

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)


## СБОР МИКРОКЛИМАТИЧЕСКИХ ПАРАМЕТРОВ В УЧЕБНЫХ АУДИТОРИЯХ

Отчет по результатам выполнения индивидуального проекта  
IT Академии Samsung

Исполнители проекта

\_\_\_\_\_ А.А. Лобанов  
(подпись)

\_\_\_\_\_ Э.С. Семенов  
(подпись)

 \_\_\_\_\_ Ц.Б. Цыриторов  
(подпись)

«29» июня 2022 г.

Руководитель проекта

Старший преподаватель каф. КИБЭВС

\_\_\_\_\_ О.В. Пехов  
(подпись)

«29» июня 2022 г.

Министерство высшего образования и науки 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)

УТВЕРЖДАЮ

Заведующий кафедрой КИБЭВС

доктор технических наук, профессор

\_\_\_\_\_ А.А. Шелупанов

«\_\_» \_\_\_\_\_ 2022 г.

ЗАДАНИЕ

на индивидуальный проект IT Академии Samsung  
студентам Лобанову Александру Алексеевичу, Семенову Эдуарду Саввичу,  
Цыриторову Цырену Биликтуевичу группы 739-1 Факультета Безопасности

1. Тема индивидуального проекта: «Сбор микроклиматических параметров в учебных аудиториях».
2. Цель проекта: разработать систему сбора климатических параметров в учебном корпусе на базе технологии LoRa.
3. Основные задачи проекта на этапах реализации:
  - Выбор контролируемых параметров (влажность, температура, давление).
  - Разработка конечного устройства (разработка конструкции, его программирование, решение вопросов питания).
  - Настройка сетевого шлюза LoRa, развертывание сети устройств.

- Настройка сетевого сервера и сервера приложений сети LoRa.
- Написание клиентского приложения.

4. Состав и содержание пояснительной записки (перечень, подлежащих проработке вопросов):

4.1. Общие разделы исполнить согласно требованиям ОС ТУСУР 01-2021.

4.2. Состав рубрик содержания основной части проекта:

- Задачи, аудитория и актуальность проекта.
- Рассмотрение существующих аналогов разрабатываемого решения.
- Архитектура и описание используемых технологий и средств.
- Краткий теоретический материал.
- Практическая часть, включающая в себя решение поставленной задачи.
- Краткое руководство по использованию системы.

5. Дата выдачи задания: «\_\_» \_\_\_\_\_ 2022 г.

6. Срок сдачи студентами законченного проекта «\_\_» июня 2022 г.

Руководитель проекта:

Старший преподаватель каф. КИБЭВС

(должность)

\_\_\_\_\_

(подпись)

Пехов О.В.

(расшифровка)

Задание приняли к исполнению:

\_\_\_\_\_

(подпись)

Лобанов А.А.

(расшифровка)

\_\_\_\_\_

(подпись)

Семенов Э.С.

(расшифровка)



(подпись)

Цыриторов Ц.Б.

(расшифровка)

## Реферат

Отчет содержит 69 страниц, 59 рисунков, 18 источников.

LORA, LORAWAN, LPWAN, IOT, STM32 NUCLEO, RAK, БАЗОВАЯ СТАНЦИЯ, МИКРОКОНТРОЛЛЕР, РАДИОМОДУЛЬ, СБОР ПАРАМЕТРОВ В УЧЕБНЫХ АУДИТОРИЯХ, USB АДАПТЕР, СЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ, БЕЗОПАСНОСТЬ В LORAWAN, КЛИЕНТСКОЕ ПРИЛОЖЕНИЕ.

Объекты исследования: системы интернета вещей.

Предмет исследования: способы проектирования IoT-сетей и создание IoT-систем.

Цель проекта: разработать систему сбора климатических параметров в учебном корпусе на базе технологии LoRa.

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

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

## Содержание

Введение .....	6
1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ .....	7
1.1 API документация .....	7
1.2 Протокол LoRaWAN .....	7
1.3 Цифровой протокол I2C .....	11
2 ОБЗОР АНАЛОГОВ .....	14
2.1 Базовые станции .....	14
2.2 Датчики .....	17
3 АРХИТЕКТУРА ПРОЕКТА .....	20
4 ПРАКТИЧЕСКАЯ ЧАСТЬ .....	22
4.1 Обзор базовой станции .....	22
4.2 API-документация сервера ВЕГА .....	24
4.3 Написание программы для взаимодействия с сервером с помощью API29	
4.4 Сбор микроклиматических параметров .....	35
5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПО ИСПОЛЬЗОВАНИЮ СИСТЕМЫ.	47
5.1. Настройка и использование базовой станции и сервера .....	47
5.2. Работа с устройством по сбору параметров .....	54
Заключение .....	56
Список использованных источников .....	57
Приложение А (Обязательное) Код для взаимодействия с сервером при помощи API .....	59
Приложение Б (Обязательное) Код прошивки микроконтроллера .....	62

## **Введение**

Идея проекта заключается в создании системы по сбору микроклиматических параметров в учебных аудиториях ВУЗа.

Задачи проекта:

- Выбор контролируемых параметров (влажность, температура, давление).
- Разработка конечного устройства (разработка конструкции, его программирование, решение вопросов питания).
- Настройка сетевого шлюза LoRa, развертывание сети устройств.
- Настройка сетевого сервера и сервера приложений сети LoRa.
- Написание клиентского приложения.

Аудитория проекта: ВУЗ.

Актуальность проекта: в настоящее время в ТУСУРе не проводится контроль микроклиматических параметров в учебных аудиториях. Существуют нормы СанПиН 2.4.2.2821-10 «Санитарно-эпидемиологические требования к условиям и организации обучения в общеобразовательных учреждениях», согласно которым, например, температура должна составлять от +18°C до +24°C и отклонение от этих значений будет препятствием для проведения занятий. Данный проект позволит отслеживать параметры в аудиториях ВУЗа, соответственно, соблюдение данных требований станет намного проще.

# **1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **1.1 API документация**

API (Application Programming Interface — «программный интерфейс приложения») — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола, программного каркаса или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы. Используется программистами при написании всевозможных приложений.

API упрощает процесс программирования при создании приложений, абстрагируя базовую реализацию и предоставляя только объекты или действия, необходимые разработчику. [2]

## **1.2 Протокол LoRaWAN**

Спецификация LoRaWAN ® представляет собой сетевой протокол с низким энергопотреблением и глобальной сетью (LPWAN), разработанный для беспроводного подключения «вещей» с батарейным питанием к Интернету в региональных, национальных или глобальных сетях и нацеленный на ключевые требования Интернета вещей (IoT). [3]

Формат сообщения данных представлен на рисунке 1.

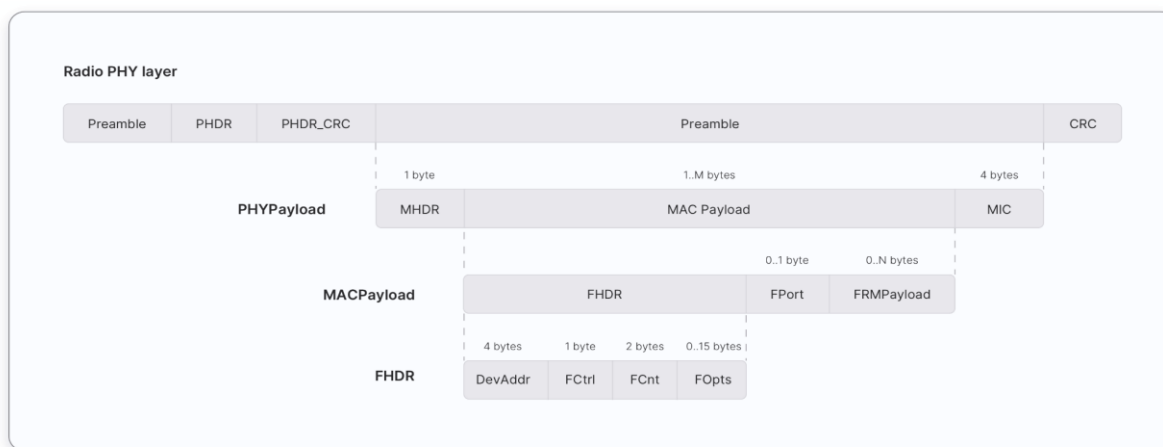


Рисунок 1 – Формат сообщения данных

Полезная нагрузка MAC сообщений данных состоит из заголовка кадра (FHDR), за которым следует необязательное поле порта (FPort) и необязательная полезная нагрузка кадра (FRMPayload).

FHDR состоит из локального адреса устройства в сети (DevAddr), поля управления фреймом (FCtrl), счетчика кадров (FCnt), поля параметров кадра (FOpts)

Сообщение данных может содержать любую последовательность MAC-команд и может одновременно содержать как MAC-команды, так и данные приложения в отдельных полях.

MAC-команды могут отправляться либо в FOpts, либо в поле полезной нагрузки кадра (FRMPayload) сообщения данных, но не в обоих одновременно.

Данные приложения могут быть отправлены в FRMPayload. Это поле не может одновременно содержать команды MAC и данные приложения.

Если поле FRMPayload содержит команды MAC или данные приложения, оно должно быть зашифровано.

### *Безопасность протокола*

LoRaWAN 1.0 использует ряд ключей безопасности: NwkSKey, AppSKey и AppKey. Все ключи имеют длину 128 бит.



NwkSKey (Network Session Key) используется для взаимодействия между узлом и сетевым сервером. Этот ключ используется для расчета и проверки кода целостности сообщения (Message Integrity Code – MIC). [4]

AppSKey (Application Session Key) используется для шифрования и дешифрования полезной нагрузки. Полезная нагрузка полностью зашифрована между узлом и сервером приложений.

Ключи NwkSKey, AppSKey при ОТАА генерируются при каждой процедуре присоединения. При этом они ни в какой момент времени не передаются, а генерируются отдельно на конечном устройстве и сервере.

Эти ключи являются результатом шифрования AES-128 с ключом AppKey. Идентичность данных ключей обеспечивается тем, что и конечное устройство, и сервер знают все параметры шифрования:

$$\text{NwkSKey} = \text{aes128\_encrypt}(\text{AppKey}, 0x01 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce})$$
$$\text{AppSKey} = \text{aes128\_encrypt}(\text{AppKey}, 0x02 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce})$$

0x01, 0x02 – номер ключа (1 – NwkSKey, 2 – AppSKey),

NetID – идентификатор сети. [5]

Случайными числами DevNonce и AppNonce обмениваются конечное устройство и сервер при процедуре присоединения. Они генерируются при каждом запросе присоединения и обеспечивают отличие сессионных ключей для каждого сеанса.

Ключ AppKey (Application Key) – это корневой ключ, специфичный для конечных устройств. Он используется только один раз – при процедуре присоединения. С его помощью подписывается обмен параметрами при процедуре присоединения, также используется для получения сессионных ключей.

В сети IoT LoRaWAN используется многоуровневая система безопасности передачи данных (рисунок 4):

Уровень 1. AES-шифрование на уровне приложения (между конечным

устройством и сервером приложений) с помощью 128-битного переменного сессионного ключа AppSKey. Этот ключ хранится на конечном устройстве и на сервере приложений.

Уровень 2. AES-шифрование и проверка целостности сообщений на сетевом уровне (между конечным устройством и сетевым сервером) с помощью 128-битного переменного сессионного ключа NwkSKey. Этот ключ хранится на конечном устройстве и на сетевом сервере и недоступен клиенту.

Уровень 3. Стандартные методы аутентификации и шифрования интернет-протокола (IPsec, TLS и т.п.) при передаче данных по транспортной сети между узлами сети (базовая станция, сетевой сервер, Join-сервер, сервер приложений). [6]

Сетевой сервер LoRaWAN хранит архив DevNonce. Если сервер получает запрос на присоединения с ранее использованным DevNonce (атака повторного воспроизведения), то он отклоняет запрос и не даёт устройству присоединиться к сети.

В сообщениях LoRaWAN используются счётчики (FCntUp и FCntDown) для защиты от отправки повторных данных. При активации устройства счетчики устанавливаются на 0. Каждый раз, когда устройство отправляет сообщение FCntUp увеличивается на единицу, а когда сервер отправляет – то увеличивается FCntDown. Если устройство или сервер получают сообщение со значением счетчика меньше или равной последнему, то данное сообщение игнорируется.

MIC обеспечивает целостность и подлинность сообщения. Код целостности сообщения вычисляется по всем полям сообщения, а затем добавляется к самому сообщению.

Безопасный обмен сообщениями представлен на рисунке 2.

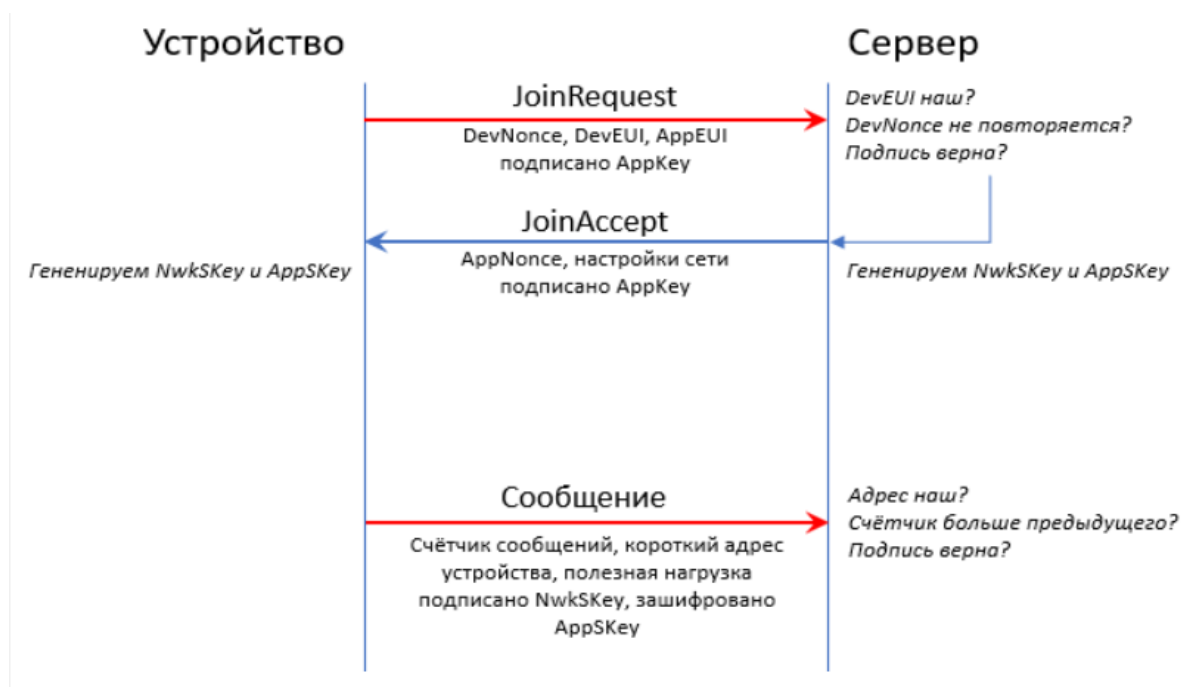


Рисунок 2 – Безопасный обмен сообщениями в сети LoRaWAN

### 1.3 Цифровой протокол I2C

I2C — последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов. Использует две двуправленные линии связи (SDA и SCL), применяется для соединения низкоскоростных периферийных компонентов с процессорами и микроконтроллерами.

I2C(I2C) — это достаточно широко распространённый сетевой последовательный интерфейс, придуманный фирмой Philips и завоевавший популярность относительно высокой скоростью передачи данных (обычно до 100 кбит/с, в современных микросхемах до 400 кбит/с), дешевизной и простотой реализации. Физически сеть представляет собой двухпроводную шину, линии которой называются DATA (SDA) и CLOCK (SCL) (необходим ещё и третий провод — земля, но интерфейс принято называть двухпроводным по количеству сигнальных проводов). Соответственно, по линии DATA передаются данные, линия CLOCK служит для тактирования (рисунок 3). К шине может быть подключено до 128 абонентов, каждый со

своим уникальным номером. В каждый момент времени информация передаётся только одним абонентом и только в одну сторону.

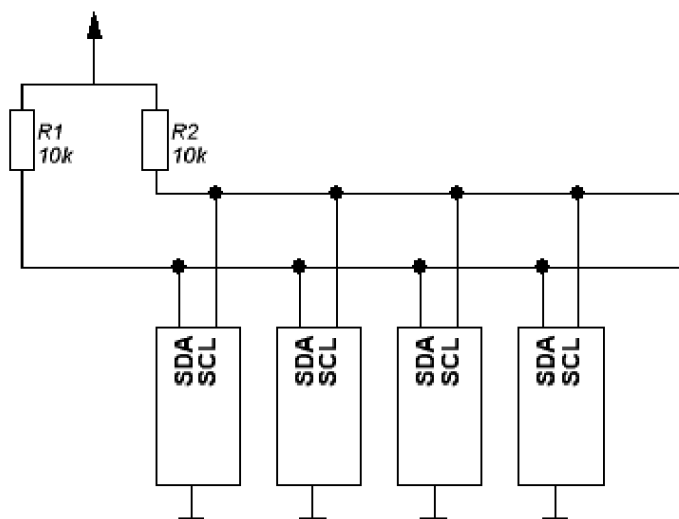


Рисунок 3 – Схема подключения устройств по протоколу I2C

Принцип подключения, следующий: есть ведущий (master) и ведомые (slave). Инициатором обмена всегда выступает ведущий, обмен между двумя ведомыми невозможен. Всего на одной двухпроводной шине может быть до 128 устройств.

Такты на линии SCL генерирует master. Линией SDA могут управлять как мастер, так и ведомый в зависимости от направления передачи. Единицей обмена информации является пакет, обранный уникальными условиями на шине, именуемыми стартовым и стоповым условиями. Мастер в начале каждого пакета передает один байт, где указывает адрес ведомого и направление передачи последующих данных. Данные передаются 8-битными словами. После каждого слова передается один бит подтверждения приема приемной стороной. На рисунке 4 представлена диаграмма по передаче одного слова с подтверждением (такт А). [7]

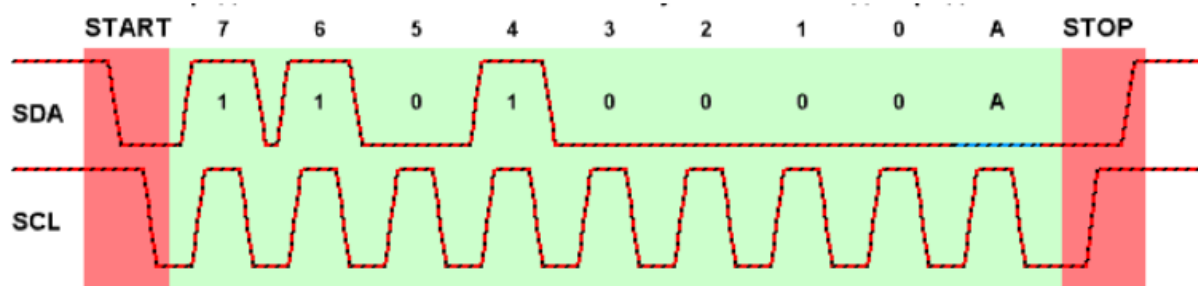


Рисунок 4 - Передача байта с битом подтверждения

## 2 ОБЗОР АНАЛОГОВ

В разрабатываемой IoT-системе используются как базовая станция, так и конечное устройство, которое собирает и передает на сервер данные. Поэтому целесообразно было исследовать рынок на наличие аналогов системы в целом.

### 2.1 Базовые станции

#### 1. R3000 LG от компании Robustel [8]

На рисунке 5 приведено фото данной станции.



Рисунок 5 - R3000 LG от компании Robustel

Ниже приведены основные характеристики данной станции:

- Поддержка международных диапазонов частот LoRaWAN (433-434 МГц и 863-870 МГц).

- Наличие LTE (B1/B3/B7/B8/B20/B28/B31, B38/B40), а также возможность работы на 3G и 2G.

- Мощность передачи сигнала – (+24,5) дБм.

- Дальность связи – до 15 км (на открытой местности).

- ГЛОНАСС/GPS – точность измерения около 2,5 м.

- Наличие следующих интерфейсов: ETHERNET: 2x 10/100 Мбит/с, 2x LAN или 1x LAN + 1x WAN, Micro SD, USB 2.0, CLI, а также различные индикаторы. Питание идет от 3-pin 5 мм розетки (f) с замком. Возможность подключения антенн: LoRa, GSM (2 шт.), ГЛОНАСС/GPS.

- Встроенные часы реального времени, watchdog, таймер.

- Совместима с частными и LoRaWAN-протоколами (классов A и C).

- Совместима с любыми облачными серверами LoRaWAN.

- Две SIM-карты.

- Промышленное исполнение (питание: 9...60 В постоянного тока, диапазон рабочих температур: -40...+75°C).

- Энергопотребление: В покое: 100 мА при 12 В при передаче данных: 400 мА (в пике) при 12 В.

Стоимость данной БС – 197100 руб., что в несколько раз превосходит используемую в проекте.

## 2. Вега БС-3 [9]

На рисунке 6 представлено фото данной БС.

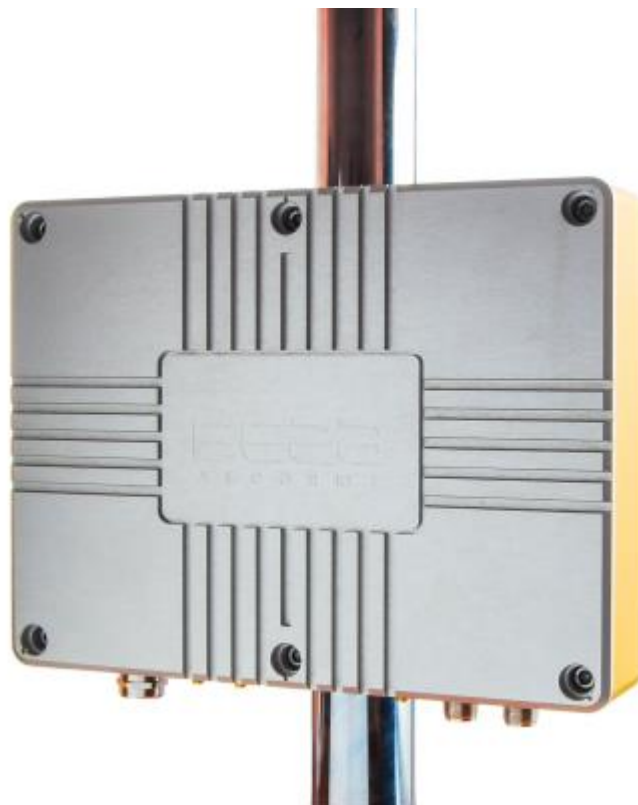


Рисунок 6 - Вега БС-3

Ниже приведены основные характеристики данной станции:

- Наличие ГНСС-модуля с поддержкой ГЛОНАСС, GPS, а также дополнительных систем SBAS, Galileo и других.
- Наличие LTE-модема.
- Поддержка международных диапазонов частот LoRaWAN (863-870 МГц).
- Дальность связи – до 15 км (на открытой местности).
- Напряжение питания – 24 В.
- Совместима с частными и LoRaWAN-протоколами (классов А и С).
- Промышленное исполнение (питание: 9...60 В постоянного тока, диапазон рабочих температур: -40...+70°C).
- Наличие USB-порта.
- Количество каналов LoRaWAN – до 16 (в этом случае потребляемая мощность становится равной 30 Вт).

Стоимость данной БС – 138700 руб.

### 3. Другие базовые станции [10]



На рынке присутствуют и другие модели базовых станций от разных производителей. На рисунке 7 представлены некоторые подробно не рассмотренные варианты.



Рисунок 7 - Существующие аналоги БС

Их технические характеристики схожи с рассмотренными ранее устройствами. Основное – работа в доступном в России частотном диапазоне 863-870 МГц, работа на большие расстояния (до 15-20 км). Ценовой диапазон – 110000-250000 (самый дорогой – Kerlink iBTS).

## 2.2 Датчики

В качестве готовых решений конечных устройств, которые бы передавали данные на сервер, были рассмотрены существующие датчики по снятию температуры, отправляющие показания по протоколу LoRaWAN.

### 1. Dragino LHT52 [11]

На рисунке 8 представлено фото данного устройства.



Рисунок 8 - Dragino LHT52

Распространенный датчик, производство – Китай. Основные характеристики приведены ниже:

- Отправка данных по LoRaWAN (класс A);
- Отсутствие дисплея;
- Данные можно получить на сервере через LoRaWAN Downlink;
- Наличие батареи (2\*AAA). Работа до нескольких лет;
- Встроен USB Type-C, по которому можно подключать внешние датчики;
- Крепится на стену.

Стоимость датчика – 4500 руб.

Удобство заключается в компактности и удобстве эксплуатации (замена батареек, а не ожидание зарядки аккумулятора, подключение дополнительных датчиков). Из недостатков – только отправляет данные, не способен принимать ответы от сервера или БС; некачественный крепеж (на клейкой ленте); при отсутствии батареек нет возможности зарядить как аккумулятор.

## 2. LoRa RAK 7204 [12]

На рисунке 9 представлено фото данного устройства.



Рисунок 9 - LoRa RAK 7204

Датчик от известной компании по производству оборудования для IoT. Ниже приведены характеристики данного устройства:

- Отправка данных по LoRaWAN;
- Отсутствие дисплея;
- Данные можно получить на сервере через LoRaWAN Downlink;
- Наличие аккумулятора (3500 мАч). Работа до 3 лет от одного заряда;
- Заряжается через Type-C.

Стоимость датчика – 6500 руб.

Аналогичен предыдущему, но большей стоимостью и аккумулятором вместо батареек.

### 3 АРХИТЕКТУРА ПРОЕКТА

Проект состоит из следующих элементов:

1. Устройство по сбору микроклиматических параметров – конечное устройство, отправляющее сообщения по беспроводной сети на шлюз и/или получающее сообщения от него. Он состоит из следующих элементов:

- Датчик – измерительное устройство.
- Модуль LoRa – премопередатчик с технологией LoRa, предназначенный для беспроводной передачи данных.
- Микроконтроллер – главный элемент конечного устройства, управляющий датчиком и модулем LoRa.

2. Базовая станция – центральный элемент построения сети на основе технологии LoRaWAN. Работает по принципу прозрачного шлюза между оконечными устройствами и сервером .

3. Сервер – элемент сети, отвечающий за управление и обслуживание всей сетью LoRa.

4. Клиент – клиентское приложение, с помощью которого пользователь имеет возможность взаимодействовать с сетью LoRa.

Схема архитектуры представлено на рисунок 10.

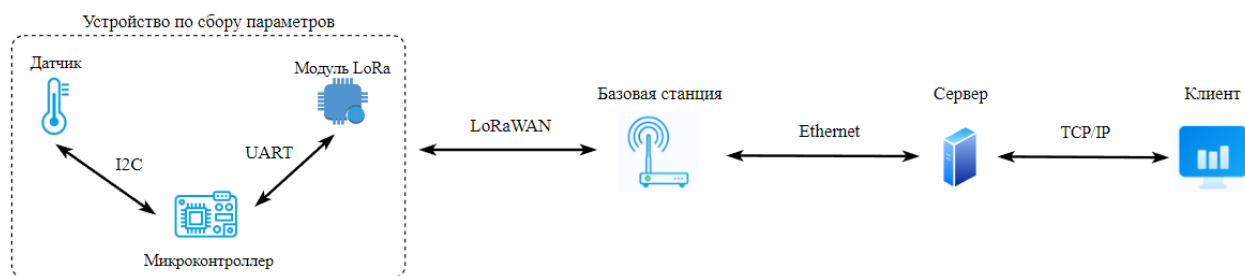


Рисунок 10 – Архитектура проекта

Для написания программы взаимодействующей с сервером Вега использовался язык программирования Python с подключенной библиотекой “websocket”. Программный код был написан с помощью IDE Visual Studio Code.

Для создания прошивки микроконтроллера Nucleo F103RB использовался язык программирования C++ с подключенными библиотеками “mbed” и “BME280”. Программный код был написан в программной платформе для разработки IoT-устройств Mbed Studio.

В устройстве по сбору параметров используются следующие технологии: шина I2C для соединения микроконтроллера и сенсора и протокол UART для связи микроконтроллера с модулем LoRa. Само устройство для связи с базовой станцией использует протокол LoRaWAN. Базовая станция в свою очередь подключается к серверу по Ethernet. А клиент взаимодействует с сервером по протоколу TCP/IP.

## **4 ПРАКТИЧЕСКАЯ ЧАСТЬ**

### **4.1 Обзор базовой станции**

Базовая станция Вега БС-2.2 от компании «ВЕГА АБСОЛЮТ» (рисунок 11) предназначена для развёртывания сети LoRaWAN на частотах диапазона 863-870 МГц. Базовая станция – это центральный элемент построения сети на основе технологии LoRaWAN® и работает по принципу прозрачного шлюза между оконечными устройствами и сервером. [13]

Питание базовой станции и сообщение с сервером осуществляется через канал Ethernet, кроме того, в БС-2.2 используется 3G модем, который поддерживает частоты:

Dual-Band UMTS (WCDMA/FDD) 900 и 2100 МГц

Dual-Band GSM 900 и 1800 МГц

Другой особенностью базовой станции Вега БС-2.2 является наличие GPS/ГЛОНАСС модуля.

Базовая станция имеет предустановленное встроенное ПО на основе операционной системы Linux. При работе с базовой станцией рекомендуется использовать антенну 868-01-A10 мощностью 10 дБм.

Указанные характеристики дальности связи (рисунок 11) достигались при различных натурных экспериментах. Реальная дальность связи зависит от многих факторов и требует измерения в конкретных условиях с помощью тестера сети.

Базовой станции Вега БС-2.2 присвоен статус телекоммуникационного оборудования российского происхождения (ТОРП). Продукция внесена в Единый реестр российской радиоэлектронной продукции (ПП РФ №878).



#### Характеристики

GPS приемник	да, со встроенной антенной
3G модем	да
Операционная система	Linux
Канал связи с сервером	Ethernet, GSM 3G
USB-порт	да
Диапазон рабочих температур, °C	-40...+70
Количество каналов LoRaWAN®	8
Частотный диапазон	863-870 МГц
Мощность передатчика	до 500 мВт (27 dBm)
Антенный разъём	N-Type female
Дальность радиосвязи в сельской местности	до 15 км
Дальность радиосвязи в плотной городской застройке	до 5 км
Потребляемая мощность	до 10 Вт
Тип питания	Passive POE 4.5(+) 7.8(-) 15Вт
Напряжение питания	12...48 В
Размеры корпуса, не более, мм	192 x 183 x 75
Степень защиты корпуса	IP67
Крепление	на балки/мачты
Габариты упаковки, мм	250 x 220 x 85
Вес комплекта в упаковке, кг	1,250

Рисунок 11 – Вид и характеристики базовой станции Вега БС-2.2

## 4.2 API-документация сервера ВЕГА

При изучении API-документации сервера ВЕГА были выделены те функции, которые необходимо реализовать для дальнейшего написания клиентского приложения. К ним относятся: авторизация пользователя (рисунок 12), получение информации о сервере (рисунок 13), получение списка устройств (рисунок 14), получение списка зарегистрированных пользователей (рисунок 15), возврат сохраненных данных с устройств (рисунок 16). [14]

### User authorization (auth\_req, auth\_resp)

#### Request message:

```
{
  "cmd": "auth_req",
  "login": string,           // case insensitive string
  "password": string        // original password string without any encoding
}
```

#### Response message:

```
{
  "cmd": "auth_resp",
  "status": boolean,
  "err_string"? : string    // [optional exist if "status" is false] – string code of error
  "token"? : string,        // [optional exist if "status" is true] – session string token (32 HEX symbols)
  "device_access"? : string, // [optional exist if "status" is true] – access level to devices (see below possible values)
  "consoleEnable": bool,    // [optional exist if "status" is true] – enable to connect to console subchart with debug information
  "command_list"? :         // [optional exist if "status" is true] – accessible commands (see below possible values)
  [
    "command_1",
    ...,
    "command_n"
  ],
  "rx_settings"? :          // [ optional exist if "status" is true] – setting of online receiving messages
  {
    "unsolicited": boolean, // online message is sending [true] or not sending [false - default]
    "direction": string,    // [optional absent if "unsolicited" is false] – direction of possible online messages (see below)
    "withMacCommands": boolean // [optional] – online message contains MAC commands
  }
}
```

Рисунок 12 – Функция для авторизации пользователя



## Server information (server\_info\_req, server\_info\_resp)

---

### Request message:

```
{
  "cmd": "server_info_req"
}
```

### Response message:

```
{
  "cmd": "server_info_resp",
  "status": boolean,           // status of command execution
  "err_string?": string,       // [optional – exist if "status" is false] error string code (see below
                                // description)
  "version?": string,          // [optional – exist if "status" is true] Version of server (e.g. "1.1.4")
  "time?":                     // [optional – exist if "status" is true] Time info
  {
    "utc": number,             // Server UTC timestamp (milliseconds from Linux epoch)
    "time_zone": string,        // Time zone (e.g. "UTC+01:00")
    "time_zone_utc_offset": number // Time offset from UTC in seconds (e.g. 3600)
  },
  "device_count?": 8           // [optional – exist if "status" is true] Device count info
  {
    "device_count_vega": number, // Count of registered devices of "Vega Absolute" company
    "device_count_other": number, // Count of registered devices of others manufacturers
    "available_device_count": number // Count of free remain devices
  }
}
```

### Example response message:

```
{
  "cmd": "server_info_resp",
  "status": true,
  "version": "1.2.0",
  "time":
  {
    "utc": 1513584871084,
    "time_zone": "UTC+07:00",
    "time_zone_utc_offset": 25200
  },
  "device_count":
  {
    "device_count_vega": 24,
    "device_count_other": 75,
    "available_device_count": 925
  }
}
```

Рисунок 13 – Функция для получения информации о сервере

## Get list of devices with attribute set (get\_device\_appdata\_req, get\_device\_appdata\_resp)

---

```
Request message:
{
  "cmd": "get_device_appdata_req",
  "keyword?":          //[optional] See possible values
  [
    string,...
  ]
  "select?:"          //[optional] Filter object
  {
    "appEui_list"?:    //[optional] List of corresponding AppEUI for request
    [
      "appEui_1",
      ...,
      "appEui_n"
    ]
  }
}
```

Рисунок 14 – Функция получения списка устройств

## Get list of registered users (get\_users\_req, get\_users\_resp)

---

### Request message:

```
{
  "cmd": "get_users_req",
  "keyword?":          //[optional] See below description
  [
    string, ...
  ]
}
```

### Possible string values of "keyword":

- "no\_command\_and\_devEui" – return list of user without "devEui\_list" and "command\_list"

### Response message:

```
{
  "cmd": "get_users_resp",
  "status": boolean,          // Main status of execution
  "err_string?": string,      //[optional exist if "status" is false] – string code of error
  "user_list":
  [
    {
      "login": string,
      "device_access": string, //Access level to devices (see below possible values). If "FULL",
                                // "devEui_list" would be ignored
      "consoleEnable": bool,   //Enable to connect to console subchart with debug information
      "devEui_list?":          //[optional absent if "no_command_and_devEui" is exist] List of
                                // DevEUI that is accessible for user
      [
        "devEui_1",
        ...,
        "devEui_n"
      ],
      "command_list?":         //[optional absent if "no_command_and_devEui" is exist] List of
                                // commands that is accessible for user
      [
        "command_1",
        ...,
        "command_n"
      ],
      "rx_settings?":          //[optional absent if "no_command_and_devEui" is exist] – setting of
                                // online receiving messages
      {
        "unsolicited": boolean, //online message is sending [true] or not sending [false - default]
        "direction?": string,   //[optional absent if "unsolicited" is false] – direction of possible
                                // online messages (see below)
        "withMacCommands?":boolean //[optional] – online message contains MAC commands
      }
    }, ...
  ]
}
```

Рисунок 15 – Функция получения списка зарегистрированных пользователей

## Return saved data from device (get\_data\_req, get\_data\_resp)

---

```
Request message:
{
  "cmd": "get_data_req",
  "devEui": string,
  "select"?: string,           //[optional] Extra optional for searching
  {
    "date_from"?: integer,     //[optional] server UTC timestamp as number (milliseconds from Linux epoch)
    "date_to"?: integer,       //[optional] server UTC timestamp as number (milliseconds from Linux epoch)
    "begin_index"?: integer,    //[optional] begin index of data list [default = 0]
    "limit"?: integer,          //[optional] limit of response data list [default = 1000]
    "port"?: integer,           //[optional] select data with noted port
    "direction"?: string,       //[optional] direction of message transition (see below description)
    "withMacCommands"?: boolean//[optional] add MAC commands to response
  }
}

Response message:
{
  "cmd": "get_data_resp",
  "status": boolean,           // Status of execution of command (global status)
  "err_string"?: string,       //[optional] If "status" = false, contains error description (see below description)
  "devEui": string,
  "appEui": string,
  "direction"?: string,        //[optional - exist if "status" = true]
  "totalNum"?: integer,         //[optional - exist if "status" = true] Total existing number of data corresponding type
  "data_list"?:                //[optional - exist if "status" = true] Data, transmitted by device
  [
    {
      "ts": integer,           // Server UTC receiving timestamp (milliseconds from Linux epoch)
      "gatewayId": string,     // Gateway IDs that receive data from device4
      "ack": boolean,          // Acknowledgement flag as set by device
      "fcnt": integer,          // Frame counter, a 32-bit number (uplink or downlink based on "direction" value)
      "port": integer,          // Port (if = 0, use JOIN operations or MAC-commands only)
      "data": string,           // Decrypted data payload
      "macData"?: string,       //[optional- exist if "withMacCommands" true and MAC command is present] MAC command data from device
      "freq": integer,          // Radio frequency at which the frame was received/transmitted, in Hz
      "dr": string,             // Spreading factor, bandwidth and coding rate "SF12 BW125 4/5"
      "rssi": integer,          //[optional - exist if packet direction "UPLOAD"] Frame rssi, in dBm, as integer number
      "snr": float,             //[optional - exist if packet direction "UPLOAD"] Frame snr, in dB
      "type": string,           // Type of packet (see below description). May contains several types joined via "+"
      "packetStatus"?: string   //[optional - exist if packet direction "UPLOAD"] Status of downlink message only (see below description)
    }, ...
  ]
}
```

Рисунок 16 – Функция возврата сохраненных данных с устройства

Все представленные выше функции были реализованы в программе.



```

9    ## Authorization on VEGA server
10   autreq = {"cmd": "auth_req", # Don't change!
11            "login": "root", # Login name
12            "password": "123456" # password
13   }

```

Рисунок 18 – Часть кода для авторизации пользователя

```

31  ##Connection to VEGA server
32  ws = create_connection("ws://192.168.17.106:8002/") # Start connection. IP Address:Port VEGA server
33  ws.send(json.dumps(autreq)) # Get Authorization command
34  autresp = ws.recv() #Status (responce) of execute command
35
36  print(autresp)
37  print("=====")

```

Рисунок 19 – Часть кода для авторизации пользователя

В качестве IP-адреса здесь указывается IP-адрес компьютера, на котором установлен IOT Vega Server.

```

{"cmd": "auth_resp", "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"], "device_access": "FULL", "rx_settings": {"direction": "ALL", "unsolicited": 1, "withMacCommands": 1}, "status": true, "token": "0747896cd916ad849332ae5f979f1f68"}

```

Рисунок 20 – Результат авторизации пользователя

При успешной авторизации пользователя в консоль выводятся те функции, которые доступны для данного пользователя, т.к. на рисунке 17 вход произведен от имени супер-пользователя, ему доступны все возможные на сервере функции.

На рисунках 21-22 представлены части кода, которые отвечают за получение информации о сервере, на рисунке 23 представлен ответ.

```

14  ## Get information from connected server
15  srvinfo = {"cmd": "server_info_req"} # Don't change!

```

Рисунок 21 – Часть кода для получения информации о сервере

```

39  ## Get server info
40  ws.send(json.dumps(srvinfo))                # Get command
41  srvinfresp = ws.recv()
42  infresp_dict = json.loads(srvinfresp)
43  time_serv_now = infresp_dict["time"]["utc"]/1000
44  local_timezone = tzlocal.get_localzone()
45  serv_time = datetime.fromtimestamp(time_serv_now, local_timezone)
46
47  print(srvinfresp)
48  print(serv_time.strftime("%Y-%m-%d %H:%M:%S"))
49  print("=====")

```

Рисунок 22 – Часть кода для получения информации о сервере

```

=====
{"cmd":"server_info_resp","device_count":{"available_device_count":998,"device_count_other":2,"device_
count_vega":0},"status":true,"time":{"time_zone":"Томск (зима)","time_zone_utc_offset":0,"utc":1654065
117235},"version":"1.2.1 [WIN]"}
2022-06-01 13:31:57

```

Рисунок 23 – Ответ от сервера

Проверить то, что данные выводятся верно можно с помощью веб-приложения IoT Vega Admin Tool (рисунок 24).

Server status	
Time	01.06.2022 06:33:39
Time zone	Томск (зима)
Version	1.2.1 [WIN]
Vega devices	0
Third-party devices	2
Third-party devices available	998
Application statistics	
Gateways	1
Devices	2
Users	5

Рисунок 23 – Информация в IoT Vega Admin Tool

На рисунках 25-26 представлены части кода, которые реализуют получение списка подключенных оконечных устройств, на рисунке 27 представлен ответ.

```

17  ## Get device list w/attributes
18  devalist = {"cmd":"get_device_appdata_req"}

```

Рисунок 25 – Часть кода для получения списка оконечных устройств

```

51  ## Get dev list w/attributes
52  ws.send(json.dumps(devalist))
53  devalistresp = ws.recv()
54
55  print(devalistresp)
56  print("=====")

```

Рисунок 26 – Часть кода для получения списка оконечных устройств

```

{"cmd": "get_device_appdata_resp", "devices_list": [{"appEui": "AC1F09FFF8680811", "devEui": "AC1F09FFFE0152FF", "devName": "test_add"}, {"appEui": "AC1F09FFF8680811", "devEui": "AC1F09FFFE015304", "devName": "test_dev", "device_type": "1", "group": "", "version": "0"}], "status": true}

```

Рисунок 27 – Список подключенных оконечных устройств

Для проверки можно так же обратиться к веб-приложению IoT Vega Admin Tool (рисунок 28).

Device name	DevEUI	Last connection ^
test_add	AC1F09FFFE0152FF	-
test_dev	AC1F09FFFE015304	19.05.2022 16:13:12

Рисунок 28 – Список оконечных устройств в IoT Vega Admin Tool

На рисунках 29-30 представлены части кода, которые реализуют получение списка зарегистрированных учетных записей, на рисунке 31 представлен ответ от сервера.

```

25  ##Get reg users
26  reguser = {"cmd": "get_users_req"} # Don't change!
27  ~

```

Рисунок 29 – Часть кода для получения списка зарегистрированных учетных записей

```

58  ## Get registered users
59  ws.send(json.dumps(reguser))
60  reguserresponse = ws.recv()
61
62  print(reguserresponse)
63  print("=====")

```

Рисунок 30 – Часть кода для получения списка зарегистрированных учетных записей



```

-----
{"cmd":"get_users_resp","status":true,"user_list":[{"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","server_info","tx"],"consoleEnable":0,"devEui_list":[],"device_access":"FULL","login":"admin","rx_settings":{"direction":"ALL","unsolicited":1,"withMacCommands":1}},{command_list":["get_device_appdata","get_data","manage_device_appdata","delete_device_appdata"],"consoleEnable":0,"devEui_list":[],"device_access":"FULL","login":"user","rx_settings":{"direction":"UP LINK","unsolicited":1,"withMacCommands":0}},{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","server_info","tx"],"consoleEnable":0,"devEui_list":[],"device_access":"SELECTED","login":"temperature_admin","rx_settings":{"direction":"ALL","unsolicited":0,"withMacCommands":1}},{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","server_info","tx"],"consoleEnable":0,"devEui_list":["SELECTED","login":"humidity_admin","rx_settings":{"direction":"ALL","unsolicited":1,"withMacCommands":1}},{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","server_info","tx"],"consoleEnable":0,"devEui_list":["SELECTED","login":"pressure_admin","rx_settings":{"direction":"ALL","unsolicited":1,"withMacCommands":1}}]}

```

Рисунок 31 – Список зарегистрированных учетных записей

Проверить правильность выданных данных можно в том же IoT Vega Admin Tool (рисунок 32).

Home Devices Gateways Users Console Exit					
<input type="text"/>		CONNECTED USERS		<a href="#">+ Add new user</a>	
Login	Device access	Permission			
admin	FULL	Custom	⚙	⚙	
humidity_admin	SELECTED	Custom	⚙	⚙	
pressure_admin	SELECTED	Custom	⚙	⚙	
temperature_admin	SELECTED	Custom	⚙	⚙	
user	FULL	Custom	⚙	⚙	

Рисунок 32 – Список зарегистрированных учетных записей в IoT Vega Admin Tool

На рисунках 33-34 представлены части кода, которые отвечают за получение сохраненных данных с устройства, ответ от сервера представлен на рисунке 35.

```

20  ##Get data from devices
21  datadev = {"cmd":"get_data_req",
22            "devEui":"AC1F09FFFE015304"
23            } # Don't change!

```

Рисунок 33 – Часть кода для получения сохраненных данных с устройства

```

65 while(True):
66     ## Get save data from devices
67     ws.send(json.dumps(datadev))          # Get command
68     datadevresp = ws.recv()
69     data_dict = json.loads(datadevresp)
70     keys = data_dict.keys()
71     keys1 = str(keys)
72     if keys1 == "dict_keys(['appEui', 'cmd', 'data_list', 'devEui', 'direction', 'status', 'totalNum'])":
73         data = [{"devEui": data_dict["devEui"]}, {"data": data_dict["data_list"][0]["data"]}, {"type": da
74         print(data)
75         realtime = data_dict["data_list"][0]["ts"]/1000
76         packet_time = datetime.fromtimestamp(realtime, local_timezone)
77         print(packet_time.strftime("%Y-%m-%d %H:%M:%S"))
78     else:
79         print(datadevresp)
80     print("=====")
81     time.sleep(5)

```

Рисунок 34 – Часть кода для получения сохраненных данных с устройства

В части кода (рисунок 34) реализован цикл с задержкой для того, чтобы на постоянной основе через каждые 5 секунд получать актуальные данные, отправляемые с окончного устройства на сервер.

```

-----
[{'devEui': 'AC1F09FFFE015304'}, {'data': '28'}, {'type': 'UNCONF_UP'}]
2022-06-01 13:36:16

```

Рисунок 35 – Сохраненные данные

Для того, чтобы проверить правильность полученных данных необходимо снова обратиться к веб-приложению IoT Vega Admin Tool (рисунок 36).

◀ Back

📅
June 1, 2022 - June 1, 2022

Request

Device name :

test\_dev

DevEUI :

AC1F09FFFE015304

Date range :

01.06.2022 - 01.06.2022

Number of packets :

190

Average SNR :

8.83

Average RSSI :

-77.85

🔍
➡ Send data
🛑
🔄

Date	Type	Data
01.06.2022 13:36:16	UNCONF_UP	28

Рисунок 36 – Сохраненные данные в IoT Vega Admin Tool

Как видно параметр `devEui` совпадает, а так же время, когда был получен пакет, и поле “Data”, которое на данный момент показывает температуру. В дальнейшем все эти функции планируется использовать для написания собственного веб-приложения. Полный листинг кода представлен в приложении А.

#### 4.4 Сбор микроклиматических параметров

##### *Выбор и описание микроконтроллера*

Для проекта использовался модуль микроконтроллера STM32 — это наиболее популярное и широко используемое за счет низкой стоимости семейство микроконтроллеров, используемых и в реальной жизни, а не только в обучении. Программирование происходило на языке C++.

Из данного семейства была выбрана отладочная плата NUCLEO-F103RB. На рисунке 37 представлено фото данного микроконтроллера.



Рисунок 37 – Отладочная плата NUCLEO-F103RB

Это стандартная учебная плата от ST на настоящий момент. Рекомендуется использовать именно её. В магазинах есть большой выбор

таких плат с разными характеристиками микроконтроллера, некоторые из них неудобны из-за малого количества памяти. Для реализации проекта памяти в данной модели достаточно.

Положительные стороны выбора данного микроконтроллера, следующие:

- Есть встроенный USB-UART (интерфейс для коммуникации платы и компьютера через консольный ввод/вывод).
- Легко доступна в продаже (как в плане цены, так и количества в продаже).
- Очень удобно перепрошивать: определяется как флэшка, поэтому на нее достаточно просто "бросить" файл, и это работает и в Linux, и в Windows.
- Совместима с Mbed без дополнительных настроек, изначально.

Минусы данной платы, следующие:

- Много различных версий с разными характеристиками (легко запутаться).
- Устаревший разъём MiniUSB (и одновременно преимущество – намного крепче использующегося во многих современных платах micro-USB).

Ниже приведены основные технические характеристики данной платы:  
[15]

- Микроконтроллер STM32 в корпусе LQFP64.
- 1 пользовательский светодиод.
- Пользовательская кнопка и кнопка сброса.
- Кварцевый генератор частотой 32,768 кГц
- Разъемы платы: разъемы расширения ARDUINO Uno V3 и разъемы расширения morpho для полного доступа ко всем входам/выходам STM32.
- Гибкие варианты питания: ST-LINK, USB VBUS и возможность питания от внешних источников.

- Встроенный отладчик/программатор ST-LINK с возможностью повторного перечисления USB: накопитель, виртуальный COM-порт и порт отладки.
- Полные библиотеки свободного ПО и примеры, доступные в пакете MCU STM32Cube.
- Поддержка широкого выбора интегрированных сред разработки (IDE), включая IAR Embedded Workbench, MDK-ARM и STM32CubeIDE.
- Внешние SMPS для генерации логического питания Vcore.
- 24 МГц HSE.
- Разъемы платы: для внешних экспериментов с SMPS выделенный разъем USB Micro-AB или Mini-AB для разъема отладки ST-LINK/MIPI.
- Совместимость с Arm Mbed.

Основное назначение микроконтроллера – командовать подключенными к ней различными устройствами через его выводы. На рисунке 38 представлены подписанные выходы платы.

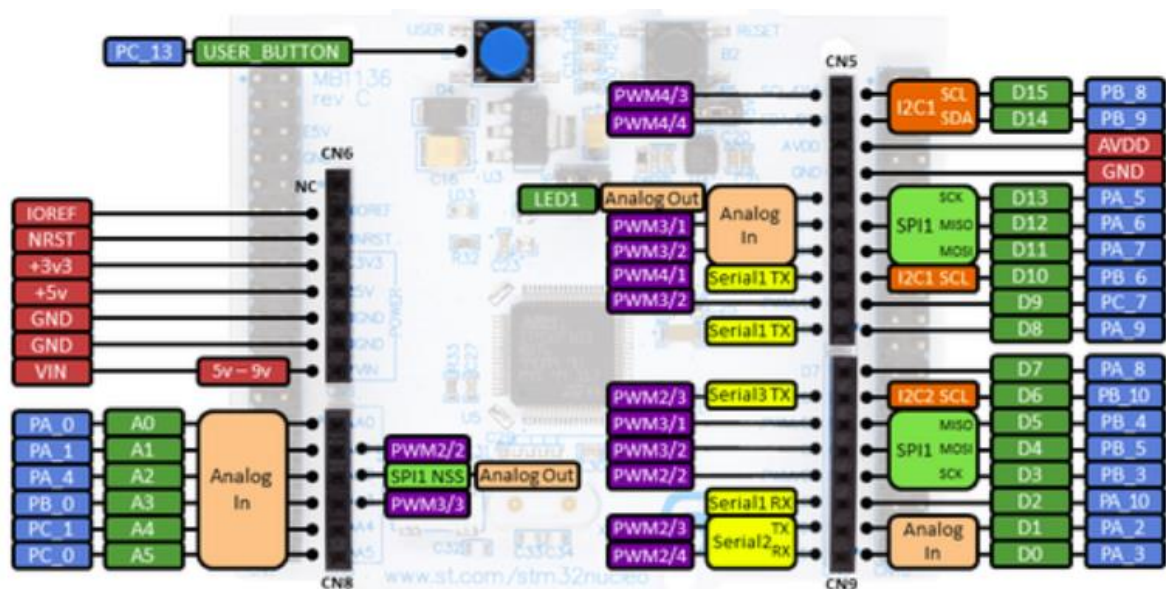


Рисунок 38 – Выводы платы микроконтроллера

### *Описание сенсора*

Согласно задаче проекта, необходимо было выбрать датчики, которые специализируются на снятии микроклиматических параметров, а именно температуры, влажности и давления. По причине того, что в распоряжении

уже были сенсоры от компании *unwireddevices* – UMDK-THP – был выбран датчик давления, влажности и температуры воздуха на базе высокоточного цифрового сенсора Bosch BME280. Также был вариант использовать сенсор BMP280, но за счет того, что он не измеряет влажность (в перспективе развития проекта планируется снимать данные по 3-м параметрам), BMP был отклонен. Схема датчика показана на рисунке 39. [16]

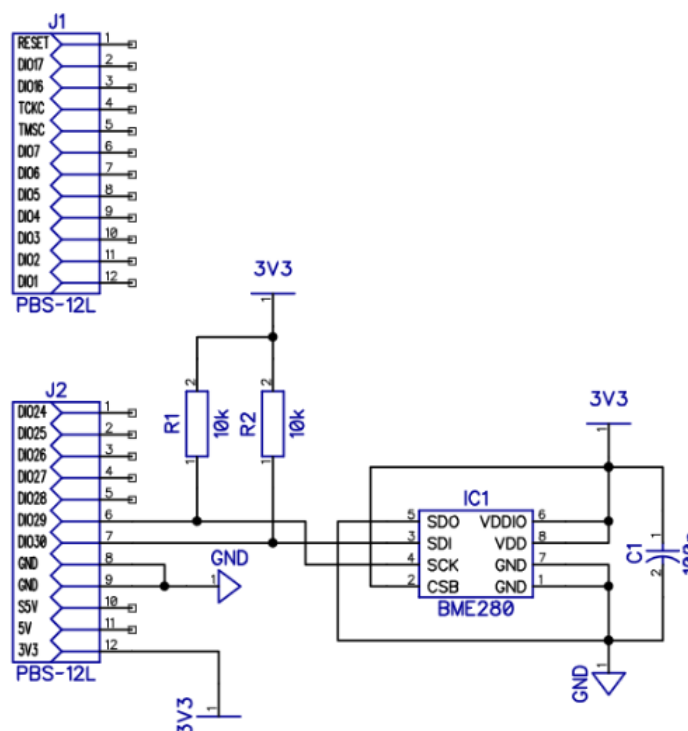


Рисунок 39 – Схема сенсора

В данном сенсоре используется шина I2C, соответственно, совместим со всеми I2C-модулями (это может быть полезно в случае, если устройство в будущем будет использовать другие датчики с таким интерфейсом). Как видно на схеме, используются GPIO 29 и GPIO 30 (на схеме – DIO29 и DIO30), идущие в сам сенсор на входы SDI и SCK. [17]

Фотография самого сенсора представлена на рисунке 40.



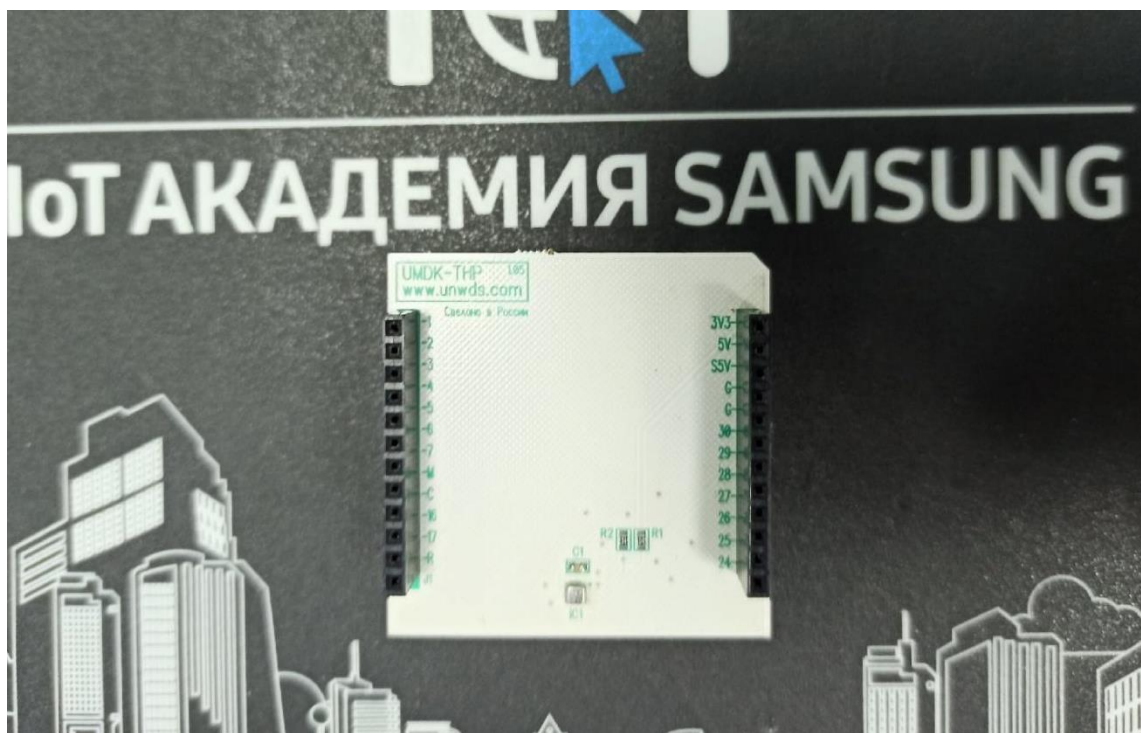


Рисунок 40 – Фотография сенсора UMDK-THP

#### *Описание LoRa-модуля*

В качестве приемо-передатчика был взят имеющийся в наличии RAK811, показанный на рисунке 41.

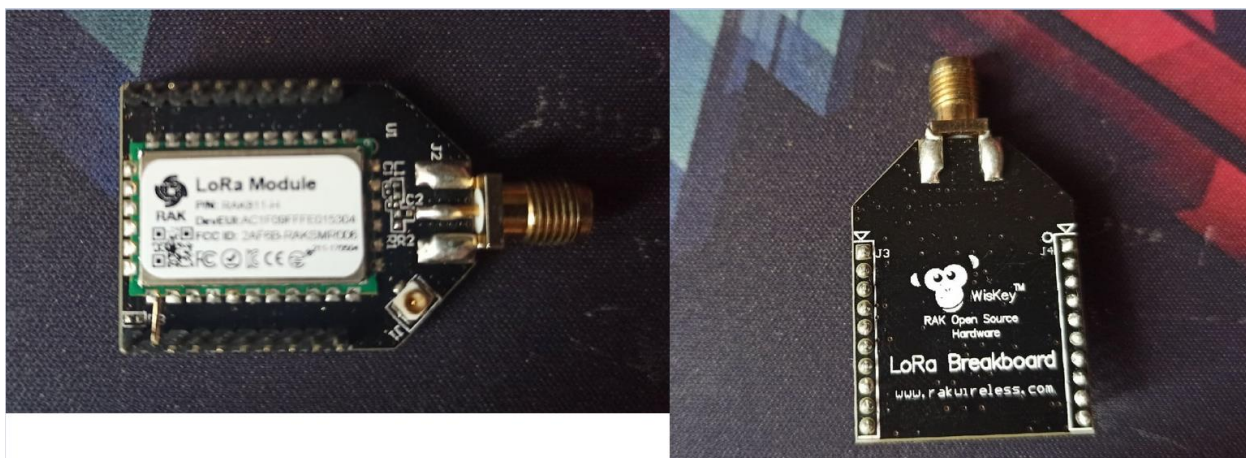


Рисунок 41 – Модуль LoRa

Стоял выбор между ним и Unwired Range (UNWR) более старого образца с немного более худшими характеристиками.

Характеристики RAK811 следующие:

- Поддержка многих стандартов, в том числе интересующий EU868.

- Дальность связи до 2 (городская среда) или до 15 км (открытый участок).
- Поддержка классов А и С протокола LoRaWAN.
- Поддерживаемые интерфейсы: UART, I2C, GPIO, ADC.
- Ток потребления 5.5 мА.
- Мощность передатчика 20 дБм.

Из-за того, что форм-фактор передатчика не соответствует стандарту, по которому его нужно подключать к микроконтроллеру, необходимо было использовать адаптер XBee USB Adapter, показанный на рисунке 42.

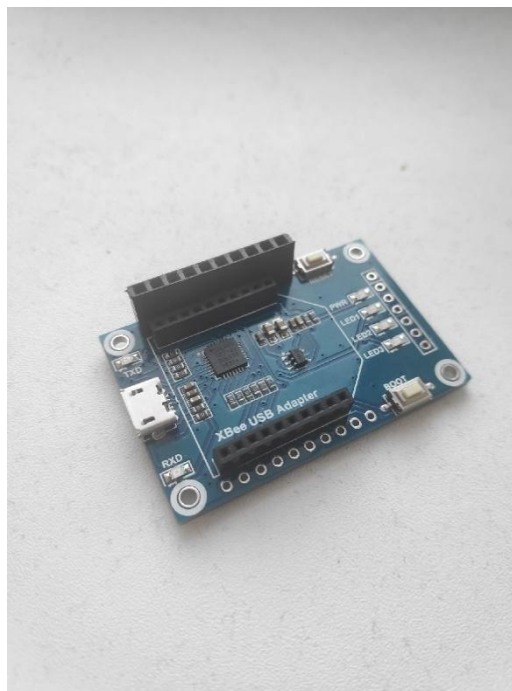


Рисунок 42 - XBee USB Adapter

Сделав небольшую доработку в виде впайки разъемов для проводов подключения в адаптер, подключение приемо-передатчика к плате стало возможным.

#### *Схема подключения*

Схема подключения устройств к микроконтроллеру представлена на рисунке 43.



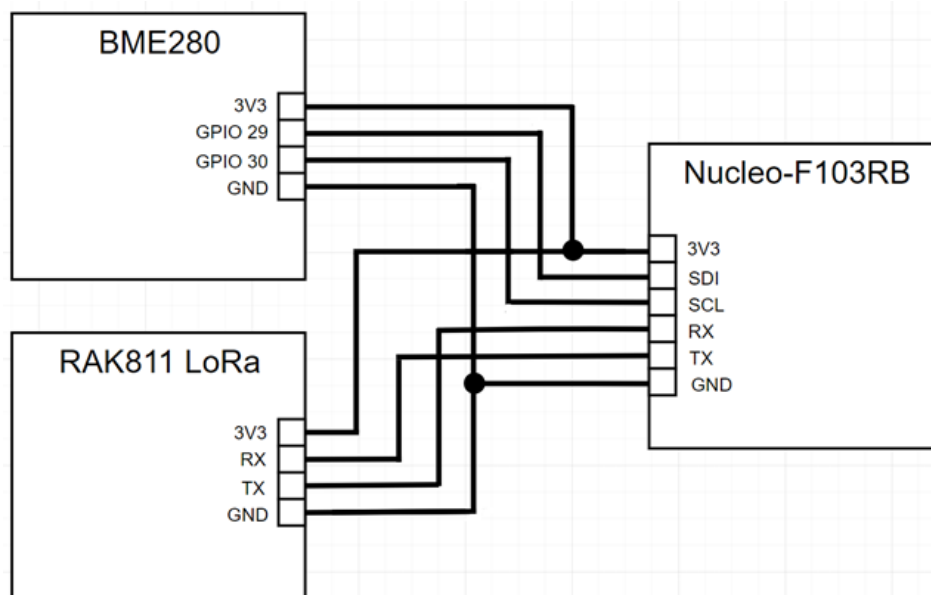


Рисунок 43 – Схема подключения

Фотография собранного прототипа представлена на рисунке 44.

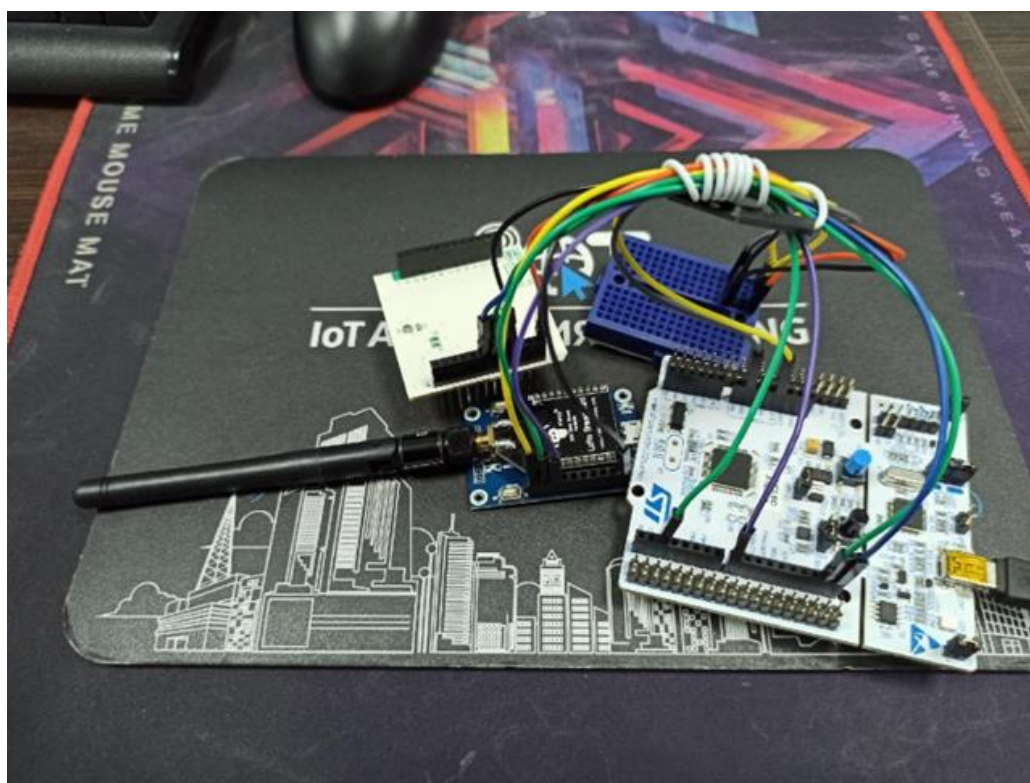


Рисунок 44 – Фотография прототипа устройства по сбору микроклиматических параметров

### *Создание прошивки*

Основная работа производилась над созданием прошивки микроконтроллера, в которую бы были включены такие компоненты, как

обработка собранных с сенсора данных, генерация команд для конечного устройства LoRa по его инициализации и подключению к базовой станции, отправка данных на сервер, а также реализация режимов сна у микроконтроллера и радио-модуля (об этом будет говорится в следующем подразделе). На рисунке 45 представлена блок-схема работы прошивки.

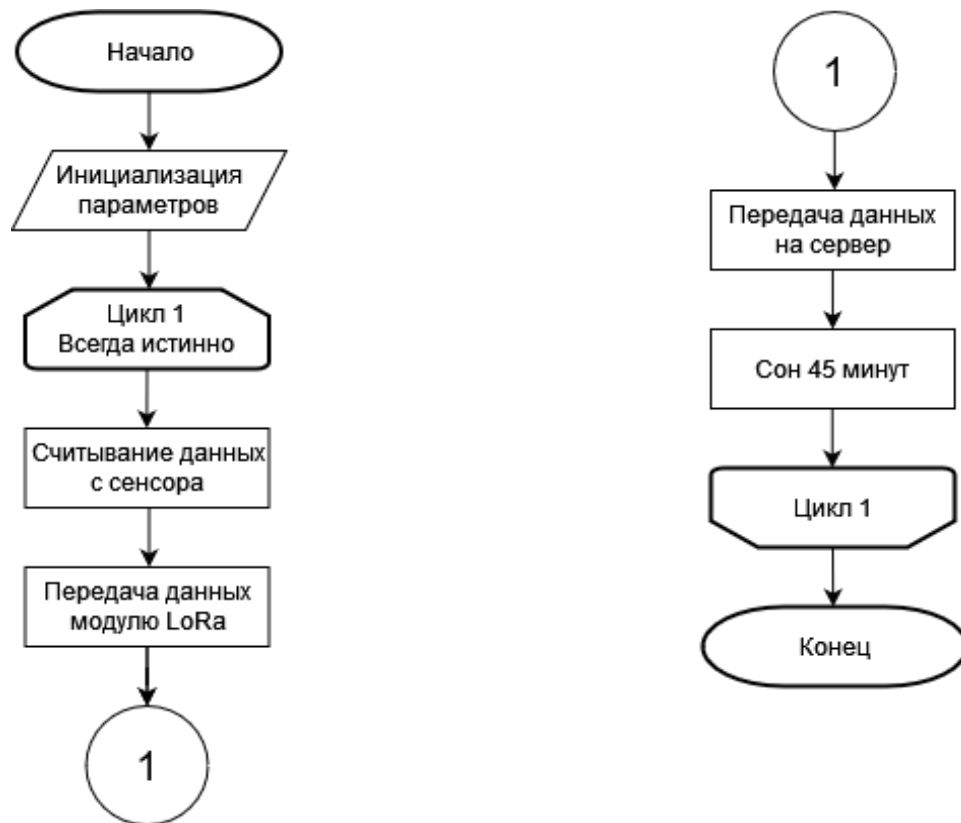


Рисунок 45 – Блок-схема кода прошивки

На рисунке 46 представлено то, как в созданной прошивке происходит преобразование полученного значения температуры (аналогично для давления и влажности) в символьный вид, готовый для передачи по сети.

```

92     char* temp;
93     int k = sensor.getTemperature();
94     temp = (char *)malloc(10 * sizeof(char));
95     int v = 0; //количество цифр в числе n
96     //разбиваем на отдельные символы число n
97     while (k > 9)
98     {
99         temp[v++] = (k % 10) + '0';
100        k = k / 10;
101    }
102    temp[v++] = k + '0';
103    temp[v] = '\0';
104    char t;
105    //инвертируем массив символов
106    for (int i = 0; i < v / 2; i++)
107    {
108        t = temp[i];
109        temp[i] = temp[v - 1 - i];
110        temp[v - 1 - i] = t;
111    }

```

Рисунок 46 – Подготовка данных к отправке по сети

Полный код прошивки представлен в приложении Б.

### *Активация устройства*

Далее был решен вопрос, связанный с инициализацией конечного устройства, а именно - проведение процедуры активации. Существует два варианта активации [18]:

- ОТАА, Over-The-Air Activation (требуется пройти процедуру присоединения (join procedure), во время которой вырабатываются сессионные ключи шифрования и адрес DevAddr).
- АВР, Activation by Personalization (не требуется проходить процедуру присоединения, ключи шифрования и адрес DevAddr записываются в устройство заранее (персонализация устройства)).

В результате был установлен первый вариант (активация по воздуху).

Во время присоединения к серверу вырабатываются сессионные ключи шифрования и локальный адрес. После выполнения этой процедуры устройство содержит:

- DevAddr – локальный адрес в данной сети;
- Network Session Key (NwkSKey) – сетевой сессионный ключ. Нужен для проверки кода целостности сообщений при обмене между устройством и сервером, а также шифрования сообщений MAC-уровня;
- Application Session Key (AppSKey) – сессионный ключ. Нужен для шифрования данных на уровне приложения.

То есть активация происходит каждый раз, когда ключи NwkSKey, AppSKey и адрес DevAddr отсутствуют. И перед тем, как присоединиться, устройству должны быть присвоены три идентификатора-ключа:

- End-device identifier (DevEUI) — уникальный идентификатор устройства.
- Application identifier (AppEUI) – идентификатор приложения.
- Application key (AppKey) — ключ, который используется в процессе присоединения к сети для получения сессионных ключей NwkSKey и AppSKey.

Три данных идентификатора указываются на сервере, вследствие чего можно отслеживать работу и переданные данные.

На схеме на рисунке 47 показана схема, каким образом, используя упомянутые ключи, происходит соединения конечного устройства с сервером.

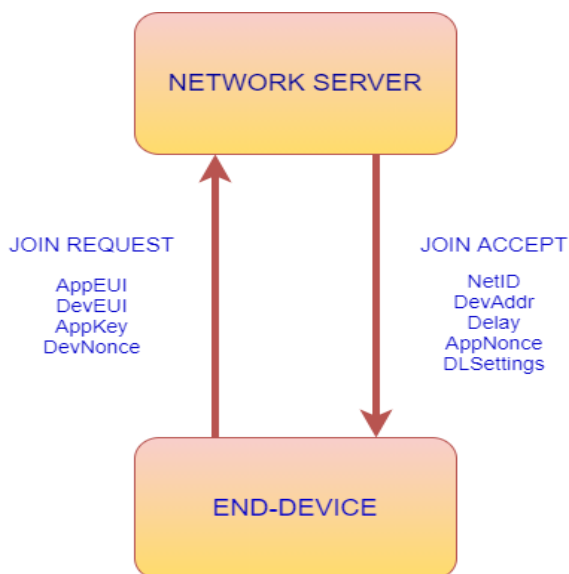


Рисунок 47 – Процедура присоединения

Второй шаг на данном этапе - подключение и переподключение устройства к серверу. Данное действие выполняется с помощью AT-команды “at+join”. С учетом того, что сообщения (параметры климата) отправляются раз в 30 минут, одного переподключения в сутки будет достаточно для стабильной работы устройства.

### *Отправка данных*

Реализация была выполнена с использованием базового протокола LoRaWAN с отправкой сообщений через AT-команды. Для этого была использована команда “”at+send=lora:1:{данные}”. В данные помещаются значения температуры, снятые датчиком и обработанные микроконтроллером. На сервере можно отследить полученные данные.

На рисунке 48 представлена часть кода, которой реализуется отправка данных на сервер (без учета составления команды AT и задержек между отправками).

```
pc.baud(115200);
dev.baud(115200);
for(int i = 0; i < strlen(command_SEND); i++)
{
    dev.putc(command_SEND[i]);
    pc.printf("%c", command_SEND[i]);
}
dev.putc('\n');
dev.putc('\r');

pc.attach(&pc_recv, Serial::RxIrq);
dev.attach(&dev_recv, Serial::RxIrq);
```

Рисунок 48 – Часть кода по отправке данных на сервер

### *Режим сна*

Последний шаг, связанный с реализацией одной из главных задач - решение вопроса об энергосбережении, то есть настройка режимов сна устройства. Подразумевалась настройка режимов сна для самого микроконтроллера и конечного устройства или радио-модуля.

Введение в сон RAK811 была задачей достаточно простой - задание в прошивке периода отправки AT-команды “at+set\_config=device:sleep:X”, где X - 1, если вводим в сон. После ввода в этот режим потребление модуля снижается до 7-14 мкА (5.5 мА в режиме приёма сигнала и 30 мА в режиме передачи сигнала). Был реализован вариант с вводом в режим через команду. Вывод из сна осуществляется отправкой простой команды “at” на модуль, вследствие чего он “просыпается”. Ввод микроконтроллера в сон осуществлялся через использование специальной команды `ThisThread::sleep_for(X)` (X - время сна в секундах).

При вызове данной функции происходят следующие действия с микроконтроллером:

- Сначала отключается бит `Tickint`, который отвечает за прерывания. Это происходит потому, что режим сна будет сбрасываться при первом внутреннем прерывании. Данный бит будет включен по прошествии времени сна.

- После этого запускается сам режим сна, из которого выход будет производиться внутренним прерыванием. При переходе в режим останавливается только тактирование, поэтому при выходе из него код продолжится с того места, на котором он остановился считываться, соответственно работа по передаче данных не будет прервана совсем.

- После окончания времени сна срабатывает “будильник”, после срабатывания которого восстанавливается тактирование и работа МК продолжается в обычном режиме.

## 5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПО ИСПОЛЬЗОВАНИЮ СИСТЕМЫ

### 5.1. Настройка и использование базовой станции и сервера

Для начала необходимо подключить базовую станцию к компьютеру с помощью провода USB-mini USB, после чего произвести инициализацию базовой станции (далее БС) на персональном компьютере посредством COM-порта, используя терминальную программу PuTTY. Далее в той же программы нужно получить необходимую информацию - IP-адрес БС, который необходим для её настройки через веб-интерфейс. (рисунки 49-50).

```
root@am335x-evm:~/bs-dashboard/manager# ifconfig
eth0      Link encap:Ethernet  HWaddr 34:03:DE:7B:72:80
          inet addr:192.168.1.228  Bcast:192.168.1.255  Mask:255.255.254.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1478151 errors:0 dropped:614 overruns:0 frame:0
          TX packets:103187 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109611064 (104.5 MiB)  TX bytes:23971656 (22.8 MiB)
          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:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@am335x-evm:~/bs-dashboard/manager#
```

Рисунок 49 – Инициализация БС на ПК

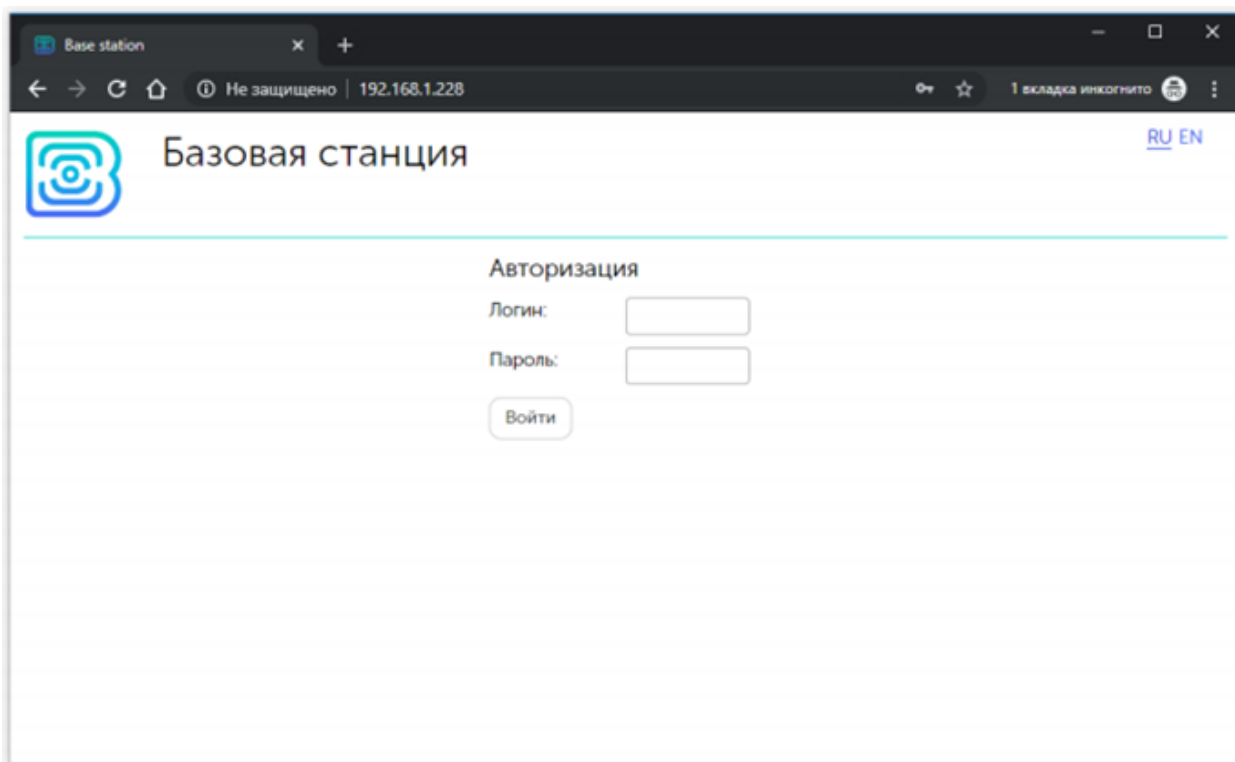


Рисунок 50 – Веб-интерфейс для настройки БС

Затем необходимо выполнить настройку основных параметров БС через веб-интерфейс (рисунок 51), описанных ниже, все остальные же параметры оставить по умолчанию:

- настройки подключения к серверу LoRaWAN - адрес будущего сервера, а также граница доступных портов для подключения внешних устройств.
- логи LoRa, здесь задается полное логирование всех событий.



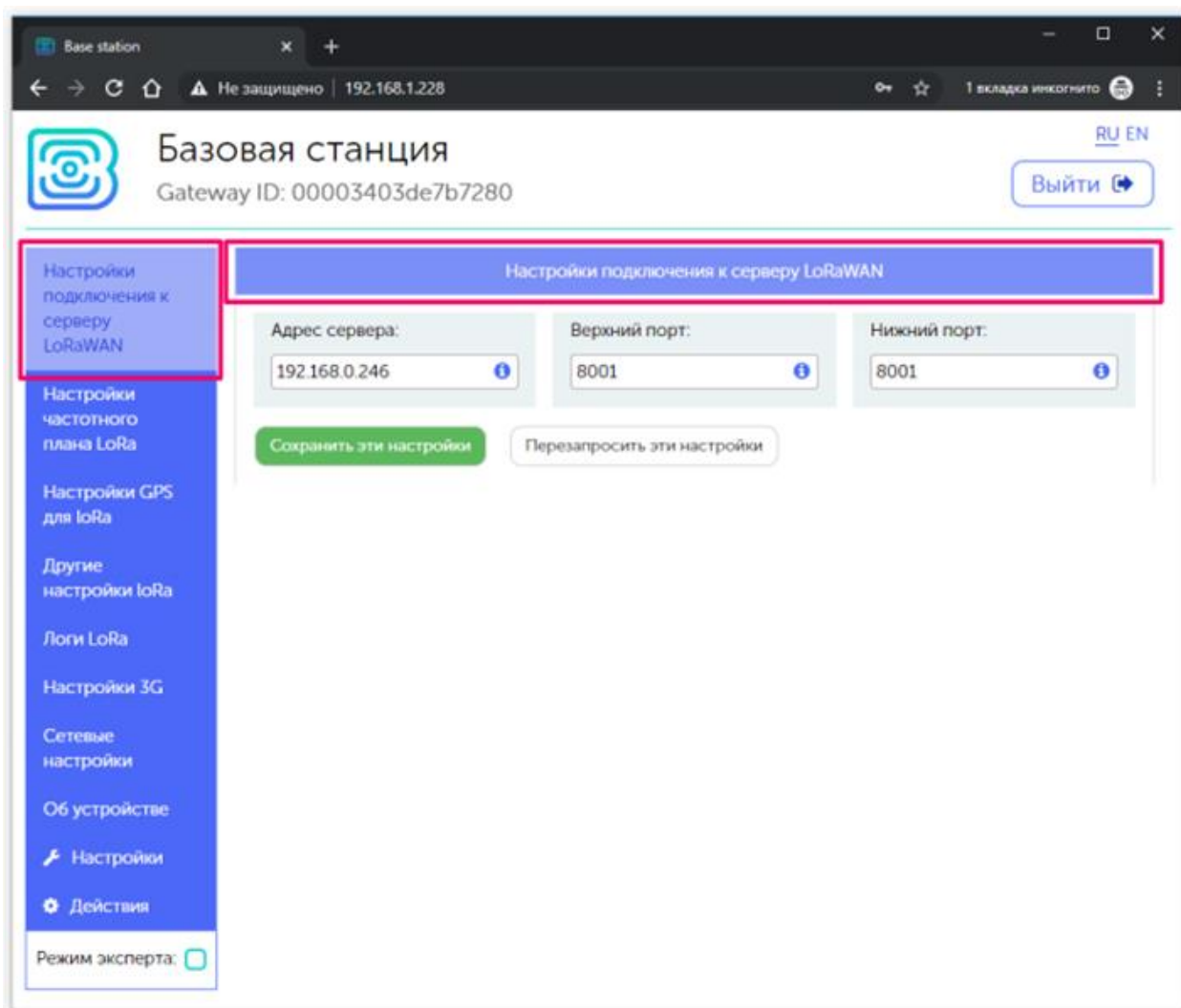
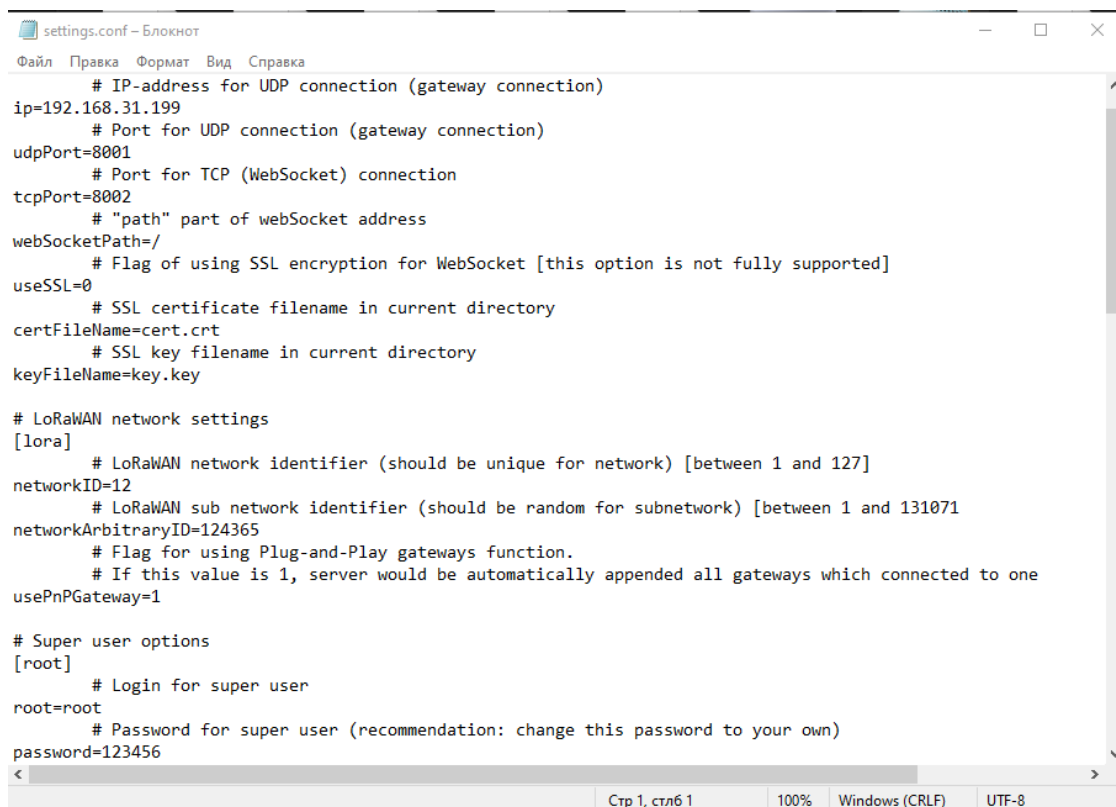


Рисунок 51 – Настройки подключения к серверу LoRaWAN

После был поставлен сетевой сервер — IOT Vega Server - это инструмент для организации сетей стандарта LoRaWAN любого масштаба, который в данной работе предназначен для управления опорной сетью базовых станций, работающих под управлением ПО Packet forwarder от компании Semtech, приема данных с оконечных устройств и передачи их внешним приложениям, а также передачи данных от внешних приложений на LoRaWAN® устройства. Полная информация о нём находится на сайте производителя.

Перед тем, как выполнить запуск непосредственно самого сервера необходимо выполнить первичную настройку, а именно изменить несколько файл конфигурации “settings.conf” (рисунок 52).



```
settings.conf – Блокнот
Файл  Правка  Формат  Вид  Справка

# IP-address for UDP connection (gateway connection)
ip=192.168.31.199
# 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

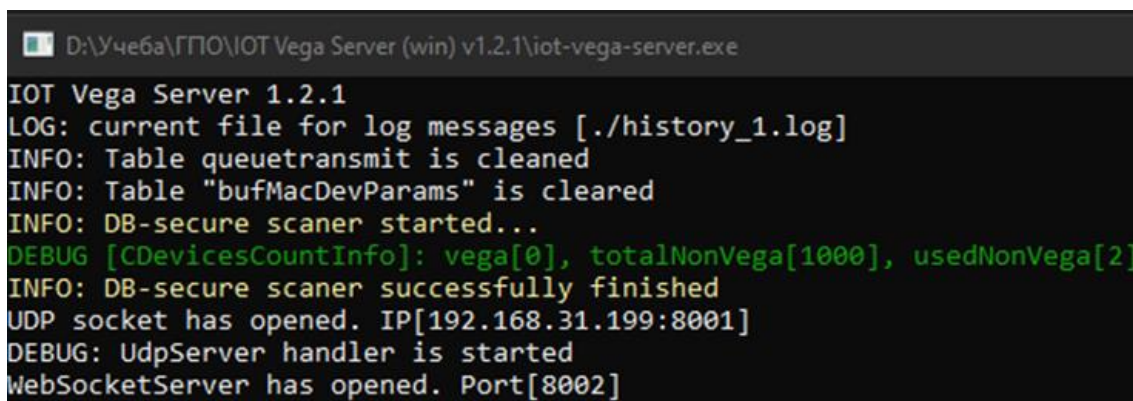
# Super user options
[root]
# Login for super user
root=root
# Password for super user (recommendation: change this password to your own)
password=123456

Стр 1, столб 1    100%    Windows (CRLF)    UTF-8
```

Рисунок 52 – Изменение файла конфигурации

В данном файле необходимо изменить строки “ip” (отвечает за IP-адрес компьютера, на котором будет расположен сервер), а также строку “password”, которая отвечает за пароль суперпользователя - он понадобится в дальнейшем при обращении к серверу через приложение IOT Vega AdminTool.

После первичной настройки нужно запустить непосредственно сам сервер. (рисунок 53)



```
D:\Учеба\ГПО\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[2]
INFO: DB-secure scanner successfully finished
UDP socket has opened. IP[192.168.31.199:8001]
DEBUG: UdpServer handler is started
WebSocketServer has opened. Port[8002]
```

Рисунок 53 – Запуск сервера

В случае, если сервер работает корректно будут выведены строки “UDP socket has opened” и “WebSocketServer has opened”.

Для удобного взаимодействия с сервером нужно использовать приложение IOT Vega Admin Tool. Оно открывает перед администратором сервера широкие возможности по управлению сетью LoRaWAN, например, с ним вы можете добавлять в сеть новые оконечные устройства, просматривать карту сети, контролировать базовые станции, а также управлять правами пользователей.

Перед тем, как начать работать с приложением необходимо настроить файл конфигурации (рисунок 54), а именно изменить постоянный IP-адрес сервера и номер порта в соответствии с ранее измененным файлом конфигурации.

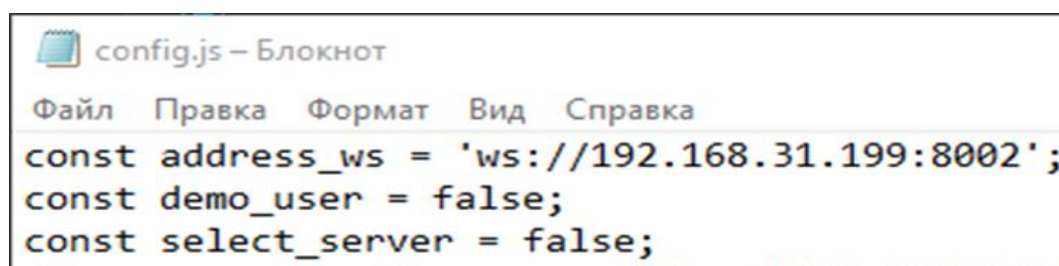


Рисунок 54 – Настройка файла конфигурации веб-приложения Admin Tool

Далее нужно произвести вход в веб-приложение, где используются логин и пароль суперпользователя, прописанные в файле конфигурации сервера. (рисунок 55)

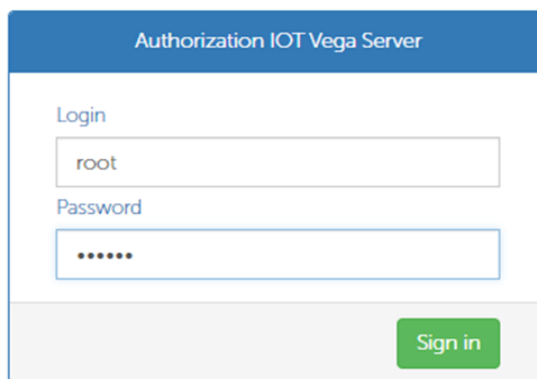


Рисунок 55 – Окно авторизации в приложение IOT Vega Admin Tool

После авторизации в веб-приложении сразу же открывается вкладка “Home”, на которой можно увидеть информацию о времени, часовом поясе и версии сервера. Также на данной вкладке можно увидеть количество БС (gateways), количество подключенных оконечных устройств (Devices) и количество пользователей (users). В случае, если какое-либо устройство (БС или оконечное устройство) поддерживает GPS, то оно будет представлено на карте. (рисунок 56)

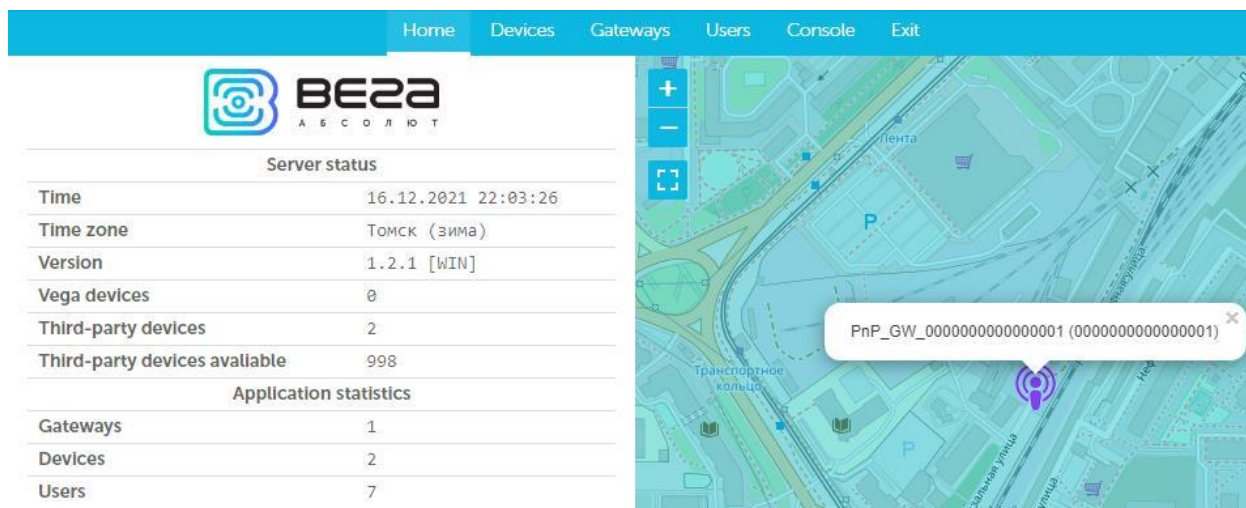


Рисунок 56 – Вкладка “Home” веб-приложения IOT Vega Admin Tool

Перед запуском кода для взаимодействия с сервером при помощи API необходимо изменить в коде строки, которые выделены на рисунке 57.

```

1  from websocket import create_connection
2  import json
3  import time
4  from datetime import datetime
5  import tzlocal
6
7  # Data section
8
9  # Autorization on VEGA server
10 autreq = {"cmd": "auth_req", # Don't change!
11          "login": "root", # Login name
12          "password": "123456" # password
13          }
14 # Get information from connected server
15 srvinfo = {"cmd": "server_info_req"} # Don't change!
16
17 # Get device list w/attributes
18 devalist = {"cmd": "get_device_appdata_req"} # Don't change!
19
20 # Get data from devices
21 datadev = {"cmd": "get_data_req",
22          "devEui": "AC1F09FFFE015304"
23          } # Don't change!
24
25 # Get reg users
26 reguser = {"cmd": "get_users_req"} # Don't change!
27
28 # =====
29 # Code section
30
31 # Connection to VEGA server
32 # Start connection. IP Address:Port VEGA server
33 ws = create_connection("ws://192.168.17.106:8002/")
34 ws.send(json.dumps(autreq)) # Get Autorization command
35 autresp = ws.recv() # Status (responce) of execute command

```

Рисунок 57 – Строки подлежащие редактированию

В строке с password необходимо ввести вместо “123456” - тот пароль, который был введен вами во время настройки файла конфигурации. То же самое необходимо сделать с IP-адресом.

## 5.2. Работа с устройством по сбору параметров

Первым делом необходимо собрать в одну цепь все модули устройства. Сделать это необходимо согласно схеме, показанной на рисунке \*\*. Все устройства питаются от микроконтроллера, который сам подключен к источнику (через USB к компьютеру, от которого одновременно загружается необходимая прошивка или от аккумулятора). Питание осуществляется выводом на 3.3 В. Заземление – GND. Выходы сенсора «питание» и «заземление» – к выводам питания 3V3 и GND на микроконтроллер, выводы для снятия показаний с сенсоров: DIO29 - к D15 на плате микроконтроллера (SCL), DIO30 – к D14 (SDA). Конечное устройство RAK 811 LoRa подключено при помощи XBee USB Adapter к D2 и D8 (RX и TX).

После этого на микроконтроллер необходимо загрузить прошивку. Если использовать Mbed OS, то сделать нужно следующее: необходимо из репозитория по ссылке [https://github.com/SkripaHella/IoT\\_Project](https://github.com/SkripaHella/IoT_Project) (или отсканировать QR-код, показанный на рисунке 58) скомпилировать и собрать проект (цифра 1), после чего нажать на кнопку (цифра 2, рисунок 59). [19]



Рисунок 58 – QR-код на репозиторий с проектом в GitHub

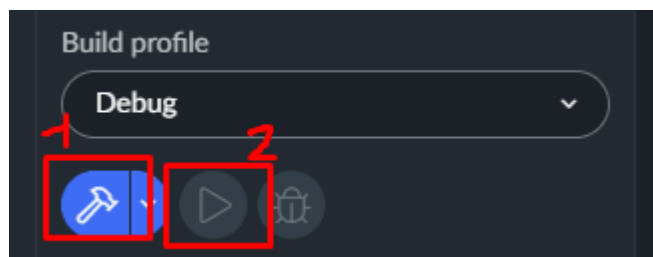


Рисунок 59 – Сборка и загрузка прошивки

После успешной загрузки начнет мигать микроконтроллер (встроенное в него системное мигание при загрузке прошивки).

При удачной загрузке устройство начнет передавать данные на сервер, на котором можно отслеживать температуру в аудиториях.

## Заключение

В ходе выполнения проекта «Сбор микроклиматических параметров в учебных аудиториях»:

- Была изучена документация API-функций сервера Вега и выбраны те функции, которые были реализованы в программе.
- Были изучены принципы работы и обеспечения безопасности протокола LoRaWAN.
- На основе изученных материалов было создано устройство по сбору микроклиматических параметров, предназначенное для работы в учебных аудиториях, в частности была написана прошивка для микроконтроллера, реализующая процесс съема, обработки и отправки на сервер данных о температуре. Также были настроены базовая станция, сервер и взаимодействие между ними и с ними описанного устройства.
- Была протестирована работа созданной системы – стабильное подключение конечного устройства к серверу, снятие показаний температуры (бывают сбои самого датчика), переподключение устройства к БС, передача данных с нескольких устройств, передача данных на большом расстоянии (с 1 на 7 этаж). Результат проверки – положительный, все перечисленные пункты представили стабильные и хорошие показатели.

Пояснительная записка была написана в соответствии с требованиями ОС ТУСУР 01-2021.



## Список использованных источников

1. Образовательный стандарт ВУЗа ОС ТУСУР 01-2021. [Электронный ресурс]. – Режим доступа: [https://storage.tusur.ru/files/40668/rules\\_tech\\_01-2021.pdf](https://storage.tusur.ru/files/40668/rules_tech_01-2021.pdf) (дата обращения 26.05.2022).
2. API. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/API> (дата обращения 31.03.2022).
3. LoRaWAN Message Types. [Электронный ресурс]. – Режим доступа: <https://www.thethingsnetwork.org/docs/lorawan/message-types> (дата обращения 28.02.2022).
4. LoRaWAN Security. [Электронный ресурс]. – Режим доступа: <https://www.thethingsnetwork.org/docs/lorawan/security> (дата обращения 28.02.2022).
5. Обеспечение безопасности в беспроводных протоколах на примере LoRaWAN [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/458394> (дата обращения 03.03.2022).
6. Сеть LoRaWAN: безопасность обеспечивается [Электронный ресурс]. – Режим доступа: <https://www.iksmedia.ru/articles/5573226-Set-LoRaWAN-bezopasnost-obespechiva.html> (дата обращения 03.03.2022).
7. Подробное описание интерфейса I2C. [Электронный ресурс]. – Режим доступа: <https://radioham.ru/i2c/> (дата обращения 25.03.2022).
8. R3000 LG, Robustel [Электронный ресурс]. – Режим доступа: <https://www.euromobile.ru/produkcija/bazovye-stancii-lorawan/r3000-lg4la/> (дата обращения 01.06.2022).
9. Вега БС-3 [Электронный ресурс]. – Режим доступа: <https://iotvega.com/product/bs03> (дата обращения 01.06.2022).
10. Базовые станции LoRaWAN [Электронный ресурс]. – Режим доступа: <https://isup.ru/articles/50/14483/> (дата обращения 01.06.2022).

11. Dragino LHT52 [Электронный ресурс]. – Режим доступа: <http://wiki.dragino.com/xwiki/bin/view/Main/User%20Manual%20for%20LoRaWAN%20End%20Nodes/LHT52%20-%20LoRaWAN%20Temperature%20%26%20Humidity%20Sensor%20User%20Manual/> (дата обращения 01.06.2022).
12. LoRa RAK 7204 [Электронный ресурс]. – Режим доступа: <https://www.reichelt.com/fr/en/environmental-sensor-lorawan-rak-7204-p271521.html> (дата обращения 01.06.2022).
13. Bera BC-2.2. [Электронный ресурс]. – Режим доступа: <https://iotvega.com/product/bs02-2> (дата обращения 07.04.2022).
14. API VEGA-lora rev21 [Электронный ресурс]. – Режим доступа: <https://iotvega.com/content/ru/soft/server/API%20VEGA-lora%20rev21.pdf> (дата обращения 07.04.2022).
15. NUCLEO-F103RB - STM32 Nucleo-64 development board with STM32F103RB MCU. [Электронный ресурс]. – Режим доступа: [https://www.st.com/content/st\\_com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-nucleo-boards/nucleo-f103rb.html](https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-nucleo-boards/nucleo-f103rb.html) (дата обращения 25.02.2022).
16. UnwiredDevices/Umdk-thp. [Электронный ресурс]. – Режим доступа: <https://github.com/unwireddevices/umdk-boards> (дата обращения 25.03.2022).
17. UMDK-THP. [Электронный ресурс]. – Режим доступа: <https://www.unwireddevices.com/docs/umdk/umdk-thp/> (дата обращения 25.03.2022).
18. Спецификация LoRaWAN. Активация оконечных устройств. [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/317218/> (дата обращения 08.05.2022).

## Приложение А

### (Обязательное)

Код для взаимодействия с сервером при помощи API

```
from websocket import create_connection
import json
import time
from datetime import datetime
import tzlocal

#### Data section
## Autorization on VEGA server
autreq = {"cmd": "auth_req", # Don't change!
"login": "root", # Login name
"password": "123456" # password
}
## 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 data from devices
datadev = {"cmd": "get_data_req",
"devEui": "AC1F09FFFE015304"
} # Don't change!

##Get reg users
reguser = {"cmd": "get_users_req"} # Don't change!
```

```
#### =====
```

```
### Code section
```

```
##Connection to VEGA server
```

```
ws = create_connection("ws://192.168.31.129:8002/") # Start connection. IP  
Address:Port VEGA server
```

```
ws.send(json.dumps(autoreq)) # Get Autorization command
```

```
autresp = ws.recv() #Status (responce) of execute command
```

```
print(autresp)
```

```
print("=====")
```

```
## Get server info
```

```
ws.send(json.dumps(srvinfo)) # Get command
```

```
srvinfresp = ws.recv()
```

```
infresp_dict = json.loads(srvinfresp)
```

```
time_serv_now = infresp_dict["time"]["utc"]/1000
```

```
local_timezone = tzlocal.get_localzone()
```

```
serv_time = datetime.fromtimestamp(time_serv_now, local_timezone)
```

```
print(srvinfresp)
```

```
print(serv_time.strftime("%Y-%m-%d %H:%M:%S"))
```

```
print("=====")
```

```
## Get dev list w/attributes
```

```
ws.send(json.dumps(devalist)) # Get command
```

```
devalistresp = ws.recv()
```

```
print(devalistresp)
```

```
print("=====")
```

```

## Get registered users
ws.send(json.dumps(reguser))          # Get command
reguserresponse = ws.recv()

print(reguserresponse)
print("=====")

while(True):
    ## Get save data from devices
    ws.send(json.dumps(datadev))      # Get command
    datadevresp = ws.recv()
    data_dict = json.loads(datadevresp)
    keys = data_dict.keys()
    keys1 = str(keys)
    if keys1 == "dict_keys(['appEui', 'cmd', 'data_list', 'devEui', 'direction', 'status',
'totalNum'])":
        data = [{"devEui": data_dict["devEui"], "data":
data_dict["data_list"][0]["data"]}, {"type": data_dict["data_list"][0]["type"]}
        print(data)
        realtime = data_dict["data_list"][0]["ts"]/1000
        packet_time = datetime.fromtimestamp(realtime, local_timezone)
        print(packet_time.strftime("%Y-%m-%d %H:%M:%S"))
    else:
        print(datadevresp)
    print("=====")
    time.sleep(5)

```

Приложение Б  
(Обязательное)

Код прошивки микроконтроллера

```
#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 gotoSleep(void)
```

```

//{
// //включение часов управления PWR
// RCC->APB1ENR |= (RCC_APB1ENR_PWREN);

// //установка бита SLEEPDEEP в регистре управления системой Cortex
// SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk; // регистр управления системой
SCB

// выбирается обычный сон или глубокий сон (как выбрано в данном случае)
// //выбор режима ожидания
// PWR->CR |= PWR_CR_PDDS; // переход к регистру питания
периферийного устройства или переход к регистру управления и установка 1
в бит PWR_CR_PDDS

// //снятие флага пробуждения, очищение
// PWR->CR |= PWR_CR_CWUF; // если флаг WUF будет иметь значение 1,
то в режим сна устройство не перейдет. Сам по себе WUF в режиме чтения.
Поэтому используем CWUF и через него меняем

// //включение запроса пина для пробуждения Ожидание Прерывания
// PWR->CSR |= (PWR_CSR_EWUP); // устанавливаем в 1 бит сброса флага
пробуждения.

// в даташите нет этого этапа, но он отчасти нужен, так как используем
пробуждение - переходим в регистр состояния управления питанием и
включаем пробуждение
// //ожидание Прерывания
// __wfi();
//}

```

```

void print_f(char temperature[], char pressure[], char humidity[])
{

```

```

pc.printf("%i\n", sensor.getTemperature());
pc.printf("-----\n");
for (int i = 0; i < 2; i++)
{
pc.printf("%c", temperature[i]);
}
pc.printf("\n");
pc.printf("-----\n");
pc.printf("-----\n");

```

```

pc.printf("%i\n", sensor.getPressure());
pc.printf("-----\n");
for (int i = 0; i < 4; i++)
{
pc.printf("%c", pressure[i]);
}
pc.printf("\n");
pc.printf("-----\n");
pc.printf("-----\n");

```

```

pc.printf("%i\n", sensor.getHumidity());
pc.printf("-----\n");
for (int i = 0; i < 2; i++)
{
pc.printf("%c", humidity[i]);
}

```

```

pc.printf("\n");
pc.printf("-----\n");
pc.printf("-----\n");

```



```

}

//char inttochar(int param)
//{
//}

int main()
{
int q=0;
while(1) {
char command_WAKE_UP[2] = {'a', 't'};
for(int i = 0; i < strlen(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) // 10
{
for(int i = 0; i < 9; i++)
{
dev.putc(command_JOIN[i]);
pc.printf("%c", command_JOIN[i]);
}
q=0;
}

```

```

wait(10);
}
q++;
//printf(data.Temperature, data.Pressure, data.Humidity);
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;
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';
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];
}
free(temp);

// типа метод по преобразованию инта давления в символы
//char* pres;
//int k2 = sensor.getPressure();
//pres = (char *)malloc(10 * sizeof(char));
//int v2 = 0; //количество цифр в числе n
//разбиваем на отдельные символы число n
//while (k2 > 9)
//{
// pres[v2++] = (k2 % 10) + '0';
// k2 = k2 / 10;
//}
//pres[v2++] = k2 + '0';
//pres[v2] = '\0';
//char t2;
//инвертируем массив символов
//for (int i = 0; i < v2 / 2; i++)
//{
// t2 = pres[i];
// pres[i] = pres[v2 - 1 - i];
// pres[v2 - 1 - i] = t2;
//}
//v2 = 0;
//for(int i = 0; i < strlen(pres);i++)
//{
// command_SEND[i+15+strlen(temp)] += pres[i];
//}

```

```

//free(pres);

// типа метод по преобразованию инта влажности в символы
//char* hum;
//int k3 = sensor.getHumidity();
//hum = (char *)malloc(10 * sizeof(char));
//int v3 = 0; //количество цифр в числе n
//разбиваем на отдельные символы число n
//while (k3 > 9)
//{
// hum[v3++] = (k3 % 10) + '0';
// k3 = k3 / 10;
//}
//hum[v3++] = k3 + '0';
//hum[v3] = '\0';
//char t3;
//инвертируем массив символов
//for (int i = 0; i < v3 / 2; i++)
//{
// t2 = hum[i];
// hum[i] = hum[v3 - 1 - i];
// hum[v3 - 1 - i] = t3;
//}
//v3 = 0;
//for(int i = 0; i < strlen(hum);i++)
//{
// command_SEND[i+15+strlen(temp)+strlen(pres)] += hum[i];
//}
//free(hum);

```

```

pc.baud(115200);
dev.baud(115200);
for(int i = 0; i < strlen(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', ':', 'l'};
for(int i = 0; i < strlen(command_SLEEP); i++)
{
dev.putc(command_SLEEP[i]);
pc.printf("%c", command_SLEEP[i]);
}

dev.putc('\n');
dev.putc('\r');
//data.Warning = '1';
ThisThread::sleep_for(5000); // или 10000 для демонстрации (а если для
реальной работы, то 2700000, то есть сон 45 минут)
}
}

```