n"
      ],
      "rx_settings"? :          //[необязательно отсутствует, если существует
                               "no_command_and_devEui"] – настройка онлайн-приема сообщений
      {
        "unsolicited": boolean, //онлайн-сообщение отправляется [true] или не отправляется
                               [false]
        "direction"? : string,  //[необязательно отсутствует, если значение "незапрошено"
                               равно false] – направление возможных онлайн-сообщений
        "withMacCommands"? :boolean //[необязательно] – онлайн-сообщение содержит команды MAC
      }
    }, ...
  ]
}
```

Рисунок 1.5 – Функция получения списка зарегистрированных пользователей

Сообщение с запросом:

```
{
  "cmd": "get_device_appdata_req",
  "keyword?":          //[необязательно] Смотрите
                        возможные значения
  [
    string,...
  ]
  "select?:"           //[необязательный] объект фильтра
  {
    "appEui_list?":     //[необязательно] Список соответствующих
                        приложений для запроса
    [
      "appEui_1",
      ...,
      "appEui_n"
    ]
  }
}
```

Возможные строковые значения "ключевого слова":

- "no_attributes" – возвращает список DevEUI без наборов атрибутов;
- "add_data_info" – добавьте дополнительные 3 поля ("last_data_ts", "fcnt_up" и "fcnt_data") к ответу.

Рисунок 1.6 – Функция получения списка подключенных устройств

Сообщение с запросом:

```
{
  "cmd": "get_data_req",
  "devEui": string,
  "select"? :
  {
    "date_from"? : integer,      // [необязательно] Дополнительный параметр для поиска
                                // [необязательно] временная метка UTC сервера в виде числа (миллисекунды с
                                // эпохи Linux)
    "date_to"? : integer,        // [необязательно] временная метка UTC сервера в виде числа (миллисекунды с
                                // эпохи Linux)
    "begin_index"? : integer,    // [необязательно] начать индексацию списка данных [по умолчанию = 0]
    "limit"? : integer,          // [необязательно] ограничение списка данных ответа [по умолчанию = 1000]
    "port"? : integer,           // [необязательно] выберите данные с указанным портом
    "direction"? : string,       // [необязательно] выберите данные с указанным портом
    "withMacCommands"? : boolean // [необязательно] направление перехода сообщения (см. скриншот описания явкс)
  }
}
```

Ответное сообщение:

```
{
  "cmd": "get_data_resp",      // Статус выполнения команды (глобальный статус)
  "status": boolean,           // [необязательно] Если "status" = false, содержит описание ошибки
  "err_string"? : string,

  "devEui": string,            // [необязательно – существует, если "status" = true]
  "appEui": string,            // [необязательно – существует, если "status" = true] Общее существующее количество
  "direction"? : string,        // данных
  "totalNum"? : integer,        // соответствующего типа
  "data_list"? :               // [необязательно – существует, если "status" = true] Данные, передаваемые устройством
  [
    {
      "ts": integer,            // Сервер UTC получает временную метку (миллисекунды от Linux epoch)
      "gatewayId": string,      // Идентификаторы шлюза, которые получают данные с устройства
      "ack": boolean,           // Флаг подтверждения, установленный устройством
      "fcnt": integer,          // Счетчик кадров, 32-разрядное число (возрастающее или убывающее значение в зависимости от значения "направление")
      "port": integer,          // Порт (если = 0, используйте только операции объединения или MAC-команды)
      "data": string,           // Полезная нагрузка расшифрованных данных
      "macData"? : string,       // [необязательно – существует, если "withMacCommands" истинно и присутствует команда MAC]
                                // Данные: MAC-команды с устройства
      "freq": integer,          // Радиочастота, на которой был принят/передан кадр, в Гц
      "dr": string,             // Коэффициент расширения, полоса пропускания и скорость кодирования "SF 12 BW 125
                                // 4/5"
      "rssi": integer,          // [необязательно – существует, если направление пакета "UPLOAD"] Кадр rssi, в dBm,
                                // как целое число
      "snr": float,             // [необязательно – существует, если направление пакета "UPLOAD"] snr кадра, в БД
      "type": string,           // Тип пакета. Может содержать несколько типов, соединенных через "+"
      "packetStatus"? : string   // [необязательно – существует, если направление пакета "UPLOAD"] Статус только
                                // исходящего сообщения
    }, ...
  ]
}
```

Рисунок 1.7 – Функция возврата сохраненных данных, полученных с устройства

Сообщение с запросом:

```
{
  "cmd": "manage_users_req",
  "user_list":
  [
    {
      "login": string,           // Логин пользователя в виде строки
      "password"? : string,     // [необязательно] – строка пароля. Должно существовать при добавлении нового пользователя
      "device_access": string,  // [необязательно] – уровень доступа к устройствам (смотрите ниже возможные значения). Если
                                // "ПОЛНЫЙ", "devEui_list" будет проигнорирован ("ВЫБРАН" - по умолчанию)
      "consoleEnable": bool,    // [необязательно] – разрешить подключение к подсхеме консоли с отладочной
                                // информацией (false – по умолчанию)
      "devEui_list"? :         // [необязательно] – список DevEUI, которые могут быть доступны пользователю
      [
        "devEui_1",           // По умолчанию пусто
        ...,
        "devEui_n"
      ],
      "command_list"? :        // [необязательно] – список групп команд, которые могут быть доступны пользователю
      [
        "command_1",          // По умолчанию пусто
        ...,
        "command_n"
      ],
      "rx_settings"? :         // [необязательно] – настройка онлайн-приема сообщений
      {
        "unsolicited"? : boolean, // [необязательно] – онлайн-сообщение отправляется [true] или не отправляется [false -
                                // по умолчанию]
        "direction"? : string,   // [необязательно] – направление возможных онлайн-сообщений
        "withMacCommands"? : boolean // [необязательно] – онлайн-сообщение содержит команды MAC
      }
    }, ...
  ]
}
```

Рисунок 1.8 – Функция для добавления учетной записи пользователя

```

Сообщение с запросом:
{
  "cmd": "delete_users_req",
  "user_list":
  [
    "login_1",
    ...,
    "login_n"
  ]
}

Ответное сообщение:
{
  "cmd": "delete_users_resp",
  "status": boolean,
  "err_string?": string,      //[необязательно существует, если "статус" равен false] – строковый код ошибки
  "delete_user_list":
  [
    {
      "login": string,
      "status": boolean,
    }, ...
  ]
}

Пример сообщения с запросом:
{
  "cmd": "delete_users_req",
  "user_list":
  [
    "user1",
    "user2"
  ]
}

Пример ответного сообщения:
{
  "cmd": "delete_users_resp",
  "status": true,
  "delete_user_list":
  [
    {
      "login": "user1",
      "status": true
    },
    {
      "login": "user2",
      "status": false
    }
  ]
}

```

Рисунок 1.9 – Функция для удаления учетной записи пользователя

Для дальнейшей реализации поставленной задачи по формированию требований API веб-приложения была изучена структура каждой выделенной функции из документации сервера ВЕГА. Дополнительно было сформулировано требование к динамическому обновлению данных на сервере.

Зная базовый функционал приложения, его структуру и логику, можно в дальнейшем добавлять в него новый функционал или, что более приоритетно – совершенствовать и отлаживать существующий.

1.2 Ознакомление с базовой станцией

Базовая станция Вега предназначена для развёртывания сети LoRaWAN и является центральным элементом построения сети. Основным принципом работы базовой станции является сбор данных с подключенных устройств и дальнейшее использование этих данных.

Первоочередно было изучено руководство по эксплуатации, прилагаемое к станции, а также руководство по развёртыванию сети LoRaWAN. [3,4]

Также были изучены технические характеристики станции Вега, представленные на рисунке 1.10.

Модель	БС-2.2
	ОСНОВНЫЕ
GPS приёмник	да
GSM модем	да
Канал связи с сервером	Ethernet 10/100 Base-T, GSM LTE
Операционная система	Linux
USB-порт	Да
Диапазон рабочих температур	-40...+70 °C
LORAWAN®	
Количество каналов LoRa	8
Частотный диапазон	863-870 МГц
Мощность передатчика	до 500 мВт
Мощность передатчика по умолчанию	25 мВт
Антенный разъём	N-Type female

Рисунок 1.10 - Технические характеристики станции Вега

1.3 Реализация проектов на языке программирования Python

Также для понимания структуры и работы серверной части веб-приложения было необходимо изучить основы языка программирования Python, поскольку ранее данный язык не изучался, и реализовать некоторые проекты на основе полученных знаний после просмотра обучающих основ языка Python курсов. [5]

В целом было реализовано шесть Python-проектов, некоторые из которых приведены ниже (Рисунок 1.11 - 1.13).

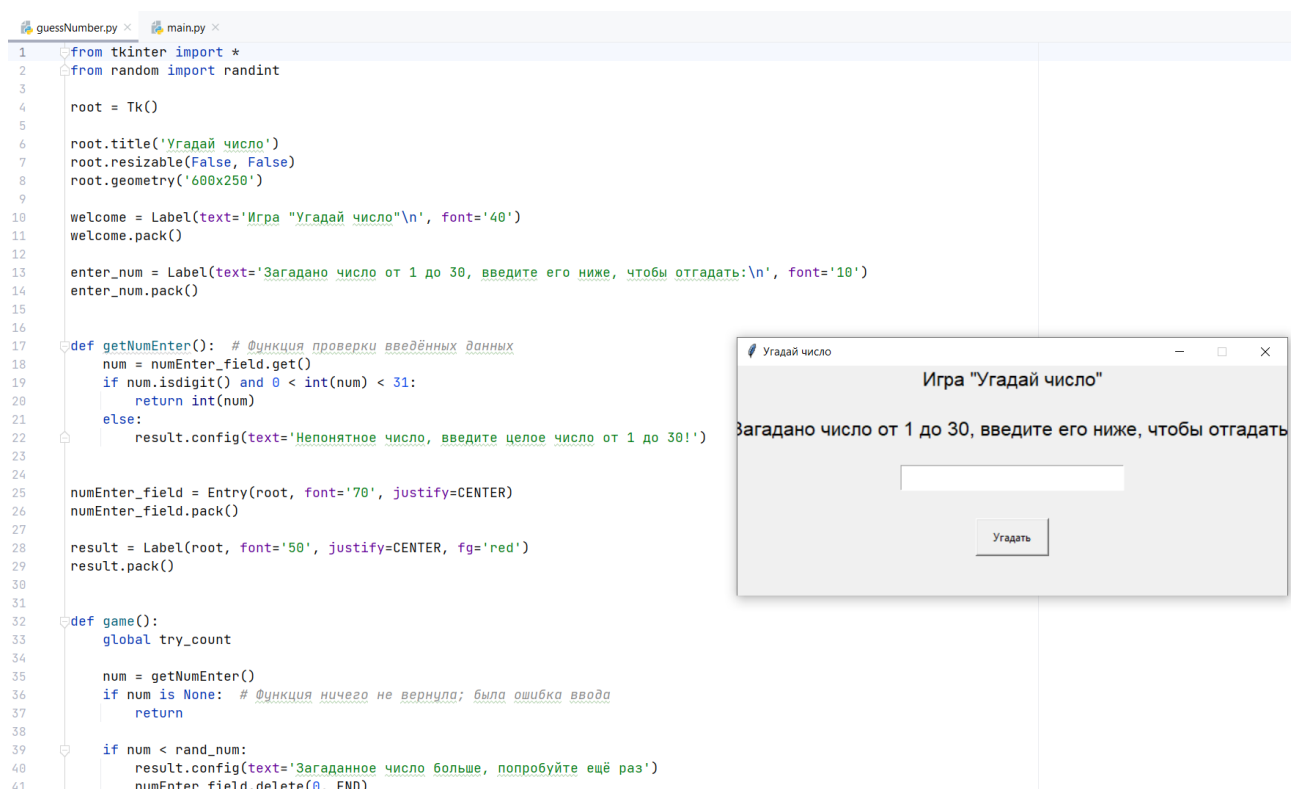


Рисунок 1.11 - Реализация проекта «Угадай число»

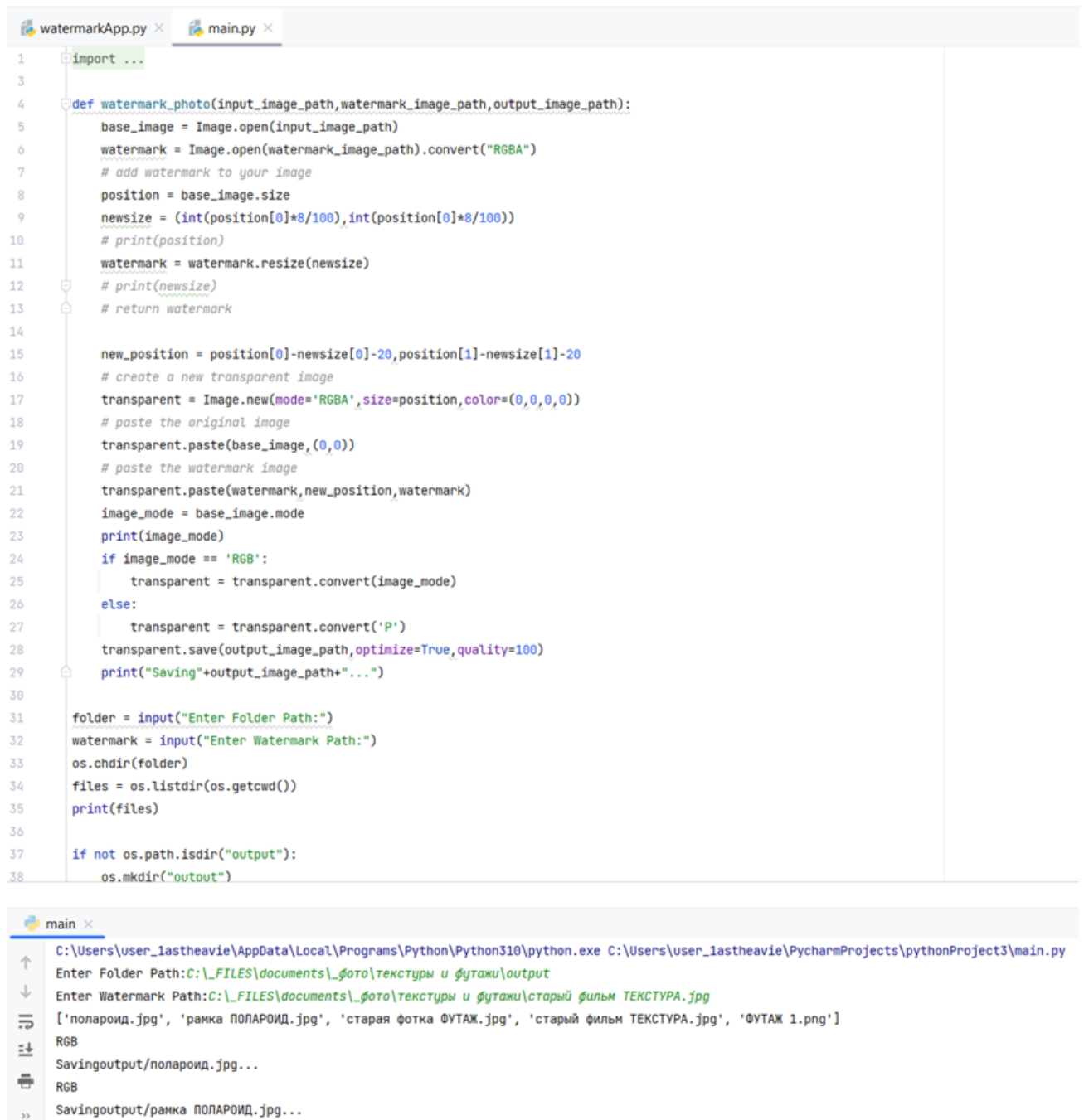


Рисунок 1.12 - Реализация проекта «Вотермарк»

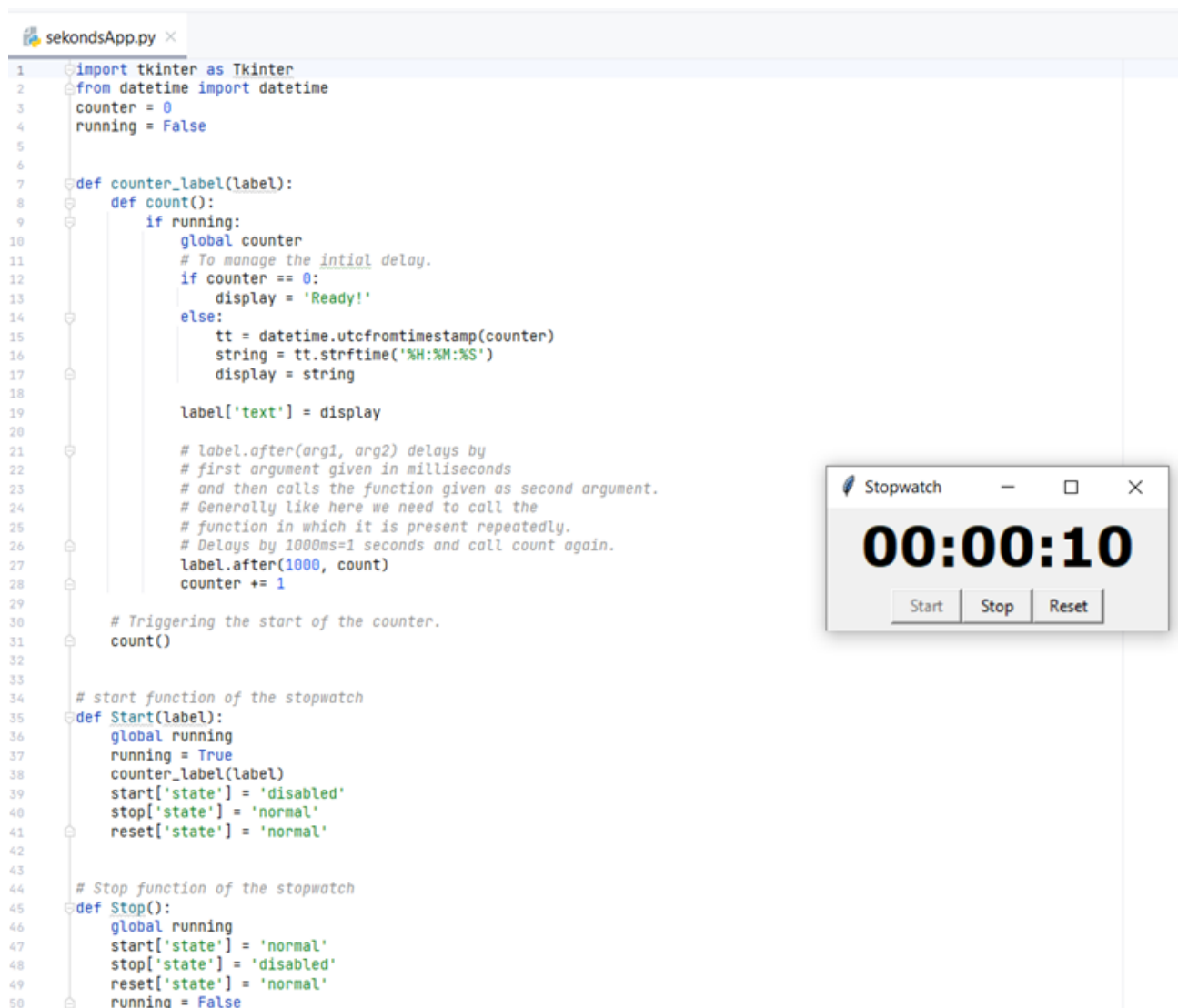


Рисунок 1.13 - Реализация проекта «Секундомер»

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.

Все реализованные проекты можно изучить в приложениях Д – К.

1.4 Назначение прав пользователей

После ознакомления с возможностями сервера по работе с пользователями была переработана UML-схема приложения для определения будущих ролей и функций у пользователей.

UML-схема представлена на рисунке 1.14.

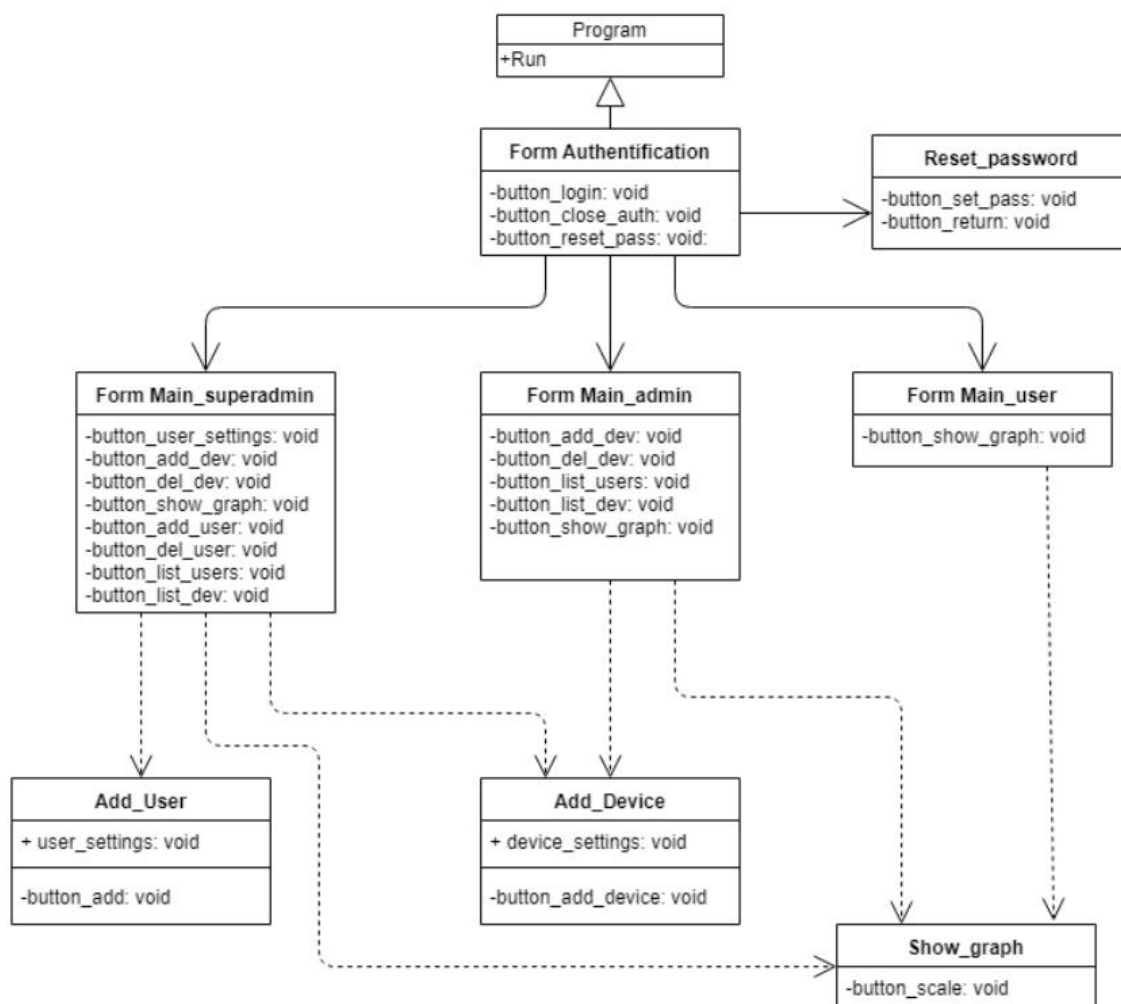


Рисунок 1.14 - UML-схема приложения

Перед началом работы с пользователями необходимо было ознакомиться с настройкой функционала пользователей и ролей в ПО Admin Tool [3], которое необходимо для работы с сервером Vega.

Для каждого созданного пользователя можно задать логин, пароль и подготовленную роль с соответствующими разрешениями и функциями.

Однако можно вручную производить настройку доступных функций и разрешений, которые поделены на три раздела:

- общий;
- внешние приложения;
- сетевое управление.

В раздел общих возможностей пользователя входит просмотр данных с устройств. отправка данных на устройства и использование канала для передачи сообщений на оконечные устройства.

В разделе “Внешние приложения” можно разрешить просмотр, добавление и удаление устройств на сервер из внешних приложений.

В разделе “Сетевое управление” можно настроить следующий функционал пользователя:

- просмотр списка пользователей;
- создание и редактирование пользователей;
- удаление пользователей;
- просмотр идентификатора устройства;
- добавление и редактирование идентификаторов;
- удаление идентификаторов;
- просмотр подключенных устройств;
- добавление, изменение подключенных устройств;
- удаление подключенных устройств из текущей сети;
- просмотр карты покрытия сети;
- просмотр серверной очереди пакетов на отправку;
- редактирование очереди на отправку;
- получение серверной статистики;
- отправка электронных писем.

Также при настройке пользователя можно указать доступ к какому-то определенному устройству.

Настройка и создание пользователей производилась в Vega Admin Tool (Рисунок 1.15 - 1.16).

Home

Devices

Gateways

Users

Exit

ADMIN TOOL V1

Q

CONNECTED USERS

+ Add new user

Login ▾	Device access	Permission		
admin	SELECTED	Custom	⚙	⚙
polzovatel	SELECTED	Custom	⚙	⚙
superadmin	SELECTED	Custom	⚙	⚙

Рисунок 1.15 - Раздел работы с пользователями в Vega Admin Tool

Common

- ☒ View data from devices
- ☒ Send data to devices
- ☒ Send TX to devices

External application

- ☒ View devices by external app
- ☒ Add/edit devices by external app
- ☒ Delete devices by external app

Network management

- ☒ View users
- ☐ Create/edit users
- ☐ Delete users
- ☒ View gateways
- ☒ Create/edit gateways
- ☒ Delete gateways
- ☒ View registered devices
- ☒ Registered/edit devices
- ☒ Delete devices from network
- ☒ View network coverage map
- ☒ View server downlink queue
- ☐ Edit server downlink queue
- ☒ Getting server statistics
- ☐ Send email

Addendum for the groups of devices

Select group devices ▾

Addendum on DevEui

☒ Unselect all

Рисунок 1.16 - Настройка пользователя в Vega Admin Tool

Далее в соответствии с UML-диаграммой были созданы пользователи со следующим ролями:

- админ;
- обычный пользователь;

- главный админ.

У роли обычного пользователя был установлен самый минимальный набор функций, заключающийся в просмотре списка подключенных устройств (View registered devices) и просмотре графиков температур (View data from device).

Интерфейс обычного пользователя представлен на рисунке 1.17.



Рисунок 1.17 - Интерфейс обычного пользователя

У роли администратора был установлен доступ к просмотру списка пользователей (View users), а также включена функция добавления, изменения устройств (Registered/edit devices) и их удаления (Delete devices from network).

Скриншот интерфейса пользователя с ролью администратора представлен на рисунке 1.18.

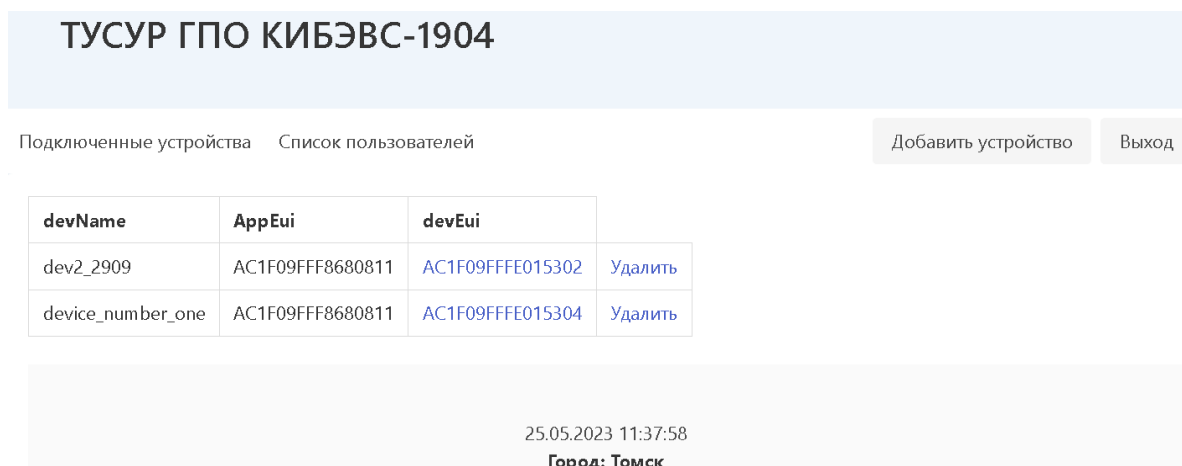


Рисунок 1.18 - Интерфейс администратора

Следующий пользователь имеет роль суперадмина. Данная роль имеет все возможные функции для работы как с устройствами (удаление, добавление и изменение), так и с пользователями.

Интерфейс пользователя суперадмин представлен на рисунке 1.19.

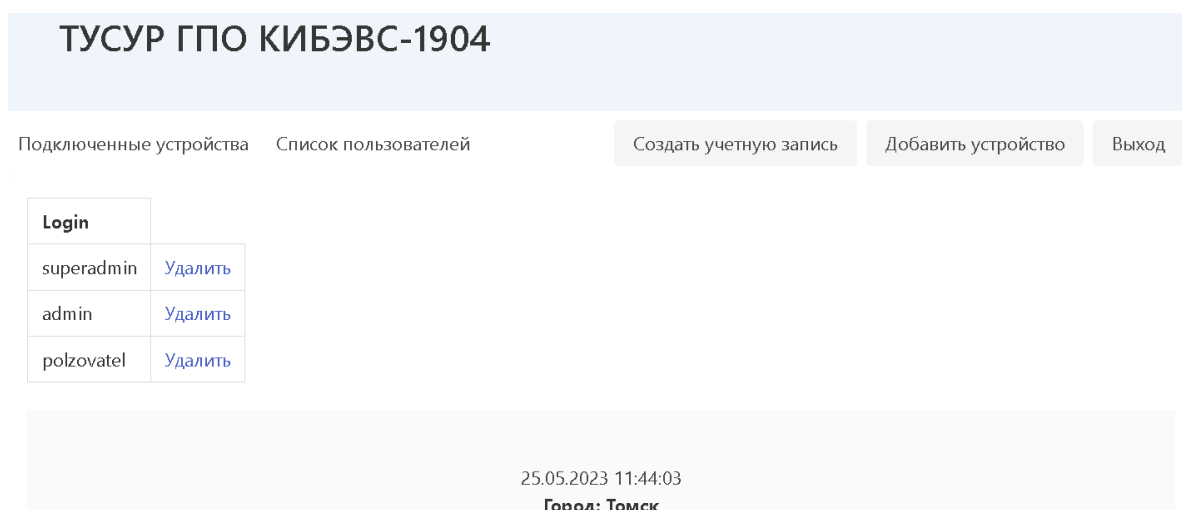


Рисунок 1.19 - Интерфейс пользователя с ролью суперадмин

1.5 Реализация сброса пароля

Для реализации сброса пароля пользователя были рассмотрены следующие методы:

- сброс пароля только администратором;
- сброс пароля с помощью URL-токена;
- сброс пароля с помощью PIN-кодов.

В способе сброса через администратора весь сброс будет проводиться администратором в ручном режиме, где от пользователя потребуется только обращение к нему.

В способе сброса через URL-токен реализация будет основана на генерации токенов и отправке их на электронную почту, который будет указывать пользователь при сбросе пароля. В письме будет указана ссылка с отправленным токеном для сброса пароля.

Сброс пароля с помощью PIN-кодов, отправляемых в сообщении на номер телефона или каким-либо другим способом, является наиболее неподходящим из-за отсутствия встроенных функций сервера.

1.6 Настройка динамического обновления времени в веб-приложении

Одной из задач проекта являлась реализация динамического обновления времени в веб-приложении для удобного его отслеживания при работе с поступающими данными от сервера станции.

В начале было реализовано динамическое обновление времени на серверной части веб-приложения, которое не видно в пользовательской части приложения (Рисунок 1.20).

```

259     time_serv_now = infresp_dict.get("time").get("utc") / 1000
260     local_timezone = tzlocal.get_localzone()
261     serv_time = datetime.fromtimestamp(time_serv_now, local_timezone)
262     context['time'] = serv_time.strftime("%Y-%m-%d %H:%M:%S")
263     context['city'] = infresp_dict.get("time").get("time_zone", 'None')

```

Рисунок 1.20 - Динамическое обновление времени на серверной стороне

Как видно из рисунка 1.20, на строках 260 – 261 устанавливается актуальное региональное время и дата, в то время как на строках 262 – 263 выполняется фиксация этого времени и установленного часового пояса.

Поскольку реализация прямой трансляции времени из серверной части в клиентскую слишком сложна и неэффективна, было принято решение реализовать динамическое обновление времени в клиентской части путем редактирования HTML-кода этой клиентской части (Рисунок 1.21 - 1.22).

```

/* функция получения текущей даты и времени */
function date_time()
{
    var current_datetime = new Date();
    var day = zero_first_format(current_datetime.getDate());
    var month = zero_first_format(current_datetime.getMonth()+1);
    var year = current_datetime.getFullYear();
    var hours = zero_first_format(current_datetime.getHours());
    var minutes = zero_first_format(current_datetime.getMinutes());
    var seconds = zero_first_format(current_datetime.getSeconds());

    return day+"."+month+"."+year+" "+hours+":"+minutes+":seconds;
}

/* выводим текущую дату и время на сайт в блок с id "current_date_time_block" */
document.getElementById('current_date_time_block').innerHTML = date_time();
</script>

<script type="text/javascript">

/* каждую секунду получаем текущую дату и время */
/* и вставляем значение в блок с id "current_date_time_block2" */
setInterval(function () {
    document.getElementById('current_date_time_block2').innerHTML = date_time();
}, 1000);
</script>
{% endblock scripts%}

```

Рисунок 1.21 - Динамическое обновление времени на клиентской стороне

ТУСУР ГПО КИБЭВС-1904

Подключенные устройства Список пользователей

Выход

devName	AppEui	devEui
dev2_2909	AC1F09FFF8680811	AC1F09FFFE015302
device_number_one	AC1F09FFF8680811	AC1F09FFFE015304

25.05.2023 11:03:45

Город: Томск

Рисунок 1.22 - Динамическое обновление времени на клиентской стороне, расположение времени

Обновление времени в клиентской части синхронизировано с обновлением времени в серверной части, что говорит о выполнении поставленной задачи, поскольку в веб-приложении на данный момент из динамических элементов присутствует только время.

1.7 Настройка динамического обновления графика

Для того, чтобы перейти на страницу с графиком необходимо нажать на идентификатор устройства в списке устройств. Для отрисовки графиков была использована JavaScript библиотека визуализации данных ChartJS. [7]

Под графиком на данной странице представлен список полученных пакетов. График представлен на рисунке 1.23.



Рисунок 1.23 - Визуализация данных

Стандартно график строится по последним 48 значениям поскольку с конечных устройств данные будут отправляться на сервер через каждые полчаса, что равняется 48 пакетам в сутки. Было реализовано динамическое обновление графика раз в 30 минут. На рисунке 1.24 представлен фрагмент кода отвечающий за обновление графика.

```

43 $(document).ready(function(){
44     var ctx = document.getElementById('myChart').getContext('2d');
45     var chart = new Chart(ctx, {
46         type: 'line',
47         data: {
48             labels: {{ context.labels|tojson }},
49             datasets: [{
50                 label: 'Температура, *C',
51                 backgroundColor: 'rgb(255, 99, 132)',
52                 borderColor: 'rgb(255, 99, 132)',
53                 data: {{ context.data|tojson }},
54                 tension: 0.1,
55             }]
56         });
57     setInterval(function() {
58         var ctx = document.getElementById('myChart').getContext('2d');
59         var chart = new Chart(ctx, {
60             type: 'line',
61             data: {
62                 labels: {{ context.labels|tojson }},
63                 datasets: [{
64                     label: 'Температура, *C',
65                     backgroundColor: 'rgb(255, 99, 132)',
66                     borderColor: 'rgb(255, 99, 132)',
67                     data: {{ context.data|tojson }},
68                     tension: 0.1,
69                 }]
70             });
71         }, 60000);

```

Рисунок 1.24 - Обновление графика

1.8 Уведомления о повышенных значениях температуры

В результате изучения данного вопроса было принято решение реализовать push-уведомления для получения информации о повышенной температуре в определенной аудитории. Технология push — один из способов распространения информации в интернете, когда данные поступают от поставщика к пользователю на основе установленных параметров. Для реализации push-уведомлений в веб-приложении необходимо наличие протокола http или https с сертификатом SSL.

SSL (Secure Sockets Layer) – протокол безопасности, создающий зашифрованное соединение между веб-сервером и веб-браузером.

SSL-сертификат – это цифровой сертификат, удостоверяющий подлинность веб-сайта и позволяющий использовать зашифрованное соединение.

Так как наше приложение запускается на локальном сервере, то реализовать данные протоколы и получить сертификат не получится, но мы можем реализовать самоподписанный сертификат SSL.

На рисунке 1.25 представлена схема работы push-уведомлений.

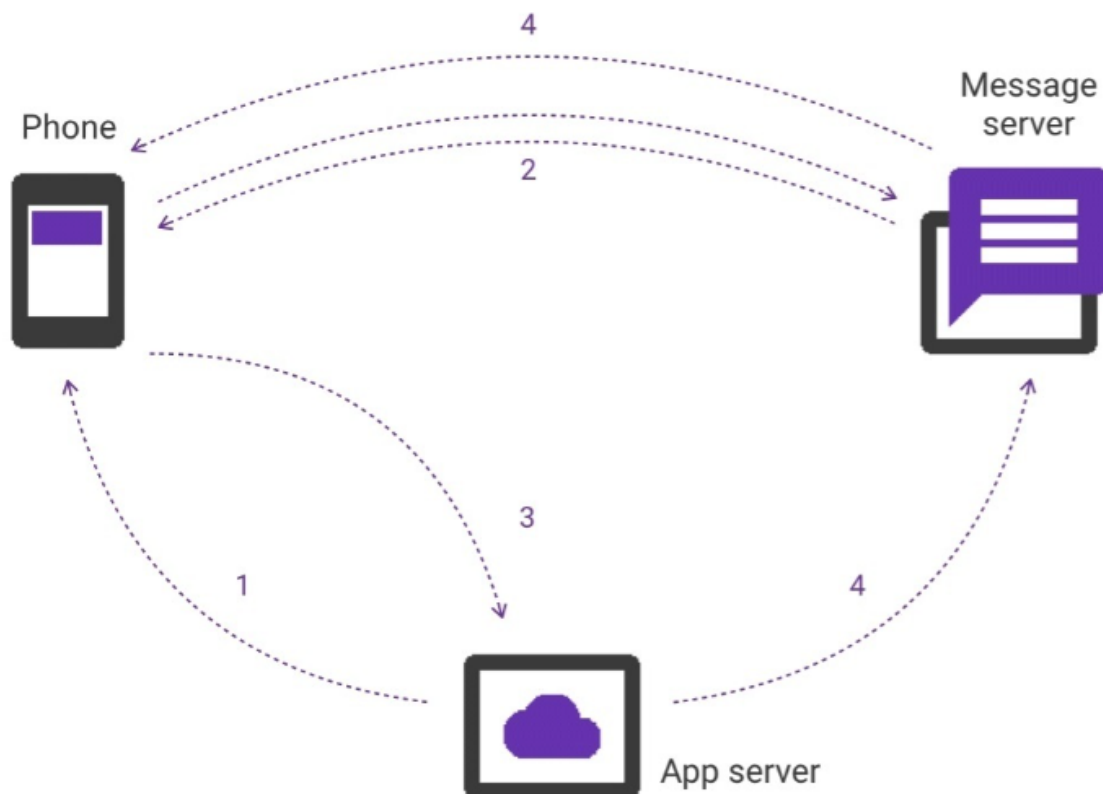


Рисунок 1.25 - Схема работы push-уведомлений

Порядок работы push-уведомлений следующий:

1. Сервер отдает страницу пользователю;
2. Клиент подключается к серверу сообщений, регистрируется и получает ID;
3. Клиент отправляет полученный ID на сервер и сервер привязывает конкретного пользователя к конкретному устройству используя ID устройства;
4. Сервер отправляет сообщение клиенту через сервер сообщений используя полученный ранее ID.

1.9 Настройка базовой станции

Так как серверное API базовой станции «BS-Dashboard» включается автоматически при подаче питания на станцию, то требуется только войти в клиентское браузерное приложение, которое работает с «BS-Dashboard».

Для входа требуется IP-адрес базовой станции, который можно узнать с помощью терминальной программы, в данном случае - PuTTY (Рисунок 1.26). После ввода логина и пароля установленных заранее необходимо прописать команду `ifconfig` для того, чтобы узнать IP - адрес сервера.

```
am335x-evm login: root
Password:
root@am335x-evm:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 6C:C3:74:3E:E7:C7
          inet addr:192.168.17.207  Bcast:192.168.17.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10617 errors:0 dropped:19 overruns:0 frame:0
          TX packets:694 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:710391 (693.7 KiB)  TX bytes:94097 (91.8 KiB)
          Interrupt:56

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:419 errors:0 dropped:0 overruns:0 frame:0
          TX packets:419 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:76339 (74.5 KiB)  TX bytes:76339 (74.5 KiB)

root@am335x-evm:~# █
```

Рисунок 1.26 - Терминальная программа PuTTY

После получения IP-адреса необходимо ввести его в поисковую строку в браузере и откроется страница входа в клиентское приложение. Далее после ввода логина и пароля откроется возможность надстройки самой базовой станции. На главном экране отображаются «Настройки подключения с серверу LoRaWAN». Данная настройка определяет куда будут отсылаются данные с базовой станции. Адрес сервера выбранный

заранее, на который будут отсылааться данные, а также его порты представлены на рисунке 1.27.

192.168.17.207

Базовая станция
Gateway ID: 00006cc3743ee7c7

RU EN
Выйти

Настройки подключения к серверу LoRaWAN

Настройки частотного плана LoRa

Настройки GPS для LoRa

Другие настройки LoRa

Логи LoRa

Настройки 3G

Сетевые настройки

Об устройстве

Настройки

Действия

Режим эксперта: ☐

Настройки подключения к серверу LoRaWAN

Адрес сервера: 192.168.17.105

Верхний порт: 8002

Нижний порт: 8001

Рисунок 1.27 - Настройка подключения к серверу LoRaWAN

Далее располагается настройка частотного плана LoRa. Станция разворачивает сеть LoRa на частотных значениях частотного плана EU868. Выбор данного параметра представлена на рисунке 1.28.

192.168.17.207/frequency

Базовая станция
Gateway ID: 00006cc3743ee7c7

RU EN
Выйти

Настройки подключения к серверу LoRaWAN

Настройки частотного плана LoRa

Настройки GPS для LoRa

Другие настройки LoRa

Логи LoRa

Настройки 3G

Сетевые настройки

Об устройстве

Настройки

Действия

Режим эксперта: ☐

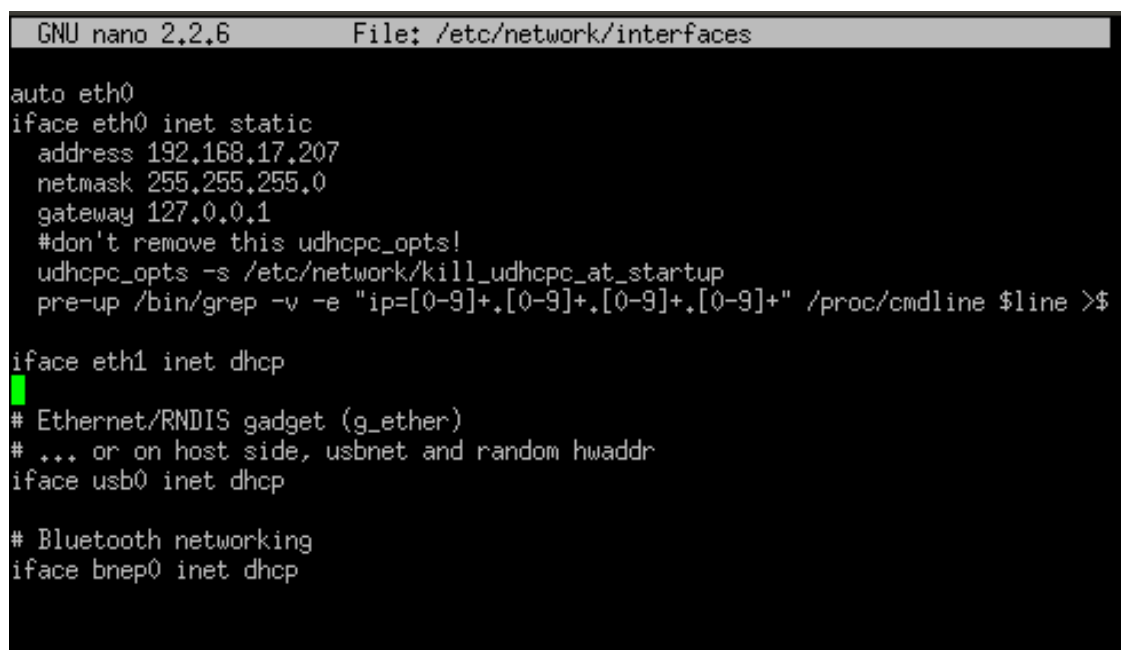
Настройки частотного плана LoRa

Vega RU868 EU868 KZ868

Перезапросить список предустановок

Рисунок 1.28 - Настройка частотного плана

Далее необходима настройка самого IP - адреса базовой станции. Для облегчения подключения IP-адрес базовой станции будет статическим. Это можно сделать двумя способами: через программу терминального доступа, в нашем случае PuTTY, представленную на рисунке 1.29, либо в настройках самой базовой станции на вкладке «Сетевые настройки», представленной на рисунке 1.30.

A screenshot of a terminal window running GNU nano 2.2.6. The title bar shows 'File: /etc/network/interfaces'. The terminal content shows the configuration for network interfaces. The 'eth0' interface is configured with a static IP of 192.168.17.207, a netmask of 255.255.255.0, and a gateway of 127.0.0.1. It also includes a pre-up script that uses grep to check for IP addresses in the command line. The 'eth1' interface is configured for DHCP. There are also comments about Ethernet/RNDIS gadget and Bluetooth networking, with corresponding DHCP configurations for 'usb0' and 'bnep0' interfaces.

```
GNU nano 2.2.6      File: /etc/network/interfaces
auto eth0
iface eth0 inet static
    address 192.168.17.207
    netmask 255.255.255.0
    gateway 127.0.0.1
    #don't remove this udhcpc_opts!
    udhcpc_opts -s /etc/network/kill_udhcpc_at_startup
    pre-up /bin/grep -v -e "ip=[0-9]+.[0-9]+.[0-9]+.[0-9]+" /proc/cmdline $line >$
iface eth1 inet dhcp
# Ethernet/RNDIS gadget (g_ether)
# ... or on host side, usbnet and random hwaddr
iface usb0 inet dhcp

# Bluetooth networking
iface bnep0 inet dhcp
```

Рисунок 1.29 - Сетевые настройки через консоль PuTTY

Возможность изменения сетевых настроек через консоль PuTTY происходит благодаря изменению файла по пути /etc/network/interfaces.

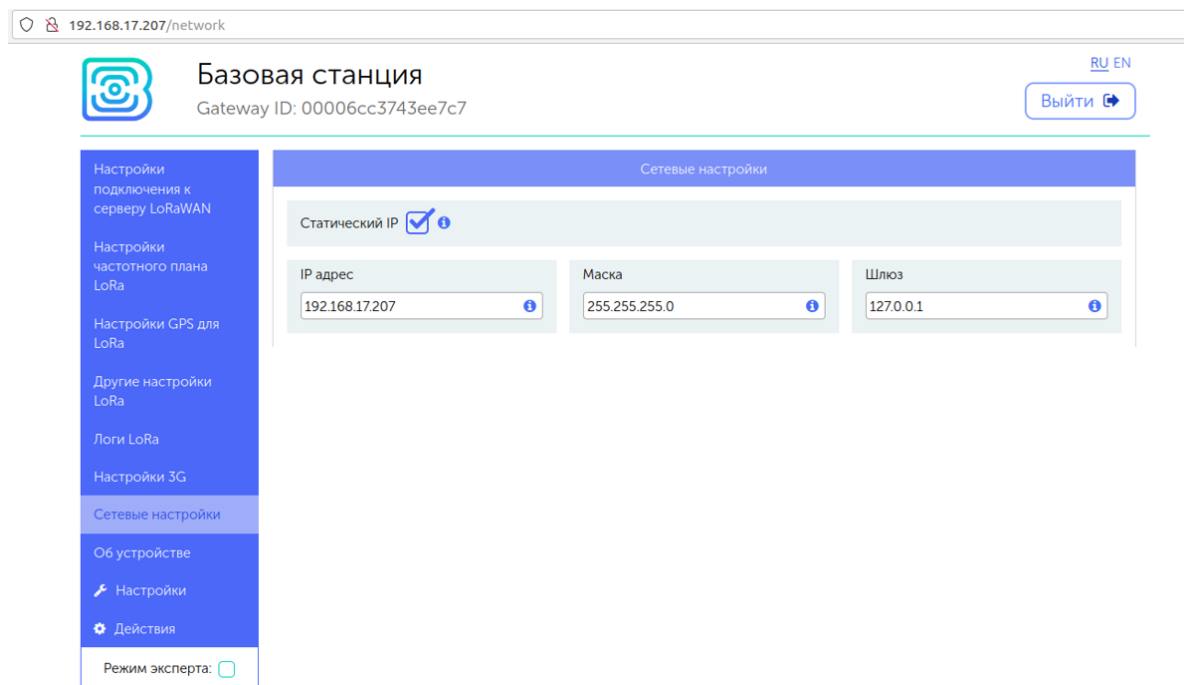


Рисунок 1.30 - Сетевые настройки через настройки базовой станции

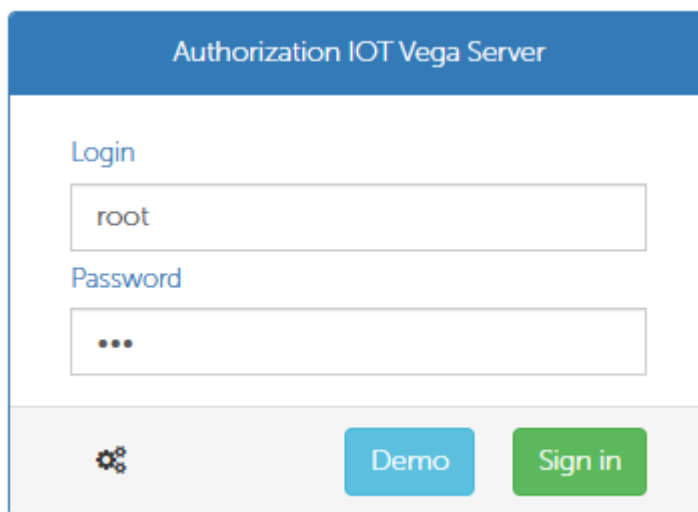
Далее необходимо запустить сервер, на который будут отправляться данные с базовой станции. Для работы было выбрано программное обеспечение - IOT Vega Server. Как упоминалось ранее IP-адрес сервера был выбран заранее, так что после установки программного обеспечения необходимо на компьютере, на котором будет располагаться сервер запустить его через консоль скриптом «`iot-vega-server.sh`». Запуск сервера представлен на рисунке 1.31

```
INFO: Table queuetransmit is cleaned
INFO: Table "bufMacDevParams" is cleared
WebSocketServer has opened. Port[8002]
UDP socket has opened. IP[192.168.17.105]
DEBUG: UdpServer handler is started
INFO: DB-secure scanner started...
DEBUG [CDevicesCountInfo]: vega[0], totalNonVega[1000], usedNonVega[0]
INFO: DB-secure scanner successfully finished
```

Рисунок 1.31 - Запуск сервера

Для управления сервером используется приложение IOT Vega AdminTool. Чтобы начать работу необходимо открыть папку с

программой, найти файл config.js и открыть с помощью любого текстового редактора. Далее необходимо указать IP-адрес нашего сервера в соответствующее поле. Далее необходимо ввести IP-адрес сервера в поисковую строку браузера с припиской на конце «/admin». Откроется окно авторизации, представленное на рисунке 1.32, в котором необходимо ввести логин и пароль.



The image shows a web interface for 'Authorization IOT Vega Server'. It has a blue header with the title. Below the header, there are two input fields: 'Login' with the text 'root' and 'Password' with masked characters. At the bottom, there is a gear icon, a blue 'Demo' button, and a green 'Sign in' button.

Рисунок 1.32 - Авторизация на сервере

Далее будет выведена главная страница, представленная на рисунке 1.33, на которой можно заметить, что в характеристике Gateways будет стоять единица, означающая количество подключенных базовых станций к нашему серверу.

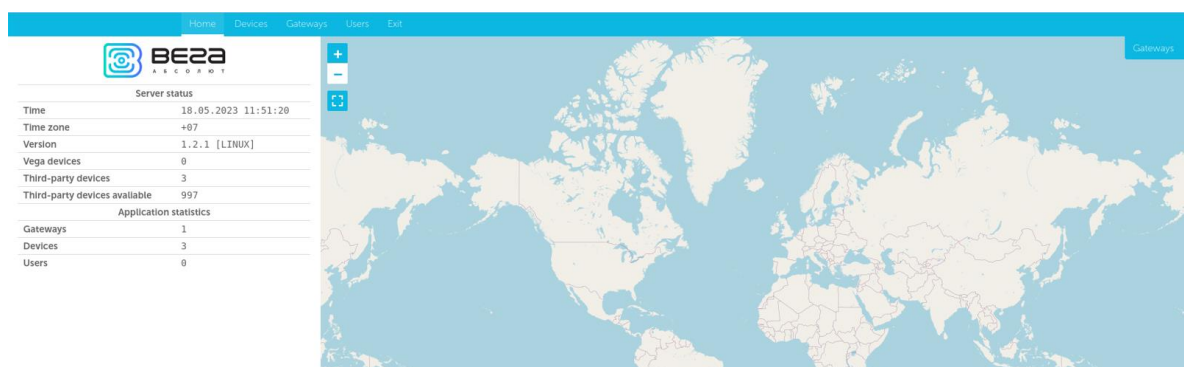


Рисунок 1.33 - Главная страница

Перейдя по одноименной вкладке в верхней панели, мы обнаружим что наша станция успешна подключена к серверу. Об этом свидетельствует совпадение имени, а также активный статус, представленные на рисунке 1.34.

Name	Gateway ID	Active	Latency			
PnP_GW_00006CC3743EE7C7	00006CC3743EE7C7	✓	65535	🔄	⚙️	🗑️

Рисунок 1.34 - Вкладка Gateways

Перейдя во вкладку Devices, представленной на рисунке 1.35, в верхней панели мы сможем добавить устройство, которое будет отправлять данные.

Device name	DevEUI	Last connection	Group				
dev_32	AC1F09FFFE0152FC	-	7391_ldr	🔄	📶	⚙️	🗑️
dev_1	DF69354C001AB1C3	-	7391_ldr	🔄	📶	⚙️	🗑️
	AC1F09FFFE015302	-		🔄	📶	⚙️	🗑️

Рисунок 1.35 - Вкладка Devices

1.10 Разработка корпуса для конечного устройства

Для разработки корпуса конечного устройства были сняты размеры каждого компонента устройства и составлено примерное расположение внутри коробки (рисунок 1.36).

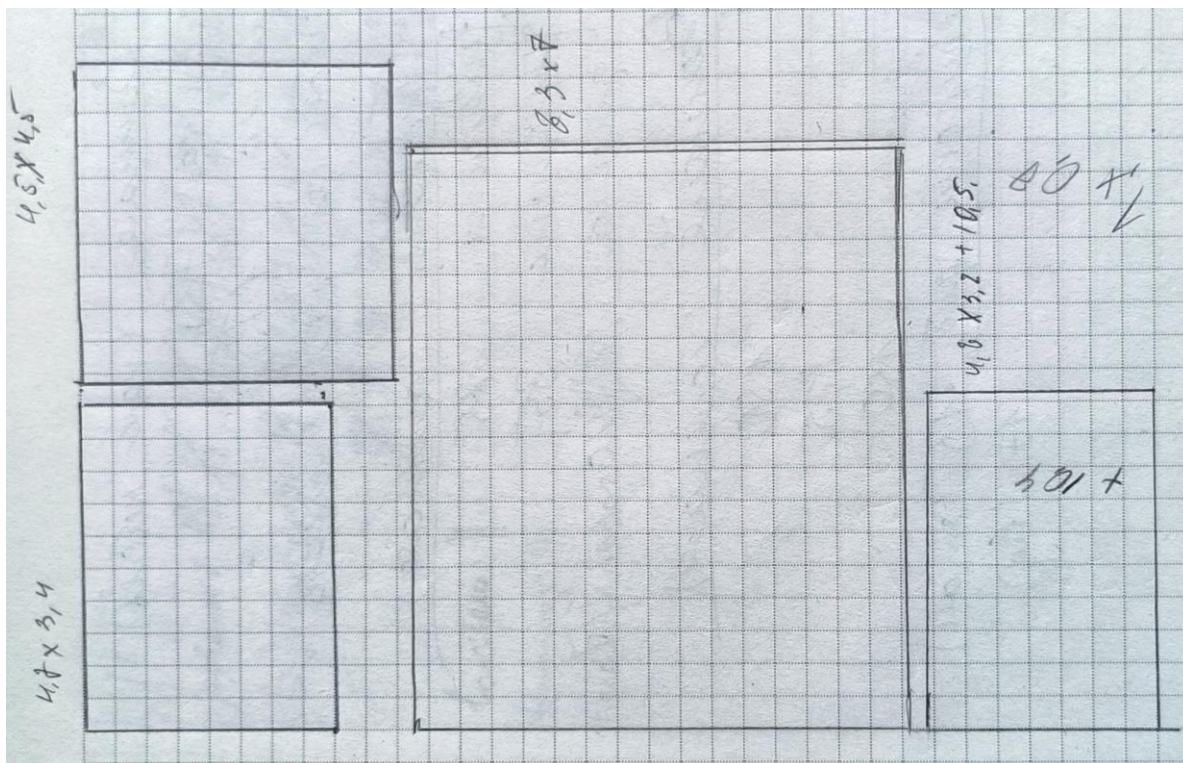


Рисунок 1.36 – Эскиз коробки

В приложении Blender 3.4 была создана первая 3d модель коробки без крышки (рисунок 1.37).

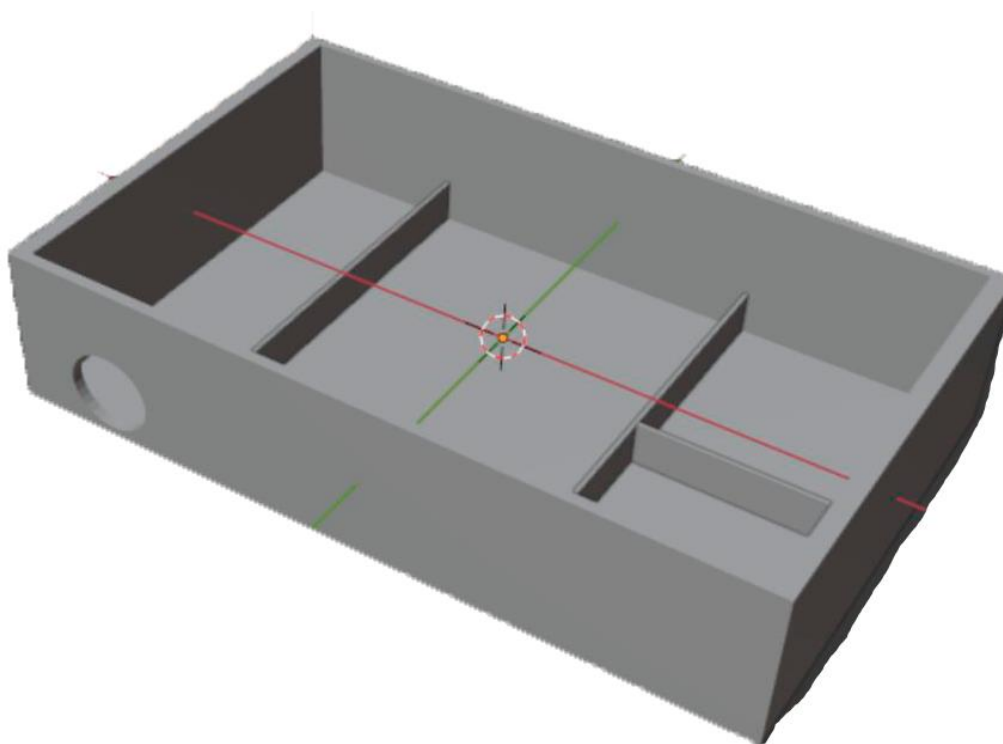


Рисунок 1.37 – Первая 3d модель коробки

Было рассмотрено несколько идей для крышки коробки: крышка на петлях, «пенал» или крышка, которая прикручивается винтами. Выбрана была последняя, так как она надежнее и ее легче реализовать. Впоследствии был произведен переход с приложения Blender 3.4 на сайт tinkercad, так как он оказался удобнее для создания 3d модели. [8]

Первая напечатанная коробка была неудачной по следующим причинам:

- Отверстие для антенны было слишком большим;
- Стенки были слишком тонкие;
- Некоторые внутренние перегородки не соприкасались со стенками;
- Отверстия для шурупов были слишком маленькие;
- У крышки была слишком большая толщина.

Фотография первой напечатанной коробки представлена на рисунке 1.38.

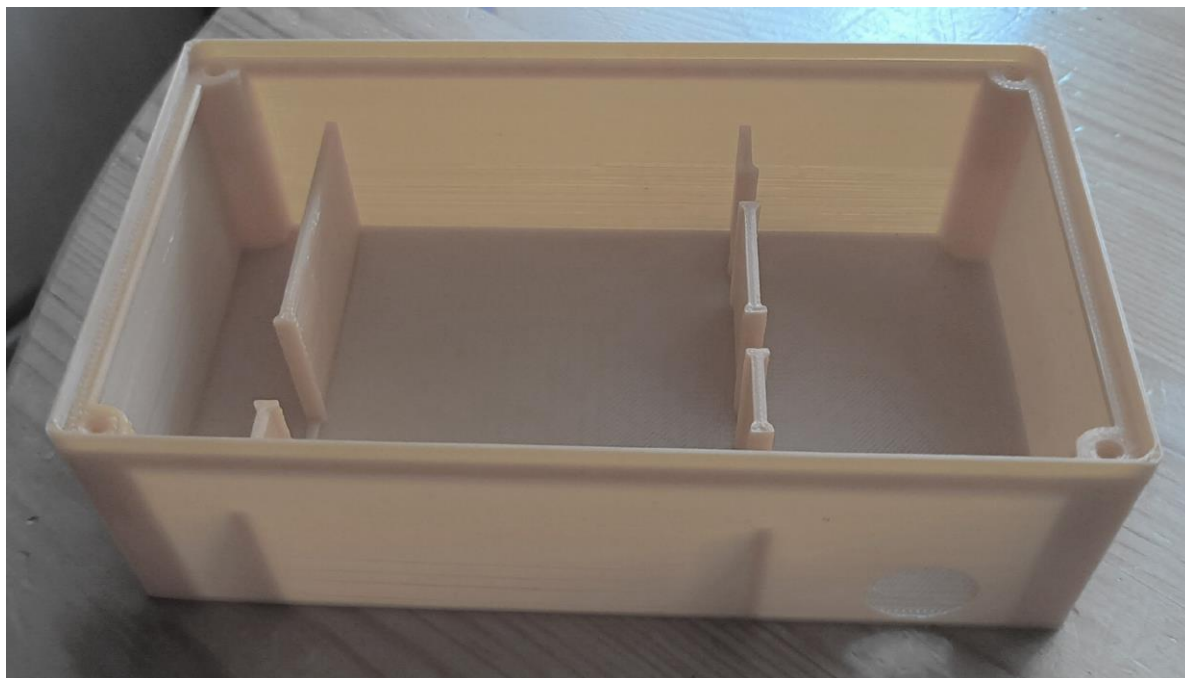


Рисунок 1.38 – Первая напечатанная коробка

В соответствии со всеми замечаниями руководителя были произведены корректировки:

- Крышка стала толщиной 2 мм;
- Отверстия для шурупов сделаны диаметром 2,7 мм, чтобы 4 мм шуруп с учетом изменения пластика при печати мог хорошо держаться;
- У самой коробки размер не был изменен, только добавлено по 1 мм на стенки, чтобы они были прочнее;
- Была убрана опора над местом, где находится плата, чтобы она свободно входила, добавлены две "ножки", которые будут удерживать плату на месте (по 2,7 мм каждая);
- Подняты внутренние перегородки;
- Для платы добавлено отверстие для подключения её к компьютеру и уменьшено отверстие для антенны.

Фотография новой модели коробки изображена рисунке 1.39.

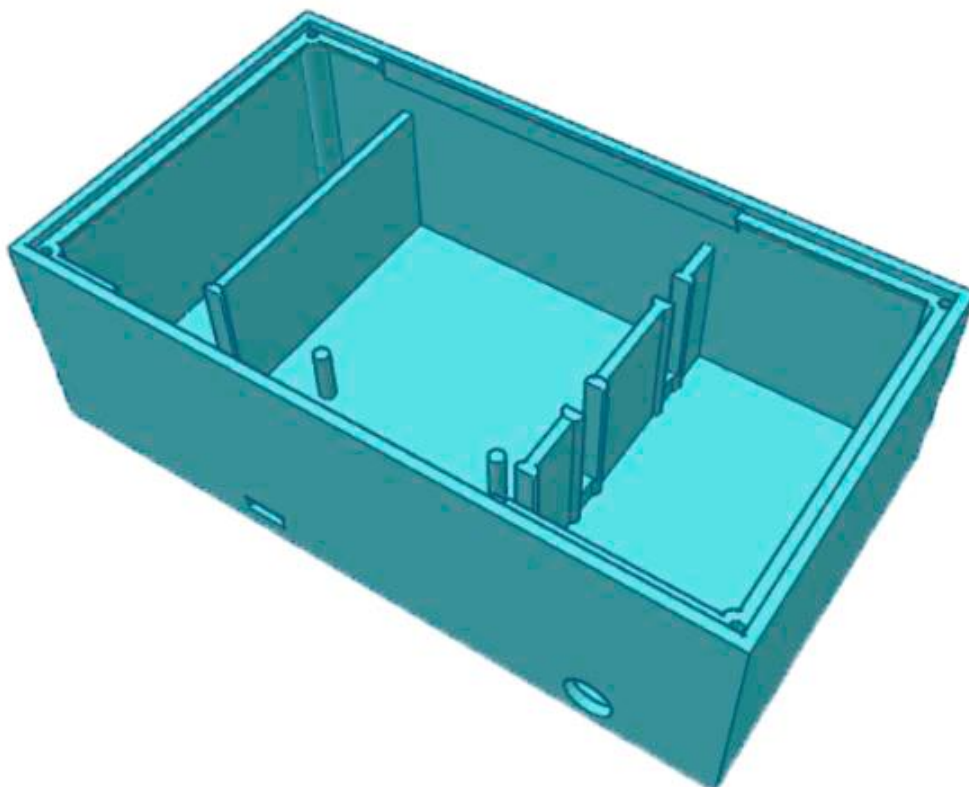


Рисунок 1.39 – Вторая модель коробка

В новой напечатанной коробке хоть и были учтены все ошибки, она оказалась узкой. Даже со специальным отверстием плата туда не поместилась. Поэтому уже на следующей модели были скорректированы все параметры модели. Дополнительно были добавлены утолщения вокруг отверстий для шурупов, чтобы корпус был прочнее. Новая модель изображена на рисунке 1.40.

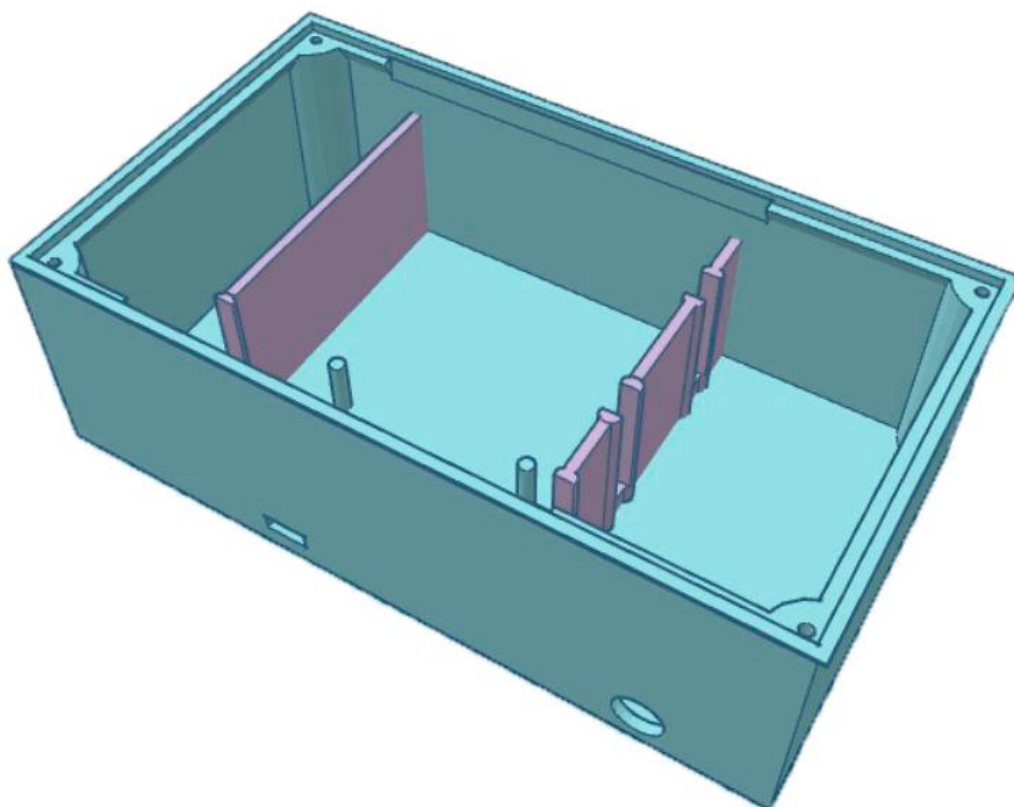


Рисунок 1.40 - Новая модель коробки

1.10 Исправление ошибки с отправкой ложных значений

В коде прошивки конечного устройства была обнаружена ошибка, которая приводила к некорректному выводу данных на сервер. Ошибка была в строке `temp[v++] = (k % 10) + '0'` и заключалась в том, что оператор “++” был применен к переменной `v` внутри массива `temp`. Ошибка приводила к некорректной записи значений температуры в строку. Фрагмент кода, в котором находилась ошибка, изображен на рисунке 1.41.

```
char* temp;
int k = sensor.getTemperature();
pc.printf("\n%d\n", sensor.getTemperature());
temp = (char *)malloc(10 * sizeof(char));
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;
}
temp[v++] = k + '0';
temp[v] = '\0';
```

Рисунок 1.41 - Фрагмент кода с ошибкой

Ошибка была исправлена изменением оператора `temp[v++]` на `temp[v]` и добавлением операции инкремента в отдельной строке после присвоения значения элементу массива `temp`. Скриншот кода с исправлением ошибки изображен на рисунке 1.42.

```

char temp[MAX_DIGITS];
int k = sensor.getTemperature();
pc.printf("\n%d\n", sensor.getTemperature());
int v = 0;
while (k > 9)
{
    temp[v] = (k % 10) + '0';
    k = k / 10;
    v++;
}
temp[v] = k + '0';
temp[v] = '\0';
char t;

```

Рисунок 1.42 - Исправленный код

Также в коде были исправлены следующие строчки:

1. В функции 'printf' параметры 'temperature', 'pressure' и 'humidity' были объявлены как константные указатели на 'char';

2. В циклах 'for' заменена проверка условия 'i < strlen(command)' на 'i < sizeof(command)', чтобы использовать размеры массива 'command', а не вычислять его длину на каждой итерации; [9]

3. Объявлен массив 'temp' фиксированного размера 'MAX_DIGITS' для хранения преобразованных значений температуры. [10]

Исправленный код находится в приложении О.

Заключение

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

- были изучены основы работы с пользователями и ролями на сервере Vega;
- были изучены основы работы со средой разработки MBED OS;
- в коде была исправлена ошибка отправки ложных данных на сервер;
- был спроектирован корпус для конечного устройства;
- были определены и созданы пользователи с ролями на сервере Vega, а также протестирована их работа в веб-приложении;
- были определены и проанализированы возможные варианты реализации сброса пароля пользователя;
- был изучен принцип работы станции Вега;
- была произведена настройка как самой станции Вега, так и серверного обеспечения;
- были сформированы API - требования для веб-приложения;
- были изучены варианты реализации уведомлений для критических значений температуры;
- был реализован график, обновляемый в режиме реального времени;
- был решен вопрос динамического обновления времени на серверной и клиентской стороне веб-приложения.

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. Самоучитель Python [Электронный ресурс]. – Режим доступа: <https://pythonworld.ru/samouchitel-python> (дата обращения 17.02.2023).
6. Руководство по разворачиванию и настройки сети LoRaWAN [Электронный ресурс] - Режим доступа: <https://iotvega.com/product/bs02-2> (дата обращения 16.02.2023)
7. ChartJS – JavaScript-библиотека визуализации данных [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/companies/developersoft/articles/185210/> (дата обращения 14.04.2023).
8. Официальный сайт Tinkercad [Электронный ресурс] – Режим доступа: <https://www.tinkercad.com/> (дата обращения 07.04.2023).
9. Разница между strlen() и sizeof() [Электронный ресурс] – Режим доступа: <https://russianblogs.com/article/3022145414/> (дата обращения 13.04.2023).
10. Типы полей моделей [Электронный ресурс] – Режим доступа: <https://metanit.com/python/django/5.2.php> (дата обращения 27.04.2023).

Приложение А

(обязательное)

Файл run.py

```
from vega import app
from vega import routes

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8000, debug=True)
```

Приложение Б

(Обязательное)

Файл routes.py

```
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

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("Устройство было успешно удалено",
"info")
else:
    flash(f"Устройство не было удалено. потому что
'{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("Учетная запись была удалена", "info")
    else:
        flash(f"Учетная запись не была удалена, потому
что '{resp.get('err_string')}' ", "error")
    return redirect(url_for('index'))

@app.route('/dev_graph/<string:dev_eui>/<string:devNa
me>/<int:limit>', methods=["GET"])
def dev_chart(dev_eui, devName, limit):
    if session.get('token') is None:
        redirect(url_for('login'))
    context = dict()
    title = f"Chart of {devName}"
    context["legend"] = devName
    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:
            every_data['ts'] =
str(datetime.fromtimestamp(every_data.get('ts')/1000,
timezone.utc).strftime("%d/%m/%Y, %H:%M:%S"))
            labels.append(every_data.get('ts'))
            data.append(every_data.get('data'))
        context["data"] = data
        context["labels"] = labels
        context["raw_data_list"] = data_list
        try:
            context["raw_data_list_keys"] =
data_list[0].keys()
        except IndexError:
            flash("Ошибка при построении графика",
'error')
            return redirect(url_for('index'))

        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
        # установка значений для запроса
        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
данному формам
            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
            class_user = form.class_user.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
if class_user is not None:
    set_data["class"] = class_user
set_data["frequencyPlan"] = {
    "freq4": 867100000,
    "freq5": 867300000,
    "freq6": 867500000,
    "freq7": 867700000,
    "freq8": 867900000
}

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
        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(autoreq)
    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("Неверный логин/пароль", 'error')
    return render_template('login.html', form=form)

```

Приложение В

(Обязательное)

Файл forms.py

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

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"]
```

Приложение Г
(Обязательное)

Файл init.py

```
from flask import Flask
import os

app = Flask(__name__, static_folder="static")
basedir = os.path.abspath(os.path.dirname(__file__))
app.config['SECRET_KEY'] = 'hard to guess'
```

Приложение Д
(обязательное)
Файл encoder.py

```
from Crypto.Cipher import AES
from Crypto import Random
from binascii import b2a_hex
import sys

# get the plaintext
plain_text = 'this is none'

# The key length must be 16 (AES-128), 24 (AES-192),
# or 32 (AES-256) Bytes.
key = b'this is a 16 key'

# Generate a non-repeatable key vector with a length
# equal to the size of the AES block
iv = Random.new().read(AES.block_size)

# Use key and iv to initialize AES object, use
# MODE_CFB mode
mycipher = AES.new(key, AES.MODE_CFB, iv)

# Add iv (key vector) to the beginning of the
# encrypted ciphertext
# and transmit it together
ciphertext = iv +
mycipher.encrypt(plain_text.encode())
```

```
# To decrypt, use key and iv to generate a new AES
object
mydecrypt = AES.new(key, AES.MODE_CFB,
ciphertext[:16])

# Use the newly generated AES object to decrypt the
encrypted ciphertext
decrypttext = mydecrypt.decrypt(ciphertext[16:])

# output
file_out = open("encrypted.bin", "wb")
file_out.write(ciphertext[16:])
file_out.close()

print("The key k is: ", key)
print("iv is: ", b2a_hex(ciphertext)[:16])
print("The encrypted data is: ",
b2a_hex(ciphertext)[16:])
print("The decrypted data is: ",
decrypttext.decode())
```

Приложение Е
(обязательное)
Файл guessNumber.py

```
from tkinter import *
from random import randint

root = Tk()

root.title('Угадай число')
root.resizable(False, False)
root.geometry('600x250')

welcome = Label(text='Игра "Угадай число"\n',
font='40')
welcome.pack()

enter_num = Label(text='Загадано число от 1 до 30,
введите его ниже, чтобы отгадать:\n', font='10')
enter_num.pack()

def getNumEnter():  # Функция проверки введённых
данных
    num = numEnter_field.get()
    if num.isdigit() and 0 < int(num) < 31:
        return int(num)
    else:
```

```
        result.config(text='Непонятное число, введите  
целое число от 1 до 30!')
```

```
numEnter_field = Entry(root, font='70',  
justify=CENTER)  
numEnter_field.pack()
```

```
result = Label(root, font='50', justify=CENTER,  
fg='red')  
result.pack()
```

```
def game():  
    global try_count  
  
    num = getNumEnter()  
    if num is None: # Функция ничего не вернула; была  
ошибка ввода  
        return  
  
    if num < rand_num:  
        result.config(text='Загаданное число больше,  
попробуйте ещё раз')  
        numEnter_field.delete(0, END)  
        try_count += 1  
    elif num > rand_num:  
        result.config(text='Загаданное число меньше,  
попробуйте ещё раз')
```



```
numEnter_field.delete(0, END)
try_count += 1
elif num == rand_num:
    result.config(text=f'Вы угадали! Количество
ваших попыток: {try_count}')

btnRead = Button(root, height=2, width=10,
text="Угадать", command=game)
btnRead.pack()

try_count = 1
rand_num = randint(1, 31)

root.mainloop()
```

Приложение Ж
(обязательное)
Файл downloadPages.py

```
from selenium import webdriver
import requests as rq
import os
from bs4 import BeautifulSoup
import time

# path= E:\web
# scraping\chromedriver_win32\chromedriver.exe
path = input("Enter Path : ")

url = input("Enter URL : ")

output = "output"

def get_url(path, url):
    driver =
webdriver.Chrome(executable_path=r"{}".format(path))
    driver.get(url)
    print("loading.....")
    res = driver.execute_script("return
document.documentElement.outerHTML")

    return res
```

```

def get_img_links(res):
    soup = BeautifulSoup(res, "lxml")
    imglinks = soup.find_all("img", src=True)
    return imglinks

def download_img(img_link, index):
    try:
        extensions = [".jpeg", ".jpg", ".png", ".gif"]
        extension = ".jpg"
        for exe in extensions:
            if img_link.find(exe) > 0:
                extension = exe
                break

        img_data = rq.get(img_link).content
        with open(output + "\\\" + str(index + 1) +
extension, "wb+") as f:
            f.write(img_data)

        f.close()
    except Exception:
        pass

result = get_url(path, url)
time.sleep(60)
img_links = get_img_links(result)

```

```
if not os.path.isdir(output):  
    os.mkdir(output)  
  
for index, img_link in enumerate(img_links):  
    img_link = img_link["src"]  
    print("Downloading...")  
    if img_link:  
        download_img(img_link, index)  
print("Download Complete!!")
```

Приложение 3
(обязательное)
Файл watermarkApp.py

```
import os
from PIL import Image

def
watermark_photo(input_image_path, watermark_image_path
, output_image_path):
    base_image = Image.open(input_image_path)
    watermark =
Image.open(watermark_image_path).convert("RGBA")
    # add watermark to your image
    position = base_image.size
    newsize =
(int(position[0]*8/100), int(position[0]*8/100))
    # print(position)
    watermark = watermark.resize(newsize)
    # print(newsize)
    # return watermark

    new_position = position[0]-newsize[0]-
20, position[1]-newsize[1]-20
    # create a new transparent image
    transparent =
Image.new(mode='RGBA', size=position, color=(0, 0, 0, 0))
    # paste the original image
    transparent.paste(base_image, (0, 0))
```

```

    # paste the watermark image

transparent.paste(watermark,new_position,watermark)
    image_mode = base_image.mode
    print(image_mode)
    if image_mode == 'RGB':
        transparent = transparent.convert(image_mode)
    else:
        transparent = transparent.convert('P')

transparent.save(output_image_path,optimize=True,quality=100)
    print("Saving"+output_image_path+".")

folder = input("Enter Folder Path:")
watermark = input("Enter Watermark Path:")
os.chdir(folder)
files = os.listdir(os.getcwd())
print(files)

if not os.path.isdir("output"):
    os.mkdir("output")

c = 1
for f in files:
    if os.path.isfile(os.path.abspath(f)):
        if f.endswith(".png") or f.endswith(".jpg"):
            watermark_photo(f,watermark,"output/"+f)

```

Приложение И
(обязательное)
Файл secondsApp.py

```
import tkinter as Tkinter
from datetime import datetime
counter = 0
running = False

def counter_label(label):
    def count():
        if running:
            global counter
            # To manage the intial delay.
            if counter == 0:
                display = 'Ready!'
            else:
                tt =
datetime.utcnow().timestamp(counter)
                string = tt.strftime('%H:%M:%S')
                display = string

            label['text'] = display

            # label.after(arg1, arg2) delays by
            # first argument given in milliseconds
            # and then calls the function given as
            second argument.
```

```
        # Generally like here we need to call the
        # function in which it is present
repeatedly.

        # Delays by 1000ms=1 seconds and call
count again.

        label.after(1000, count)
        counter += 1
```

```
    # Triggering the start of the counter.
    count()
```

```
# start function of the stopwatch
```

```
def Start(label):
    global running
    running = True
    counter_label(label)
    start['state'] = 'disabled'
    stop['state'] = 'normal'
    reset['state'] = 'normal'
```

```
# Stop function of the stopwatch
```

```
def Stop():
    global running
    start['state'] = 'normal'
    stop['state'] = 'disabled'
    reset['state'] = 'normal'
    running = False
```



```

# Reset function of the stopwatch
def Reset(label):
    global counter
    counter = 0
    # If reset is pressed after pressing stop.
    if not running:
        reset['state'] = 'disabled'
        label['text'] = '00:00:00'
    # If reset is pressed while the stopwatch is
    running.
    else:
        label['text'] = '00:00:00'

root = Tkinter.Tk()
root.title("Stopwatch")

# Fixing the window size.
root.minsize(width=250, height=70)
label = Tkinter.Label(root, text='Ready!',
fg='black', font='Verdana 30 bold')
label.pack()
f = Tkinter.Frame(root)
start = Tkinter.Button(f, text='Start', width=6,
command=lambda: Start(label))
stop = Tkinter.Button(f, text='Stop', width=6,
state='disabled', command=Stop)
reset = Tkinter.Button(f, text='Reset', width=6,
state='disabled', command=lambda: Reset(label))

```

```
f.pack(anchor='center', pady=5)
start.pack(side='left')
stop.pack(side='left')
reset.pack(side='left')
root.mainloop()
```

Приложение К
(обязательное)
Файл searchApp.py

```
import os
text = input("input text : ")
path = input("path : ")
def getfiles(path):
    f = 0
    os.chdir(path)
    files = os.listdir()
    # print(files)
    for file_name in files:
        abs_path = os.path.abspath(file_name)
        if os.path.isdir(abs_path):
            getfiles(abs_path)
        if os.path.isfile(abs_path):
            f = open(file_name, "r")
            if text in f.read():
                f = 1
                print(text + " found in ")
                final_path =
os.path.abspath(file_name)
                print(final_path)
                return True
    if f == 1:
        print(text + " not found! ")
        return False
getfiles(path)
```

Приложение Л
(обязательное)
Файл migalka.cpp

```
#include "mbed.h"

// Частота мигания в миллисекундах
#define BLINKING_RATE      500ms

int main()
{
#ifdef LED1
    DigitalOut led(LED1);
#else
    bool led;
#endif

    while (true) {
        led = !led;
        ThisThread::sleep_for(BLINKING_RATE);
    }
}
```

Приложение М

(обязательное)

Файл sensor.cpp

```
#include "mbed.h"
#include "BME280.h"
#include "OPT3001.h"

DigitalOut led(LED1);
BME280 sensor_bme(D14, D15);
OPT3001 sensor_opt(D14, D15);
EventQueue eventQueue(/* счетчик событий */ 50 *
EVENTS_EVENT_SIZE);
Serial pc(SERIAL_TX, SERIAL_RX);
LowPowerTicker lpTicker;

void printTemperature(void)
{
    pc.printf("%2.2f degC, %04.2f hPa, %2.2f %%\r\n",
sensor_bme.getTemperature(),
sensor_bme.getPressure(), sensor_bme.getHumidity());
    pc.printf("%ld lux\r\n",
sensor_opt.readSensor());
    led = !led; // переключатель LED
}

void tickerIRQ (void)
{
    eventQueue.call(printTemperature);
}
```

```
int main()
{
    pc.baud(115200);

    lpTicker.attach(tickerIRQ, 1); // каждую секунду

    while(1)
    {
        eventQueue.dispatch(0);
        sleep();
    }
}
```

Приложение Н

(обязательное)

Файл hash.cpp

```
#include "mbed.h" //Заголовочный файл для работы с
Mbed
#include "mbedtls/sha256.h" //Заголовочный файл для
работы с SHA-256
#include "mbedtls/platform.h" //Заголовочный файл для
работы с функцией //вывода отладочной информации
mbedtls_printf
#include <string.h> //Заголовочный файл для работы со
строками

static const char hello_str[] = "Hello, world!";
//массив СИМВОЛОВ
static const unsigned char *hello_buffer = (const
unsigned char *)
hello_str; //указатель на массив СИМВОЛОВ
static const size_t hello_len = strlen(hello_str);
//длина массива
//СИМВОЛОВ

static void print_hex(const unsigned char buf[],
size_t len)
{
    for (size_t i = 0; i < len; i++) {
        if (buf[i] <= 0xF) {
            mbedtls_printf("0%x", buf[i]);
```

```

    } else {
mbedtls_printf("%x", buf[i]);
    }
}
mbedtls_printf("\n");
}

int main() {
    int exit_code = MBEDTLS_EXIT_FAILURE;
    //Платформено-зависимая операция инициализации MbedTLS
    if((exit_code = mbedtls_platform_setup(NULL)) != 0)
    {
        printf("Platform initialization failed with
error %d\n\n",
exit_code);
        return MBEDTLS_EXIT_FAILURE;
    }

    mbedtls_aes_context ctx; //Контекст, содержащий
информацию о ключе
    mbedtls_aes_init(&ctx);
    static const char key[] = "secret key!qwerty";
    //Секретный ключи
    static const char data[] = "supertextsuperty"
    //Данные для шифрования
    "\x10\x10\x10\x10"
    "\x10\x10\x10\x10"
    "\x10\x10\x10\x10"
    "\x10\x10\x10\x10";
    unsigned char data_encrypted[32]; //

```



```

    unsigned char iv[16] = {'0', '0', '0', '0', //Вектор
инициализации
    '0', '0', '0', '0',
    '0', '0', '0', '0',
    '0', '0', '0', '0'};
    exit_code = MBEDTLS_EXIT_FAILURE;
    //Добавление 128-ми битного ключа в контекст
    if ((exit_code = mbedtls_aes_setkey_enc(&ctx,
(unsigned const char
*)key, 128)) != 0)
    {
        printf("Can't init aes encryption - error %d\n",
exit_code);
        return MBEDTLS_EXIT_FAILURE;
    }
    exit_code = MBEDTLS_EXIT_FAILURE;
    //Шифрование строки data длиной 32 в data_encrypted
    if ((exit_code = mbedtls_aes_crypt_cbc(&ctx,
MBEDTLS_AES_ENCRYPT, 32,
iv, (unsigned const char *)data, data_encrypted)) !=
0)
    {
        printf("Can't encrypt by aes cbc - error %d\n",
exit_code);
        return MBEDTLS_EXIT_FAILURE;
    }
    printf("Plaintext: \"%s\"\nCiphertext:", data);
    //Вывод текста

```

```
    print_hex(data_encrypted, sizeof data_encrypted);  
    //Вывод зашифрованных данных  
    mbedtls_aes_free(&ctx); //Освобождение контекста  
    //Отключение микроконтроллера  
    mbedtls_platform_teardown(NULL);  
}
```

Приложение О
(обязательное)
Файл main.cpp

```
#include "mbed.h"
#include "BME280.h"
#include <arm_acle.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', ':', 'l',
':'};

```

//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);
}
}

```