TARTU UNIVERSITY

FACULTY OF SOCIAL SCIENCES

NARVA COLLEGE

INFORMATION TECHNOLOGY SYSTEMS DEVELOPMENT

Aleksei Kromski

# AN EMPIRICAL COMPARISON AND CONSTRUCTION OF IOT APPLIANCES FOR SUPPLY CHAIN VISIBILITY IN LOGISTICS

Bachelor's thesis

Supervisor:

*Lect.* Chen-Wan Lin, PhD

NARVA 2024

# TABLE OF CONTENTS

# INTRODUCTION

The logistics market is very much associated with the creation, distribution and support of IoT devices. IoT devices are tightly integrated into internal and external systems called the supply chain visibility (SCV), which provide the opportunity to have direct access to the flow of logistics supplies of goods, which can provide a various range of information, for example: current status of shipping, position, environment parameters and etc. This market has a very large number of solutions of different profiles, which have a major problem associated with the limitations and closedness of the system, which leads to difficulty in scaling and supporting both software and hardware. This problem is acute for both small and large businesses.

Small businesses that want to integrate any system to meet their logistics needs face several types of problems: financial constraints and difficulty in implementation. At its core, a small business is always forced to conserve the resources it has, so integrating a fairly large and extensive system can lead to large costs for specialists who will support and expand the solution. On the other hand, we can look at large enterprises that face a different kind of problem. For large enterprises that work in the field of logistics or use the logistics integration of another company in their product, they have the problem of difficult integration of new solutions. Large companies tend to leave outdated technologies and maintain them, while they could start to use more standard tool for more flexible integration to new businesses. A system that is too large and complex should be replaced with standards that would help a large company change its direction in a relatively short time, which also entails fewer risks for business, since if many companies use this standard, then it can be argued that the system will work without any problems.

It can be said with great accuracy that the transition to a certain standard will bridge the gap between small and large businesses, which will give them the opportunity to cooperate with each other without any problems, which leads to increased productivity and optimization of resources

Thus, the integration of a common software standard and the creation of IoT devices into the logistics industry requires a detailed understanding of the unique challenges faced by enterprises of different sizes. While small businesses may prioritize cost-effectiveness and ease of implementation of the Internet of Things, large enterprises need to overcome the

complexities of system integration and standardization, which will bring greater productivity in the future.

The focus of this thesis is the development of an affordable IoT device designed to collect

- environmental parameters: humidity, pressure, temperature
- position parameters: altitude, coordinates

Since very often, when ensuring high-quality delivery of goods, a control system is required for various parameters listed above. For example, to ensure timely delivery of food products, you must always take into account the temperature inside the truck. By providing the driver with this information using this device, spoiled product deliveries can be prevented. Using readily available and cost-effective sensors from popular markets, the device being developed aims to democratize access to advanced SCV (Supply Chain Visibility) technology and more.

An IoT device for data collection will be based on the advantages and versatility of the Arduino platform, namely scalability, price, performance. These points are indispensable for a company that decides to create a device on its own. The practical part also includes building a server application to ensure uninterrupted collection, processing, storage, retrieval and monitoring of data in real time. The software must be adapted to the needs of logistics companies and more. The application should make it easy to integrate new features, thereby increasing the quality and value of the product. One of the fundamental things in this project is that all software and materials for creating the device, as well as extensions of the server application, will be free. This will allow many organizations to integrate or implement this standard into their business or SCV (Supply Chain Visibility) infrastructure. By providing this application for free, small companies can consider integrating this idea into their business and be competitive in the market, as well as have easier cooperation with larger companies. On the part of large companies, there will be an opportunity to redesign the concept of their software and hardware environment for a greater number of integrations.

This project is aimed at making a major contribution to the field of logistics and looking at old things in a new way and giving more opportunities to small businesses, as well as improving the system of large companies by simplifying the integration of an IoT device and a special and simple application.

IoT firmware will be created to perform the following functions:

- Collect sensor data
- Create string with data in special format
- Send data to server receiver

Server software will be created to perform the following functions:

- Receive data from the IoT appliance.
- Analyse and process the collected data.
- Store the data in a database for later access.
- Provide sorted data upon user request.
- Broadcast data in real time for monitoring.

Web interface will be created to perform the following functions:

- Provide ability to search telemetry
- Preview telemetry in live-time mode
- Configure access rights to application
- Control of the devices

To get a complete picture of the work performed over the firmware of the device, server software and web application, in Appendix 1 you can find a table that describes all the tasks completed in more detail

# LIST OF TERMS AND DEFINITIONS

IoT – *"the ability to make everything around us starting from (i.e. Machine, Devices, Mobile phone and Cars) even (Cities and Roads) are expected to be connected to the Internet [17]"*

SCV – *"is defined as the stakeholders' capability to have access to accurate and timely information about the flow of goods [18]"*

# MAIN CONTENT

## RESEARCH BACKGROUND

In order to identify all the positive and negative aspects of my IoT device and software application, it is worth using comparative analysis. Comparative analysis will help to obtain a number of key aspects, which depend on the focus of the analysis, to obtain a more correct and clear picture of the whole. The first analysis is aimed at identifying the difference in the functionality of an IoT device offered by a third-party company. The second analysis is aimed at making a comparative analysis of the proposed service software. The parameters selected include the ability to view information in real time, perform a query to obtain data for a certain period of time, a user management system, and a device management system

### COMPARATIVE ANALYSIS OF DEVICE FUNCTIONALITY

Comparative analysis is carried out on the basis of: product functionality and pricing. A special point is that a comparative analysis based solely on price will not be complete. This is due to the extensive and widespread practice of large firms to refrain from disclosing price details publicly. Features of this reluctance relate to the adoption of a customer-centric approach, the inherent complexities associated with pricing direction, the competitive environment in the industry, considerations of product exclusivity and the adoption of business strategy in the formation of dynamic pricing.

## COMPARATIVE ANALYSIS OF SERVICE FUNCTIONALITY

When searching for devices with similar parameters, I found a fairly large number of devices that cover all the parameters listed in the table

| Solution name | Company name | Support 4G LTE FDD, 3G | GPS | Temperature | Altitude | Pressure | Humidity | Additional functionality | Price | Link |
|---|---|---|---|---|---|---|---|---|---|---|
| **TK419** | Eelink | +, Full | + | - | - | - | - | IButton authorization | 43 USD / 40.39 EUR (18.04.2024) | https://www.eelinktracker.com/4G-car-GPS-tracker-TK419-with-iButton-for-driver-ID-identify-tracking.html |
| **GPT49-Eelink** | Eelink | +, Full | + | - | - | - | - | Long-life battery | 44 USD / 41.33 EUR (18.04.2024) | https://www.eelinktracker.com/4g-lte-wcdma-gps-tracker-long-battery-life.html |
| **Saga and Saga P** | Controlant | +, Full | + | + | - | - | - | Long-life battery | - | https://www.controlant.com/devices |
| **TempTale GEO X** | Sensitech | +, Full | + | + | - | - | + | Long-life battery, light | - | https://www.shareddocs.com/hvac/docs/2000/Public/0A/temptale-geo-x-ls.pdf |
| **TempTale GEO X** | Sensitech | +, Full | + | + | - | - | + | Long-life battery, light | - | https://www.shareddocs.com/hvac/docs/2000/Public/09/temptale-geo-xp-ls.pdf |
| **TempTale GE** | Sensitech | +, Full | + | + | - | - | + | Long-life battery, light | - | https://www.shareddocs.com/hvac/docs/2000/Public |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **O Ultra** | | | | | | | | | | /0F/temptale-geo-ultra-food.pdf |
| **Atlanta (this thesis project)** | Aleksei Kromski | +, 2G - WIFI | + | + | + | + | + | Long-life battery | Free | |

*Figure 1 Table of different devices and their functions*

Devices from the following companies were selected: Eelink, Controlant and Sensitech. Comparing the functional part of these devices, we can draw the following conclusions. All devices support cellular communications as a data transfer method. Devices from Sensitech and Controlant have a mixed type of operation, namely, they use different types of cellular networks, from 4G to 2G, which makes them more competitive with my device, since my device only supports a 2G network, but at the same time has the ability to work on WIFI, which can show more stable results, so this functionality of my device covers the disadvantage of lack of connection to 3G+ networks. The reason why this module, which supports only 2G networks, was chosen is the price and its capabilities. But this module can be easily replaced with a newer one that is supported by the 3G+ network. The next parameter is to get the geoposition. All devices in the table have this capability, and devices from Sensitech can also determine positions using cellular communications by using triangulation calculation systems and location over WiFi. Making a comparison, I can highlight that my component of my device is slightly inferior to the capabilities of the device from Sensitech, but at the same time it covers the same needs. But as described above, network module can be improved by purchasing a more advanced network module. The following is a comparison of the temperature tracking capability parameter. Devices from Sensitech and Controlant have this functionality, like my device, while devices from Eelink do not provide this feature. The next parameter is altitude and pressure, none of the selected devices provide these capabilities, while my device can track 2 types of altitude based on data from GPS and data from the BMP180 sensor. These parameters are very important when transporting medicines or food products on any type of transport, since failure to comply with these parameters can lead to product damage. Sensitech devices have the ability to measure humidity, unlike other companies in the table. My device also provides this

opportunity. Next comes a parameter that is an additional functionality and does not particularly affect the subsequent assessment, since all devices have batteries with a long life expectancy, and mine is not exclusive. The main problem is to correctly estimate the operating time and this parameter, unfortunately, is not publicly available. Also, devices from Sensitech have an additional light parameter, which provides a way to understand what level of light is in the container, which is a definite advantage, since the device of this project does not have such an option.

As described above, not all devices have specific information published in the public domain, since companies are targeting customers and a more flexible pricing system, which makes their business more competitive. But if we make a comparison with the Eelink company, then my device comes out at the same price, but has more extensive functionality of use, as well as the ability to update components, since the device can be used without any permission and will not have legal problems

Summing up the results of this comparative analysis, I can say that my device is superior to the devices of Eelink and Controlant, since it has more possibilities for getting data and breadth of application; we should also not forget that my device is very easy to update with additional components, with suitable qualifications of the developer. Among the minuses, it can be noted that my device works using an outdated 2G connection network, since all the devices on the list support 3G+ networks, but as mentioned above, my device covers this minus by using a WiFi module if necessary.

As a second comparative analysis, based on the table below

| **Company** | **Live tracking** | **Search option (with dates)** | **Device management system** | **User management system** | **Independent data types system** | **Link** |
|---|---|---|---|---|---|---|
| **Roambee** | + | + | + | + | + | https://www.roambee.com/technology-sensors/ |
| **Sensitech** | + | + | + | + | + | https://www.sensitech.com/en/products/sensiwatch-platform/ |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Project44** | + | + | + | + | + | https://www.project44.com/visibility |
| **Atlanta (this thesis project)** | + | + | + | + | + | https://atlanta.alekseikromski.com |

*Figure 2 Table of devices on a market with supported functionality*

the following parameters were selected: viewing data in real time, the ability to search by specific dates, device management system, user management system.

Based on the results of all the above parameters, we can notice that all the listed criteria are present in the service software of different companies. My project also contains all these options and is in no way inferior to ready-made solutions from other companies. The main difference is that all versions of service software from companies have some kind of price, while my project is freely available and has no restrictions on use by third parties. Thus, my solution will require much less financial costs, namely only for maintaining the system and its placement on a third-party or our own server

## COMPONENTS OF DEVICE

The practical part starts with selecting the right components, which is critical to ensure the functionality, reliability and cost-effectiveness of the device. The component selection process took into account factors such as market popularity, price, quality and quantity of documentation, and community support.

**NEO-6M GPS**


*Figure 3 NEO-6M GPS module [1]*

The NEO-6M GPS module was chosen as a geo-position tracking device, which has a large number of positive reviews on platforms due to its reliable operation. The price and quality of this module are one of the strongest features, which is very important to take into account in the scope of this work. The documentation for this module is very extensive and gives a clear idea of the capabilities of the module. Also, this module is very well supported by the community, the conclusion was made based on a search for available resources for using this module. During operation, no difficulties with use or unexpected failures of performance were identified.

*"The NEO-6M GPS module is appeared in the figure beneath. It accompanies an antenna in outside. This module has an outer receiving antenna and implicit EEPROM. To get crude GPS information it simply need to begin a sequential correspondence with the GPS module utilizing Software Serial. [2]"*

**GY-68 BMP180**

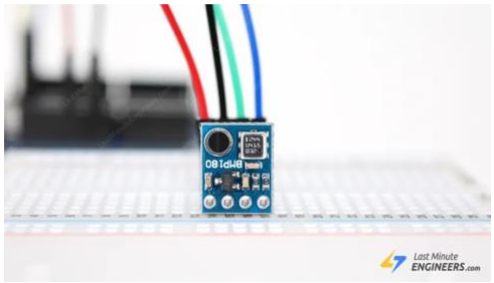



*Figure 4 GY-68 BMP180 module [3]*

The GY-68 BMP180 was chosen as a device for monitoring temperature, altitude and pressure. On the market of modules for Arduino, this module is not very popular, which does

not make it less reliable. This module does not have very well described documentation, but there are a large number of resources for integrating it into the device. The price of this module corresponds to its capabilities and quality and reliability, which means cost performance ratio is good, which makes it suitable for the design of this thesis

**SIM800L**

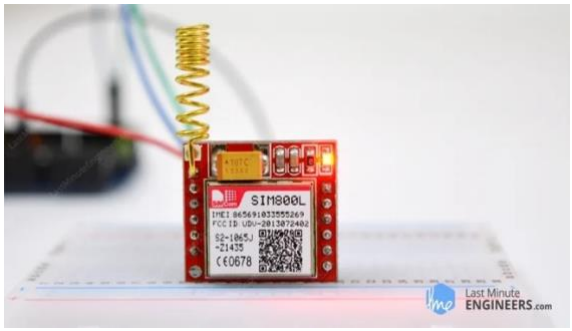



*Figure 5 SIM800L module [4]*

In order to ensure communication between the device and the server, it was decided to use a cellular network module. This module has gained great popularity among developers and enthusiasts on the Arduino platform due to its wide range of functionality and reliability. The documentation for this module is written very well and makes it possible to understand how the module works without using third-party resources.

*"SIM800L has a Quad Band 850/900/1800/1900 MHz with small dimensions, namely size 15.8 x 17.8 x 2.4 mm and weight: 1.35g. SIM800L has a low power consumption with a power supply voltage range 3.4 ~ 4.4 v [5]"*

This module also has a very extensive number of software libraries that greatly simplify working with this device. These libraries replace the process of sending AT commands to the device and reading them, since the libraries have a certain abstraction layer and the end user, namely the developer, can only work out the application logic, which is a definite advantage.

**DHT11**

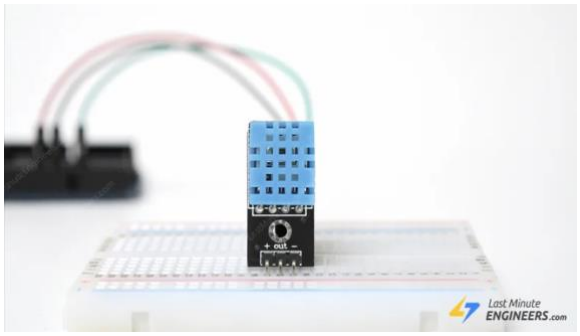



*Figure 6 DHT11 module [6]*

A DHT11 sensor was purchased to track humidity and temperature. The documentation for this module describes in great detail the principle of its operation, and this sensor has gathered a large community around it. This conclusion was made based on searching and viewing information on the Internet, which makes this sensor one of the easiest to integrate and subsequently reliably use

"*DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form [7]"*

**MT3608 DC-DC STEP UP**


*Figure 7 MT3608 DC-DC STEP UP [8]*

In order for the device to convert and stabilize the battery voltage from 3.7 to 5 volts or lower, the MT3608 converter was used. The simplicity of integration and settings makes it one of the best solutions in the market, and durability allows you to count on its performance throughout the entire service life.

**ARDUINO UNO R3 + WIFI MODULE**

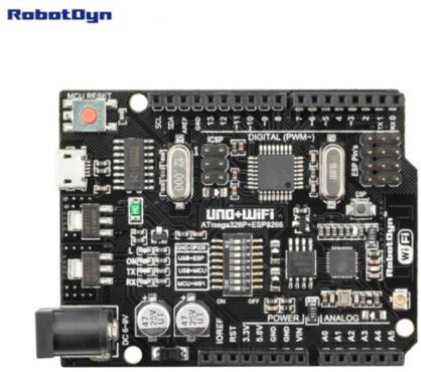



*Figure 8 ARDUINO UNO R3 + WIFI MODULE [9]*

The Arduino UNO R3 board with a Wi-Fi module was chosen as the central node that will carry out all the main work, namely, collecting and sending data. This board is a fairly popular solution on the market due to its price and large set of functions. This device contains two microcontrollers on board that can communicate with each other, which allows us to divide and delegate areas of responsibility when operating the device. A fairly extensive amount of documentation can be found on the Internet, which undoubtedly gives a better chance of solving any problem.

*"Atmega328P is fast becoming a recognised microcontroller that is at the heart of many private and industrial based research projects due to ease of use and flexible design [10]"*

*"The ESP8266 is packed with features and hence it requires minimal external circuitry and the entire solution, including the front-end module, is designed to occupy minimal PCB area [11]"*

## COPPER WIRES, RESISTOR KIT, DESOLDERING BOARD, BATTERIES WITH POWER SUPPLY MODULE

In order for the device to work, we need to buy wires to ensure the connection of modules to the board with a microcontroller, resistors for a voltage divider, breadboards for placing sensors and correct wire decoupling, batteries with a charging module, so that our device is autonomous


*Figure 9 Copper wire [12]*

*Figure 10 Resistor kit [13]*

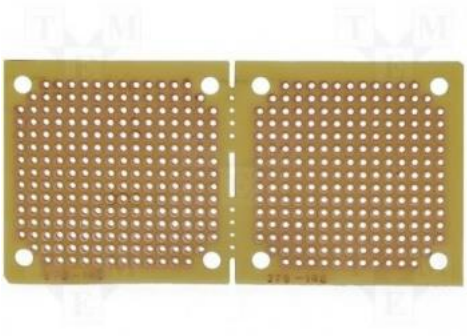


*Figure 11 desoldering  board [14]*


*Figure 12 Li-On batteries [15]*

### PRICE TABLE

You can find price table in Appendix 2. This table represents the cost of all components including tax (20%) and shipping

## DESIGN OF IOT DEVICE

### MODULES CONNECTION

The device design process is a very important stage in the development of an IOT device, which will help to proactively identify problems that may arise during the assembly process of the device. The first important point is the correct placement of step-up voltage stabilizers. Each sensor and module has its own voltage limits that must not be violated. Thus, turning to the documentation, I highlighted this table of voltages for each module

| Module name | Voltage |
|---|---|
| NEO-6M GPS | 2.7-3.7V |
| DHT11 | 3-5.5V |
| Arduino R3 + WIFI | 6-9V |
| SIM800L | 3.7-4.2V |
| GY-68 BMP180 | 3-5.5V |

*Figure 13 Modules voltage table*

Thus, based on this table, we can conclude that we will need 4 voltage converters. One for DHT11 and GY-68 BMP180, second for NEO-6M GPS, third for SIM800L and fourth for Arduino R3 + WIFI.

18650 3.7V standard batteries with a discharge module will be used as the main autonomous power supply. This option is ideal for an IOT device, as it has a large amount of electricity reserves in a small size

Next, following the table and schema, you can see how the connection of all the modules to the Arduino R3 + WIFI board was done.

| Module | Arduino R3 + WIFI pin |
|---|---|
| NEO-6M GPS | TX, RX → 3D, 4D |
| DHT11 | DAT → 2D |
| SIM800L | TX, RX → 5D, 6D |

| | |
|---|---|
| **GY-68 BMP180** | SDA, SCL → A4, A5 |

*Figure 14 Table for connecting pins to arduino board*

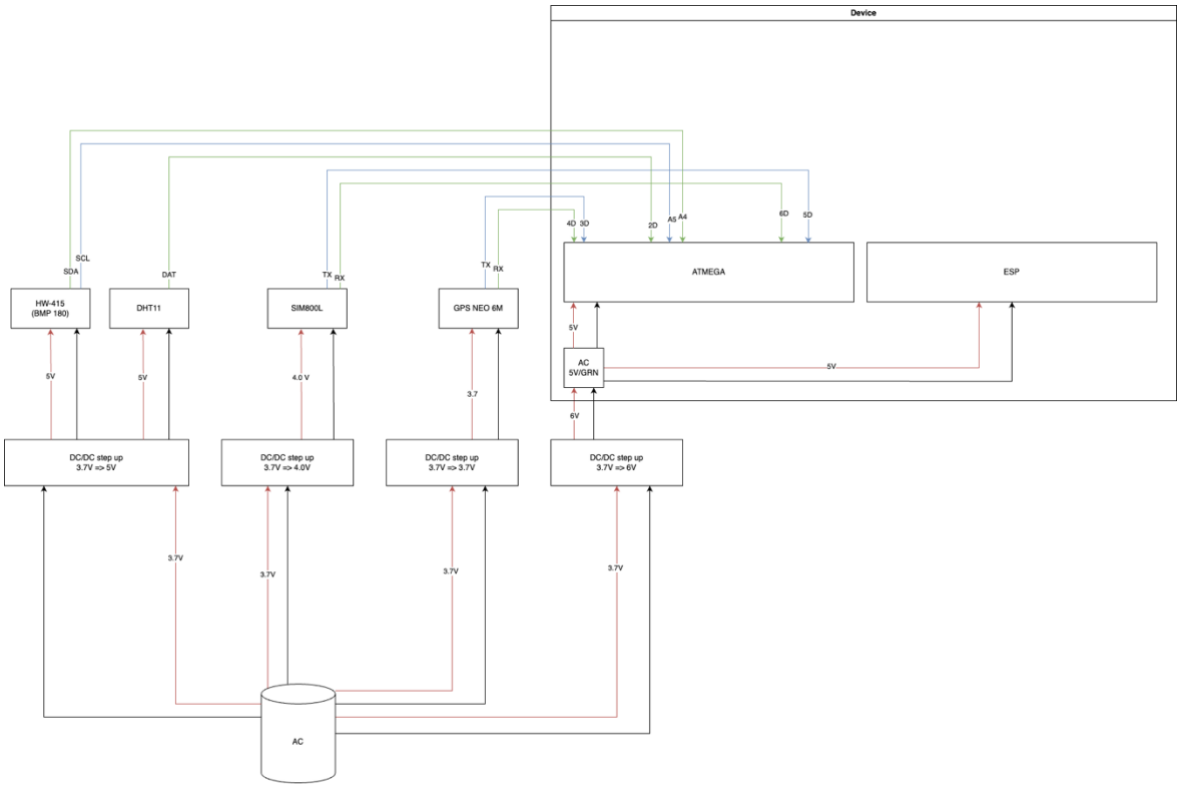



*Figure 15 Device schema*

This diagram clearly shows how the Atmega328P microcontroller has the ability to collect data from modules, format and send it directly, through the built-in capability of the Arduino UNO R3 + WiFi board to the ESP8266 microcontroller, which in turn will provide the logic for sending data to the server

An important point is the design of connecting the SIM800L module, because the logic one level on the SIM800L module is 3.3 V, so to connect the RXD pin to the Arduino, you must use a voltage divider. To do this, we will need 10 kOhm and 20 kOhm resistors and solder everything on the breadboard according to the diagram below, taking into account the correct connection to the board

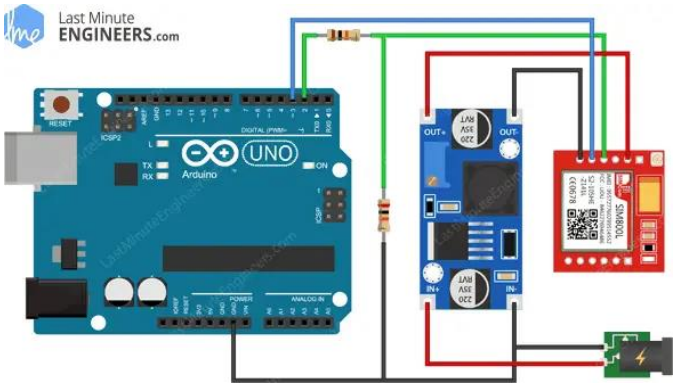



*Figure 16 SIM800L connection [16]*

Thus, we can conclude that this connection method is optimal for all modules and complies with all the listed voltage requirements, and also has a clear use of the connection interface to the Atmega328P microcontroller

**TIME PROBLEM**

During the development of the architecture of this device, I encountered the problem of accurate timestamping for datapoints. In solving this problem, the use of the **NEO-6M GPS** module resulted in the simplest and most versatile solution. Accordingly, this approach entailed the use of geopositioning technology to obtain accurate time information. Accordingly, a single timestamp was set for the transmitted data packet string format, indicating the moment of receiving data from the NEO-6M GPS sensor.

Thus, when receiving information on the server side, data points are accompanied by timestamps, which may not accurately indicate the temporal sequence of events, but timestamps can never be conveyed in the past tense. However, this discrepancy is not a significant obstacle since the data is transmitted and processed as a single package.

## FIRMWARE CODE

C++ was chosen as the programming language; this language has a large number of advantages such as speed of code execution, simple syntax, strict typing and simple integration with the Arduino platform. In order to ensure modularity of the firmware, two firmware for Atmega328P and ESP8266 will be developed separately, they are located in

separate packages in the repository from where a person can take the code and integrate it into their device.

**ATMEGA328P**

The code for Atmega328P is located in the package from atmega in the main root of the repository. After opening the atmega.ino file, you can see that at the beginning all the pin numbers that are connected to the GPIO interface of the board are determined, as well as the initialization of all the necessary libraries. After which there are global settings where we specify the scanning timeout at 4 seconds, specify the unique device ID in the uuidv4 standard and the GPS operation bound rate at 9600 according to the documentation. In the main setup function we have a definition of the main bound to which data will later be sent to the ESP8266. Here the construction of the header begins where we install the device in a special format. The GY-68 BMP180 connection is also being checked according to the documentation. In the main loop we have four lines, 3 of which are responsible for calling scanning functions, and the last one for printing to a specific bound for the ESP8266. Below we can observe three functions of which the first is responsible for scanning humidity and temperature from the DHT11 sensor, the second is responsible for scanning altitude, pressure and temperature from the GY-68 BMP180 sensor, and the rub scans information from the NEO-6M GPS sensor to obtain coordinates and time

All installed data in a line has its own specific format, the first few letters must contain a label that shows what this information belongs to, after which there is a separation by two colons and the information itself that was taken from the sensor. You must include a semicolon at the end to separate the information. This format has a very flexible wide range of use and sending data of almost any telemetry. Format example:

*DEVICE::e66a7ede-f475-484b-ad51-8e60adca3435;HUM::42.00;TEMP::23.00;GEO::0.000000,0.000000;TIME::2000-0-0T0:0:0Z;ALT::0.00;PRS::98799;ALT2::213.40;TEMP2::23.7*

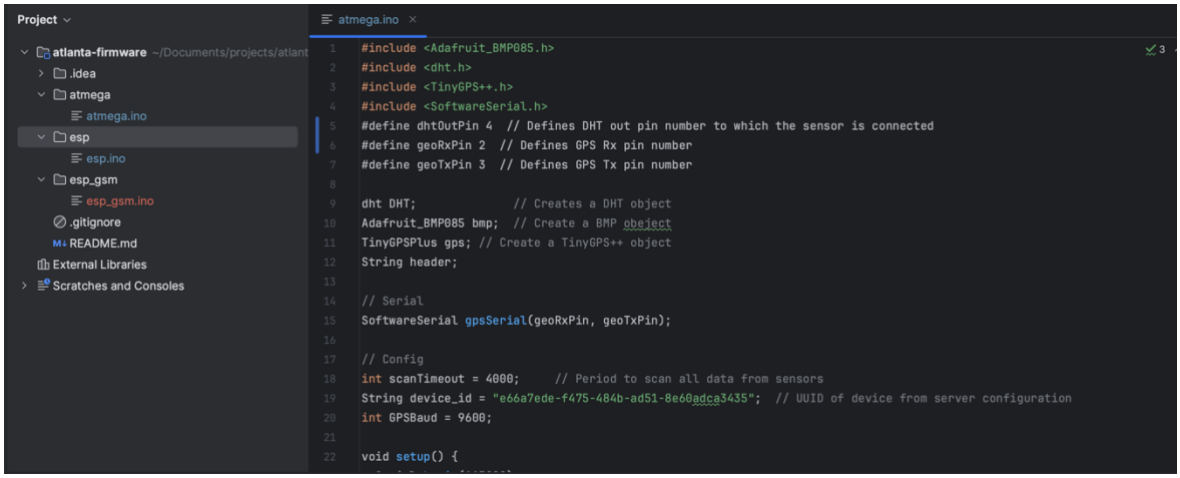



*Figure 17 Atmega firmware code picture*

## ESP8266

The code for ESP8266 wi-fi firmware is in the esp package from the root of the repository. First of all, the program installs the SSD and Wi-Fi password. The next function is a setup in which we install a specific bounty from which data will be read from Atmega328P. Next comes connecting to Wi-Fi and checking its status. In the Main loop we can see that we are checking, if we have some unread information from the Atmega328P microcontroller, then we will connect to the server using the IP address on a specific port that is specified by developer. After which the Wi-Fi client will print the line received from Atmega into the open TCP connection and close the connection, after which the server that received this line will process it


*Figure 18 ESP firmware code picture*

In the case of using SIM800L, the code will look a little different, the code will be located from the root of the repository in the esp_gsm directory, this code has an initial setting, namely setting the IP address and port where we will send data, after which the module itself is initialized and its registration is checked online. As soon as SIM800L is ready, we must check the quality of the connection to the cellular network and open a tcp connection via the AT command and send a line in a special format received from the Atmega328P microcontroller, after which the server will process the information in the same way

## SERVER SIDE APPLICATION

The server part of the application is designed to provide access to information through external interfaces, as well as the collection and processing of information through the operation of specific algorithms.

### CORE

For the seamless and correct operation of all services (modules), a core system was developed for the simultaneous launch of several modules, each of which is able to work independently, while maintaining communication with other modules. This approach has a large number of advantages, such as: modularity, scalability and independence. All modules have a common interface, which ensures smooth and high-quality integration with the application core. So that modules can communicate with each other and use each other if necessary, a dependency management system was created. A dependency management system is used to ensure that modules run correctly using a dependency tree mechanism. At startup, the kernel polls all modules, builds a dependency tree, and initiates modules in the correct sequence, facilitating communication between modules by passing the necessary dependencies. Also, to transmit data, a special structure and channel mechanism of the Golang language is used, which ensures instant delivery of information.

Below you can see a diagram of the core system

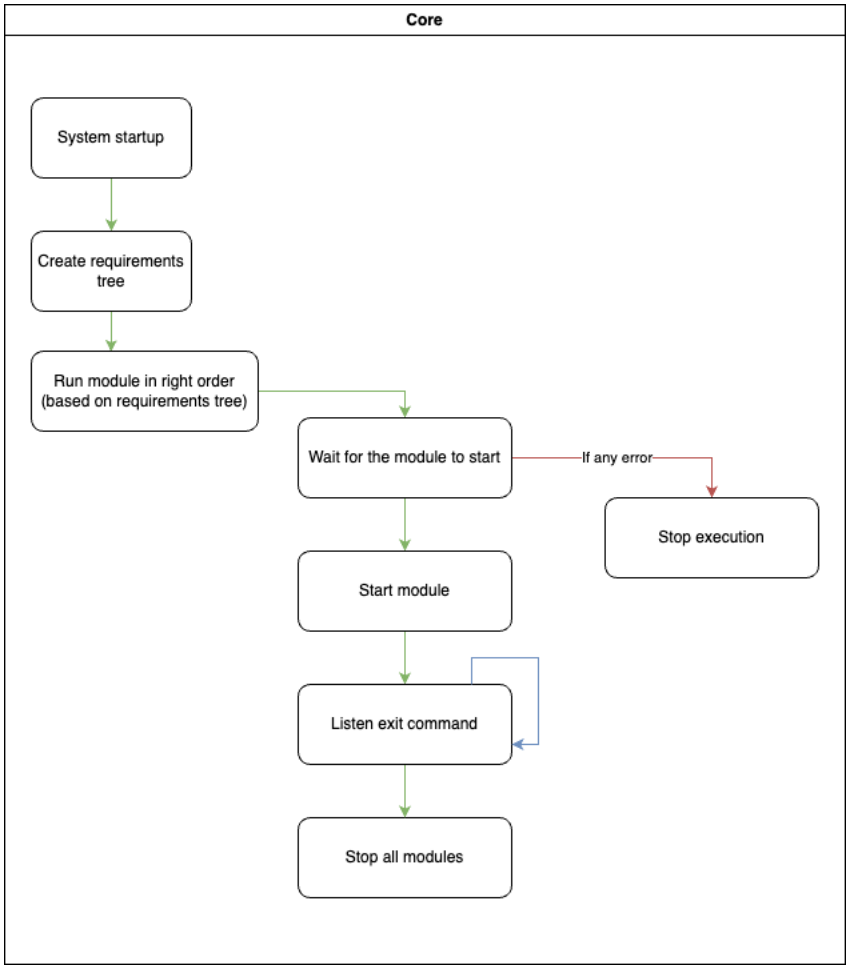



*Figure 19 Schema of core system*

## TCP_CONSUMER MODULE

The tcp consumer module includes the interface implementation provided by the core system. During initialization, this module requests a storage for storing data as a dependency, after which this module opens tcp-server and starts listening to it. As soon as the device sends data to an open TCP connection, the use of an additional parser begins. This parser can work with a string of a certain format that was listed above. The parser has a simple structure that determines the label of a datapoint, after which it determines which data model should be used and integrates the information received from the line into this model. Next, validation takes place since not all data can be successfully transmitted or transmitted in an incorrect format. This can happen if the connection between the device and the server is interrupted. After the server has successfully received and processed the data using the parser, it will save the data to the database using the storage dependency, after which it will

create and send a new event to the event channel. This information will be read by gin_server to provide live-preview functionality via the socket system

Below you can see a diagram of the tcp_consumer module

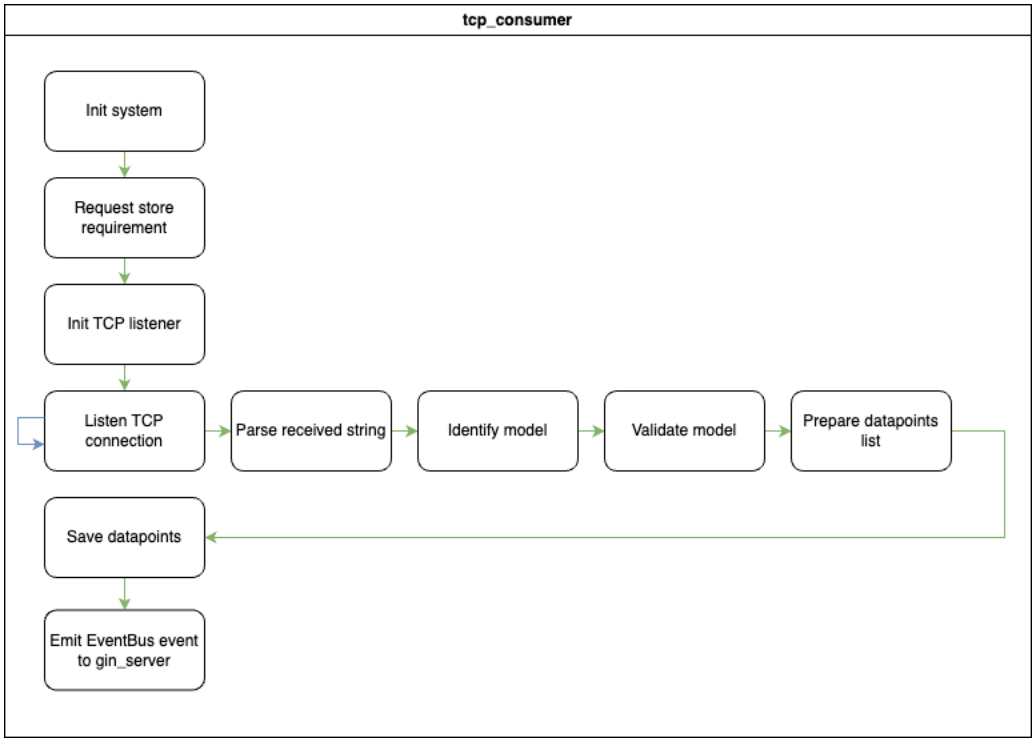
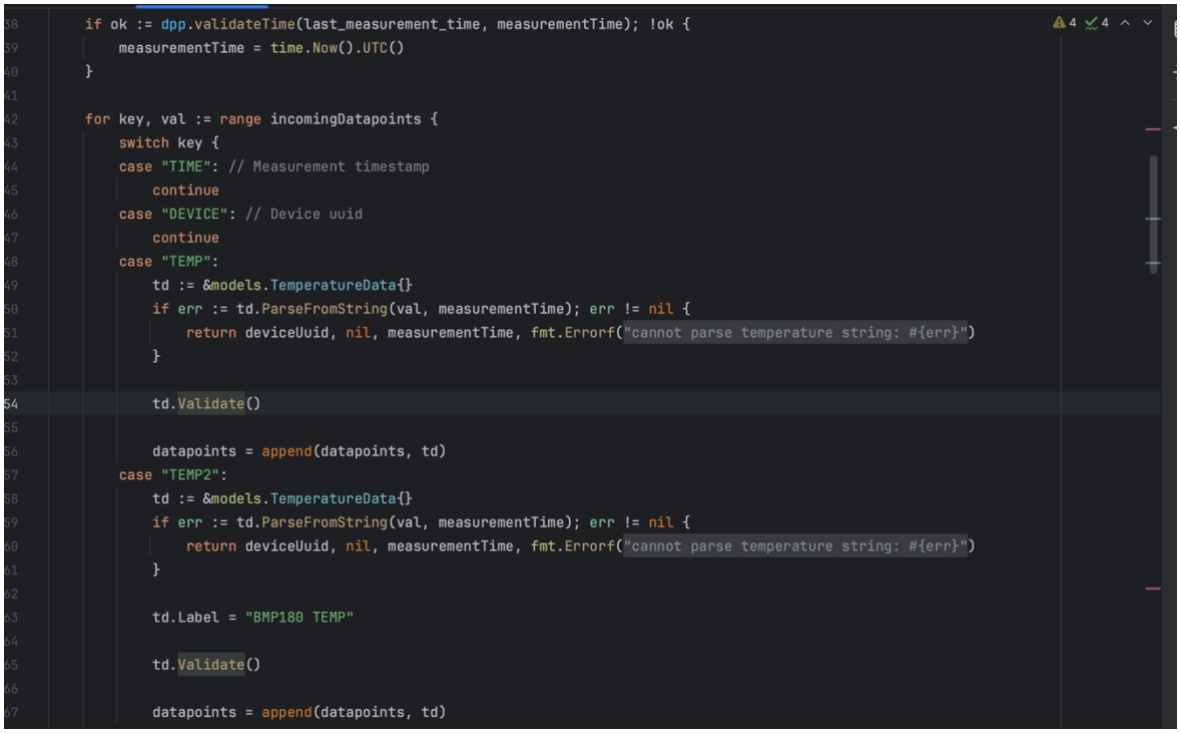



*Figure 20 Schema of tcp_consumer*

Of the screenshots below, we can observe how the datapoints processed



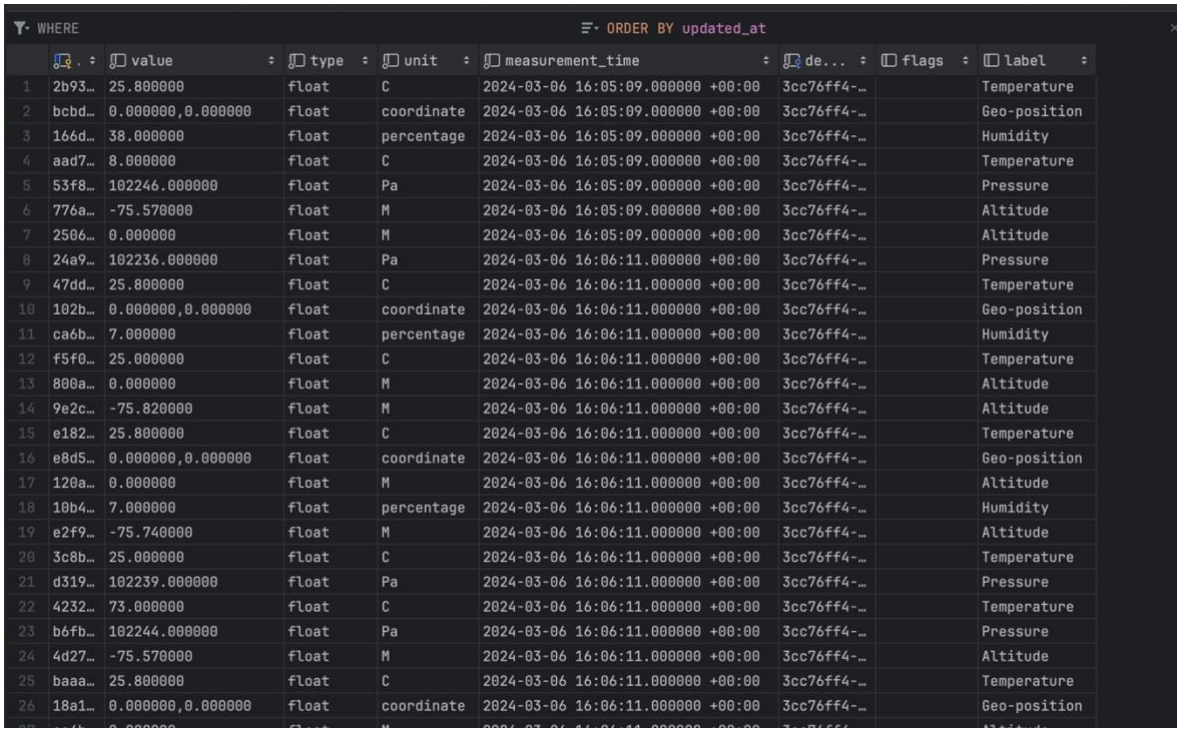
```
if ok := dpp.validateTime(last_measurement_time, measurementTime); !ok {
    measurementTime = time.Now().UTC()
}

for key, val := range incomingDatapoints {
    switch key {
    case "TIME": // Measurement timestamp
        continue
    case "DEVICE": // Device uuid
        continue
    case "TEMP":
        td := &models.TemperatureData{}
        if err := td.ParseFromString(val, measurementTime); err != nil {
            return deviceUuid, nil, measurementTime, fmt.Errorf("cannot parse temperature string: #{err}")
        }

        td.Validate()

        datapoints = append(datapoints, td)
    case "TEMP2":
        td := &models.TemperatureData{}
        if err := td.ParseFromString(val, measurementTime); err != nil {
            return deviceUuid, nil, measurementTime, fmt.Errorf("cannot parse temperature string: #{err}")
        }

        td.Label = "BMP180 TEMP"

        td.Validate()

        datapoints = append(datapoints, td)
```

*Figure 21 Code example of datapoints processing*

Of the screenshots below, we can observe how the date is from the format

*DEVICE::e66a7ede-f475-484b-ad51-8e60adca3435;HUM::42.00;TEMP::23.00;GEO::0.000000,0.000000;TIME::2000-0-0T0:0:0Z;ALT::0.00;PRS::98799;ALT2::213.40;TEMP2::23.7*

Added to the database as the datapoint



WHERE ORDER BY updated_at

| | . | value | type | unit | measurement_time | de... | flags | label |
|---|---|---|---|---|---|---|---|---|
| 1 | 2b93… | 25.800000 | float | C | 2024-03-06 16:05:09.000000 +00:00 | 3cc76ff4-… | | Temperature |
| 2 | bcbd… | 0.000000,0.000000 | float | coordinate | 2024-03-06 16:05:09.000000 +00:00 | 3cc76ff4-… | | Geo-position |
| 3 | 166d… | 38.000000 | float | percentage | 2024-03-06 16:05:09.000000 +00:00 | 3cc76ff4-… | | Humidity |
| 4 | aad7… | 8.000000 | float | C | 2024-03-06 16:05:09.000000 +00:00 | 3cc76ff4-… | | Temperature |
| 5 | 53f8… | 102246.000000 | float | Pa | 2024-03-06 16:05:09.000000 +00:00 | 3cc76ff4-… | | Pressure |
| 6 | 776a… | -75.570000 | float | M | 2024-03-06 16:05:09.000000 +00:00 | 3cc76ff4-… | | Altitude |
| 7 | 2506… | 0.000000 | float | M | 2024-03-06 16:05:09.000000 +00:00 | 3cc76ff4-… | | Altitude |
| 8 | 24a9… | 102236.000000 | float | Pa | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Pressure |
| 9 | 47dd… | 25.800000 | float | C | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Temperature |
| 10 | 102b… | 0.000000,0.000000 | float | coordinate | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Geo-position |
| 11 | ca6b… | 7.000000 | float | percentage | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Humidity |
| 12 | f5f0… | 25.000000 | float | C | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Temperature |
| 13 | 800a… | 0.000000 | float | M | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Altitude |
| 14 | 9e2c… | -75.820000 | float | M | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Altitude |
| 15 | e182… | 25.800000 | float | C | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Temperature |
| 16 | e8d5… | 0.000000,0.000000 | float | coordinate | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Geo-position |
| 17 | 120a… | 0.000000 | float | M | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Altitude |
| 18 | 10b4… | 7.000000 | float | percentage | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Humidity |
| 19 | e2f9… | -75.740000 | float | M | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Altitude |
| 20 | 3c8b… | 25.000000 | float | C | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Temperature |
| 21 | d319… | 102239.000000 | float | Pa | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Pressure |
| 22 | 4232… | 73.000000 | float | C | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Temperature |
| 23 | b6fb… | 102244.000000 | float | Pa | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Pressure |
| 24 | 4d27… | -75.570000 | float | M | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Altitude |
| 25 | baaa… | 25.800000 | float | C | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Temperature |
| 26 | 18a1… | 0.000000,0.000000 | float | coordinate | 2024-03-06 16:06:11.000000 +00:00 | 3cc76ff4-… | | Geo-position |

*Figure 22 Datapoints table*

## GIN_SERVER MODULE

The main purpose of the gin_server is to publish the API to the public. gen_tserver is a regular http server that uses the Gin Framework to implement all API capabilities. The structure of this package was developed in such a way that it was possible to have several versions of the system API. When initializing this module, we request a storage dependency. After that, we initialize the creation of the first api version and register the application api routes, and the websocket-hander, after the server starts

API version 1 is divided into several files that are strictly responsible for their area of responsibility and have handlers inside. The v1.go file implements a special API interface in which we configure the core settings and also register all the necessary endpoints and bind them to certain handlers.

| Endpoint urn | Comment |
|---|---|
| **/api/roles/get** | To get all roles |
| **/api/roles/upsert** | To CUD roles |
| **/api/users/get** | To get users |
| **/api/users/upsert** | To CUD users |
| **/api/endpoints/get** | To get endpoints |
| **/api/endpoints/upsert** | To CUD endpoints |
| **/api/devices/upsert** | To create or update devices |
| **/api/devices/delete/*** | To delete device by id |
| **/api/devices/get** | To get devices |
| **/api/auth/logout** | To logout |
| **/ws/connect** | To connect to websocket handler |
| **/api/datapoints/info/labels** | To get labels |
| **/api/datapoints/find** | To find datapoints |
| **/api/datapoints/info/devices** | To get devices |
| **/api/healthz** | To check server |
| **/api/users/current-user** | To get current user (after login) |
| **/api/users/current-user-upsert** | To update current user info |
| **/api/store/get/*** | To get some value from KeyValue storage |
| **/api/store/upsert** | To CUD some value to KeyValue storage |
| **/api/pushapi/get** | To CRUD push api config |
| **/api/pushapi/upsert** | To CRUD push api config |

*Figure 23 Table of endpoints*

The gen_server module also includes an authorization system that is used in the first version of the API and also when trying to connect to a websocket handler. All handlers use a storage dependency to retrieve, create, update or delete data by using the storage interface, which will be described below. In turn, the websocket handler uses a ready-made library written by the author of this thesis at-socket-server. This library allows you to establish authorized access to a websocket and transfer data in a specific format

Below you can see a diagram of the gin_server module

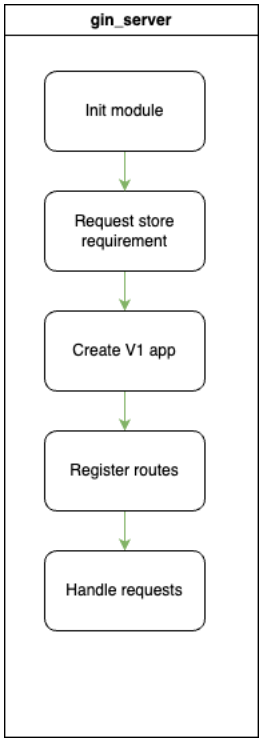



*Figure 24 Gin server schema*

Of the screenshots below, we can observe the json example of search endpoint result

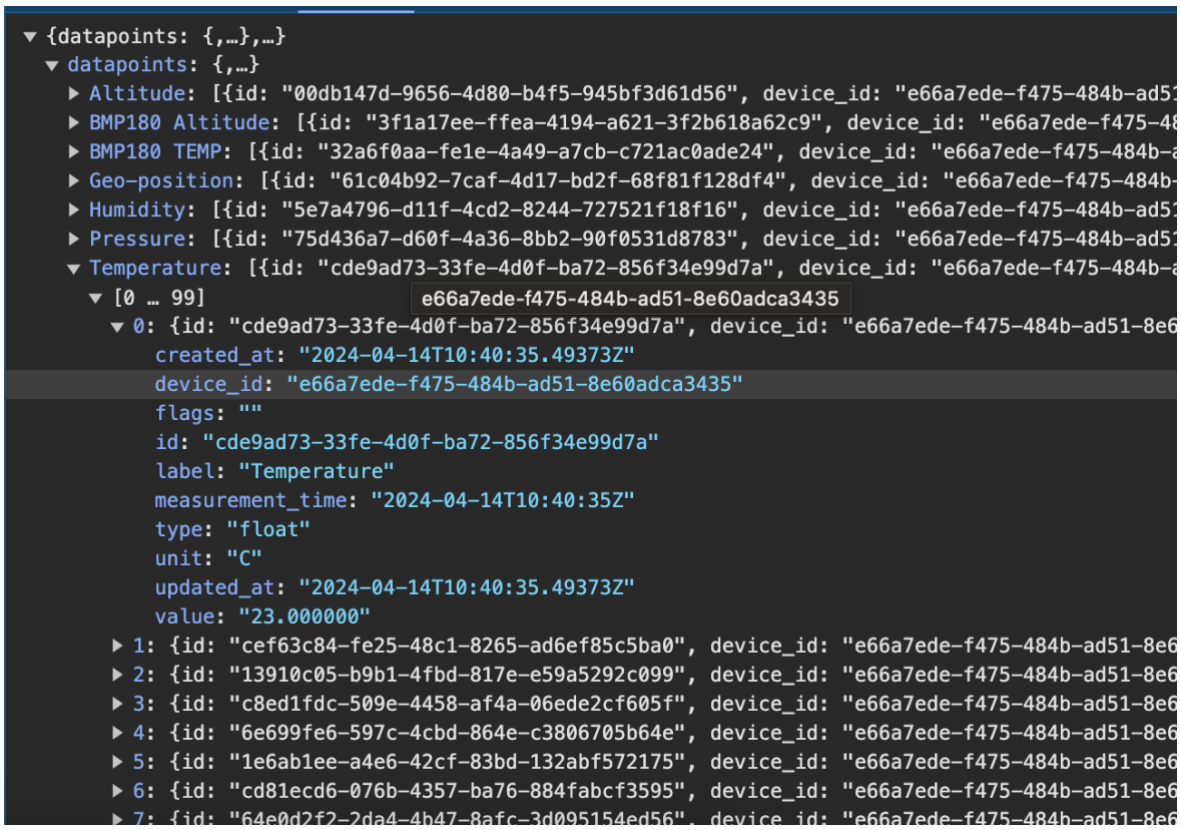



*Figure 25 Json response from server for search endpoint*

## STORAGE MODULE & POSTGRES IMPLEMENTATION

The storage module is an abstraction for implementing various other means of storing information. This module contains an interface that has all the necessary interfaces to implement the operation of this application. This approach allows you to use different data storage systems independently of each other and provide a convenient way to switch between these data storage systems

Postgres implementation of storage interfaces includes a migration system. The immigration data is needed to initialize the database in the correct format with the information already provided. This sub-module has various files that implement all the interfaces of the storage interface. When starting, this module initializes the connection to the server using the transmitted configuration, then tries to ping the server, after which it starts all migrations

and notifies the system that it is ready for work, after which the core system will implement this module into other submodules to implement the system of dependencies

Below you can see a diagram of the storage / postgres module

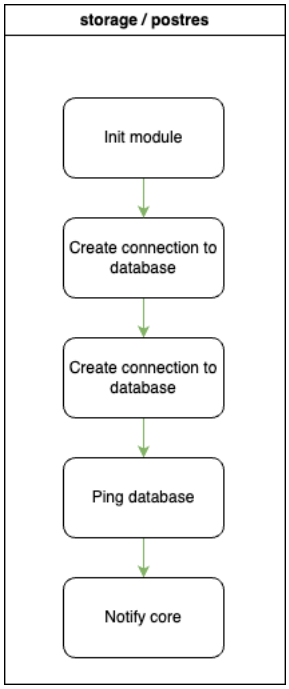



*Figure 26 Schema of storage / postgres implementation*

## POSTGRES DATABASE STRUCTURE

The Postgres system was chosen as the main database since it has extensive documentation and has also gained its importance among developers of various applications

The database for the application was developed taking into account all needs such as saving telemetry, setting access parameters for users, creating users, creating devices and storage based on the key-value principle

Below you can see a diagram of the database structure

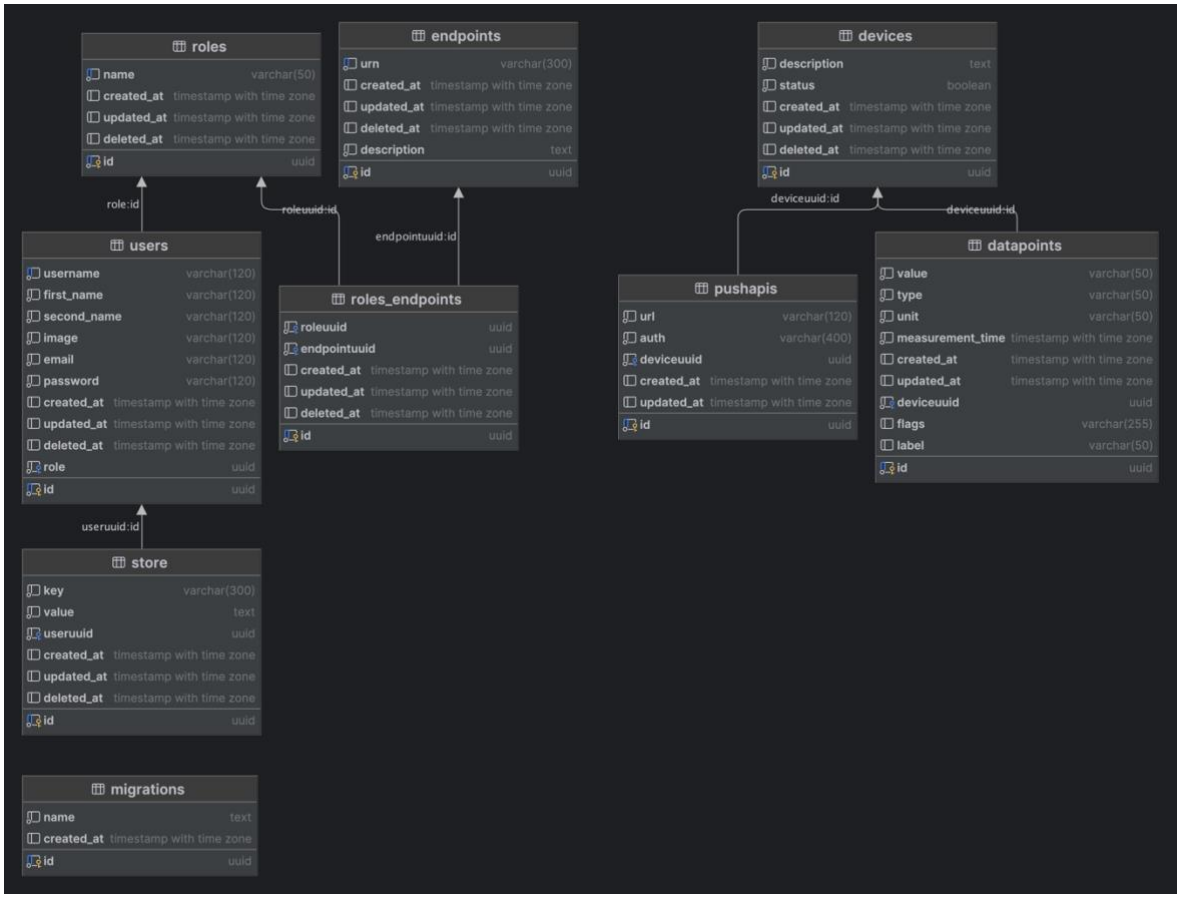



*Figure 27 Database tables schema*

## PUSH API MODULE

The push api module works on a simple principle: as soon as we received some incoming telemetry for a certain device and we were able to successfully process it, we access the storage and get a list of api endpoints that are tied to certain devices and send the incoming data, namely telemetry, to an external resource. The push api module works on a simple principle: as soon as we received some incoming telemetry for a certain device and we were able to successfully process it, we access the storage and get a list of api endpoints that are tied to certain devices and send the incoming data, namely telemetry, to an external resource

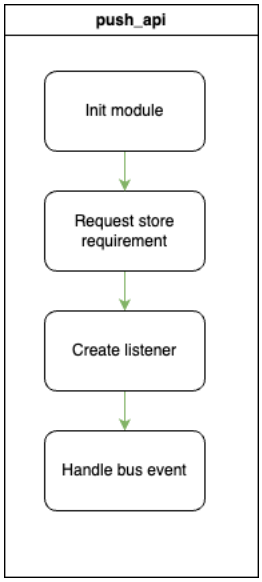



*Figure 28 Push api schema*

## CLIENT SIDE WEB APPLICATION

To create the web interface, the React framework was used - it is a flexible and good tool for creating a strong and advanced front-end experience. The NextUi library was used as the basis for the design. This library has a large number of components and good documentation, making it very easy to use. The axios library was used as a library for creating requests on the server side. The entire system of components was divided into a special structure where we have a separate folder with pages into which other components included in it are then imported. These components use the same wrapper class, which has a very convenient modal window for viewing information. When you click on the help button provided by the wrapper, a modal window opens where we can read about this block, what it does and how to work with it, and typical errors that occur. The interface implements the endpoints of the server part, namely:

Has access to telemetry search and display it as a graph or map, as well as saving search parameters in key-value storage

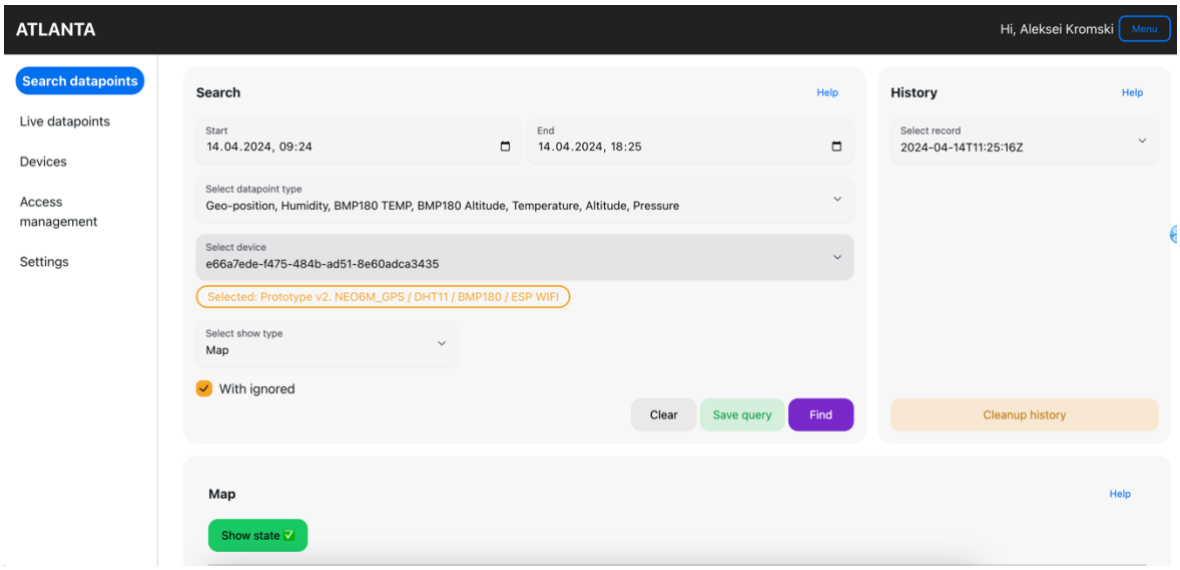



*Figure 29 Web application, search page with history functionality*

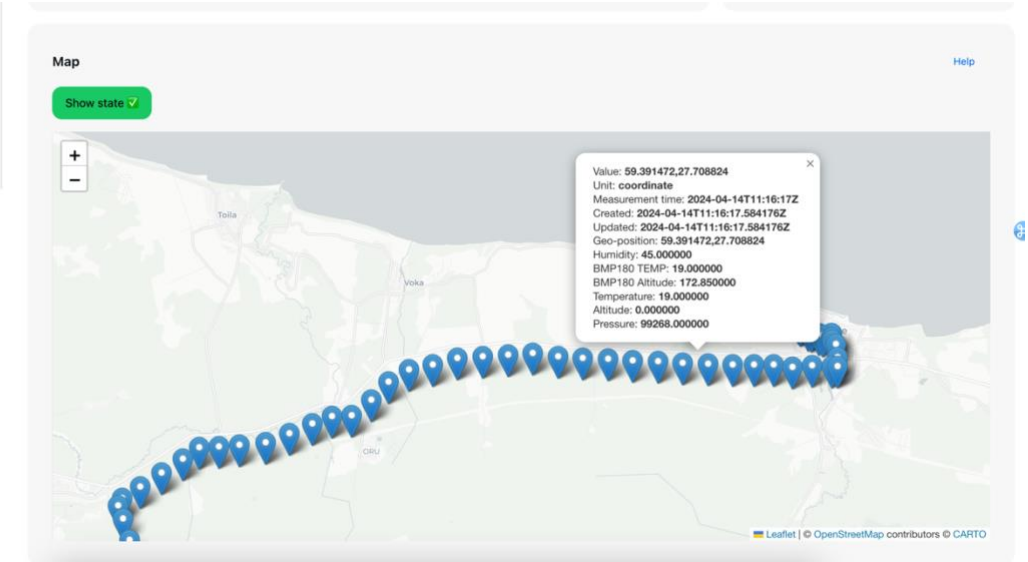



*Figure 30 Web application, search result in map representation*

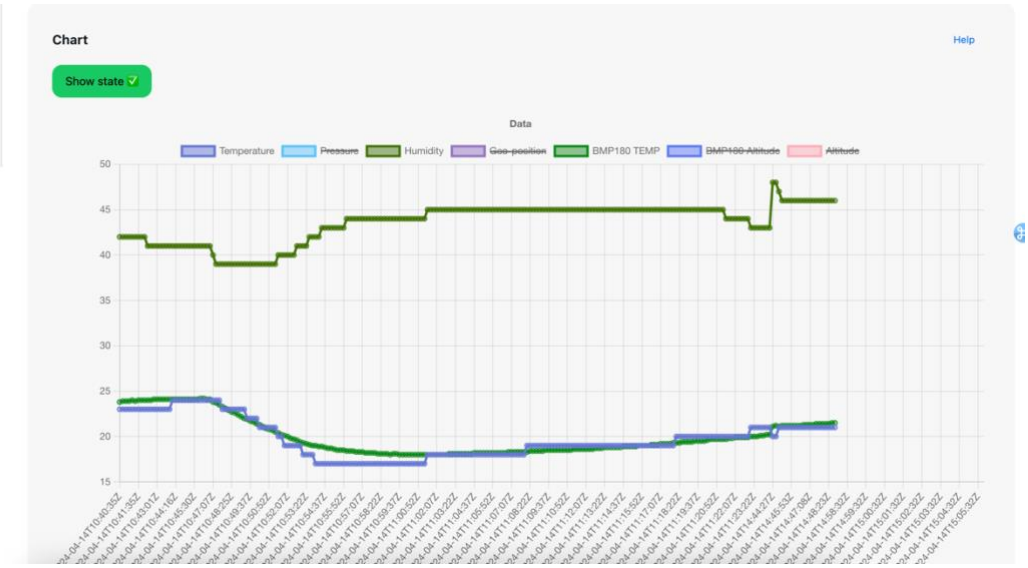



*Figure 31 Web application, with graph representation*

View data in Live mode by connecting to the server via websocket. As soon as we have new incoming telemetry, the interface will allow us to select the device to which the telemetry was received and view it in three modes: namely, map, table and graph

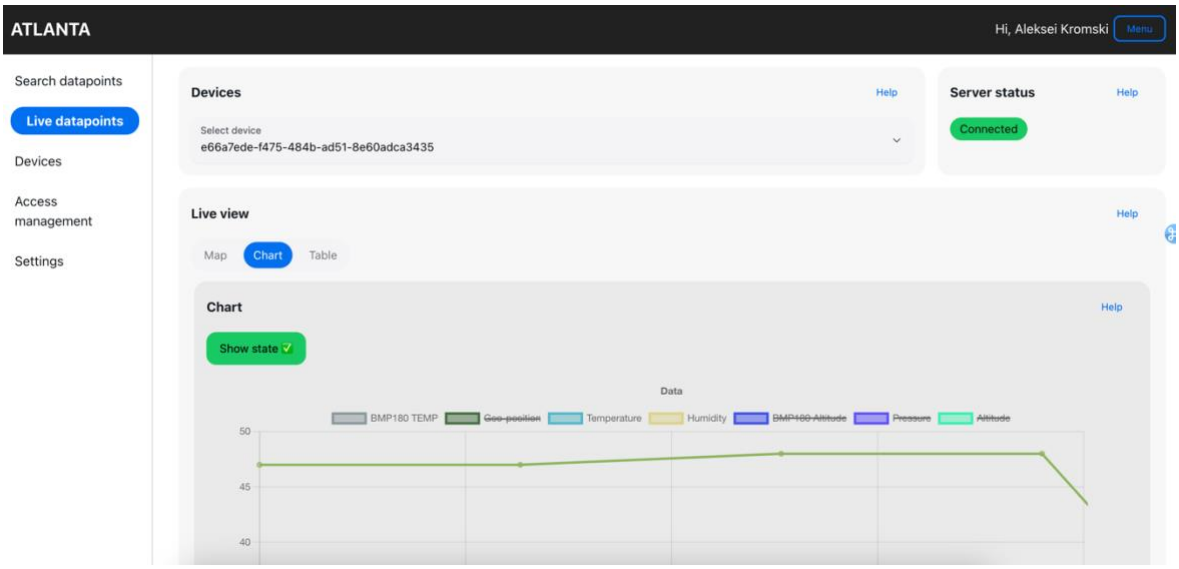



*Figure 32 Web application, live preview of incoming data*

Device management, namely: creation, deletion, updating

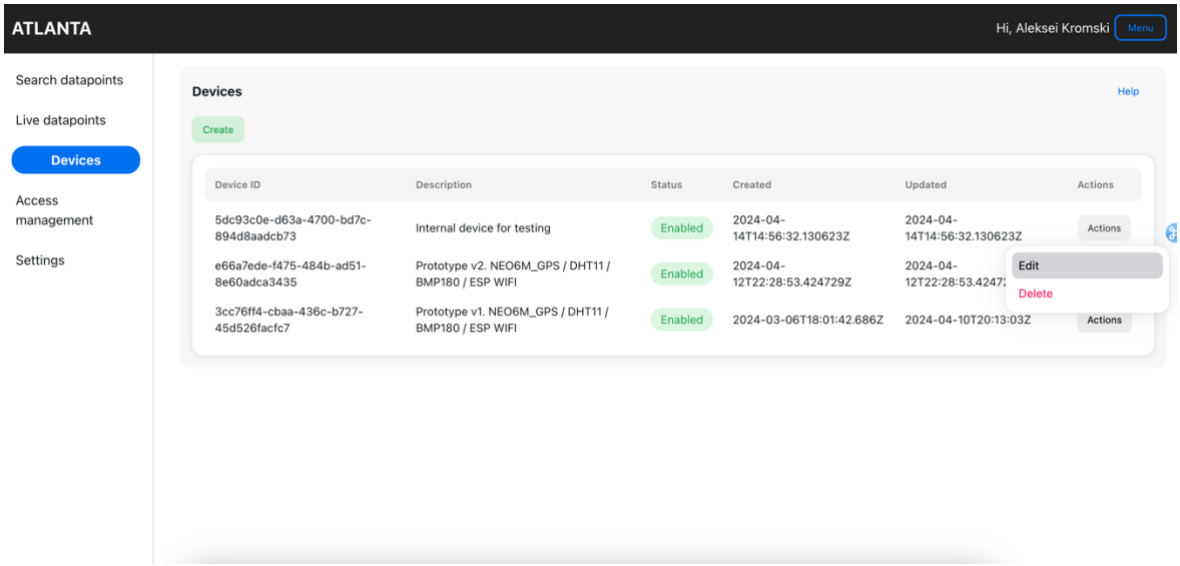



*Figure 33 Web application, device control system*

Management of endpoints, roles and users. The administrator can assign his own list of available endpoints to each role, thus limiting access

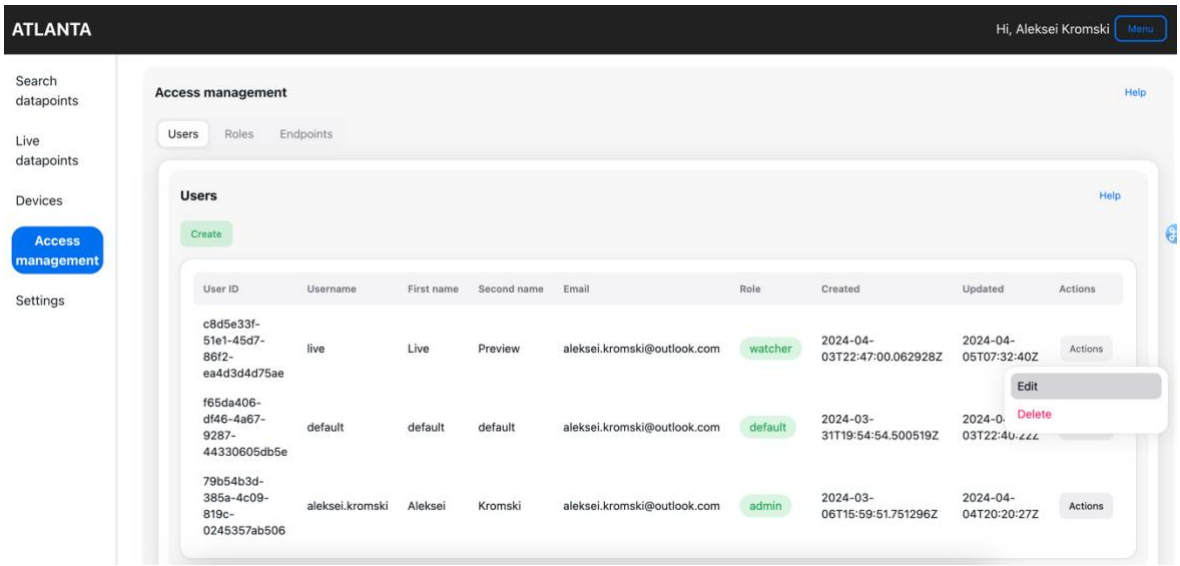



*Figure 34 Web application, management of users*

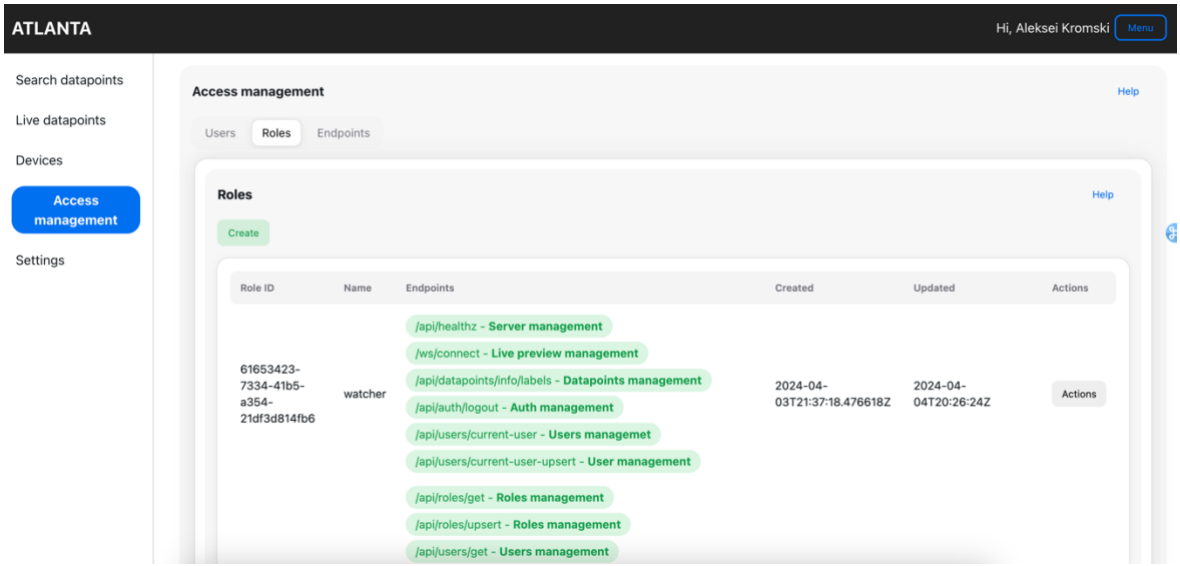



*Figure 35 Web application, management of roles*

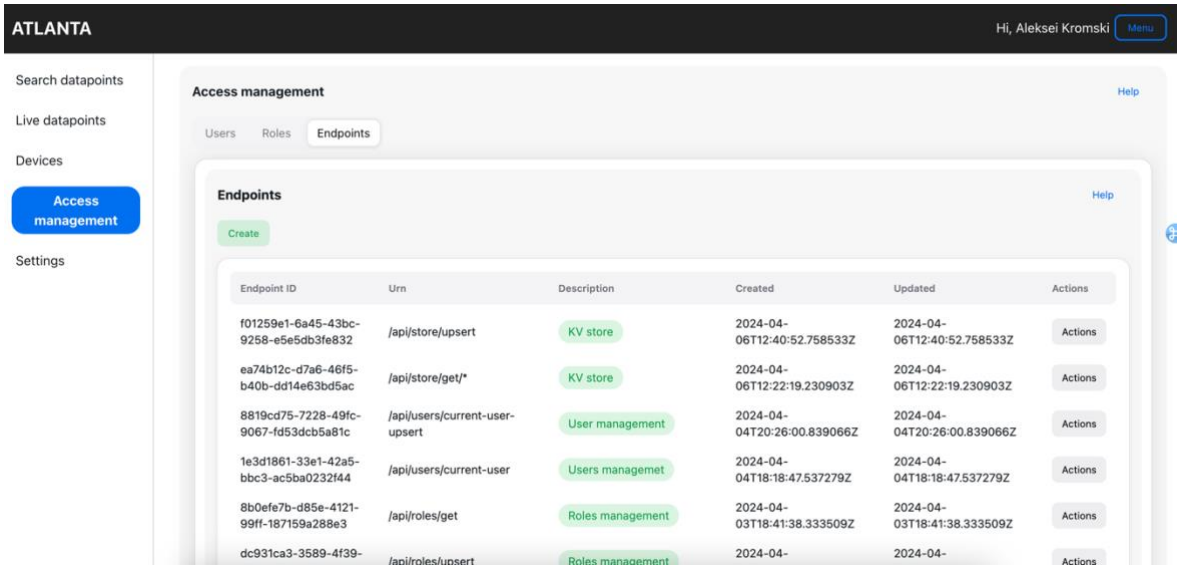



*Figure 36 Web application, management of endpoints*

Apply device to special external api endpoint

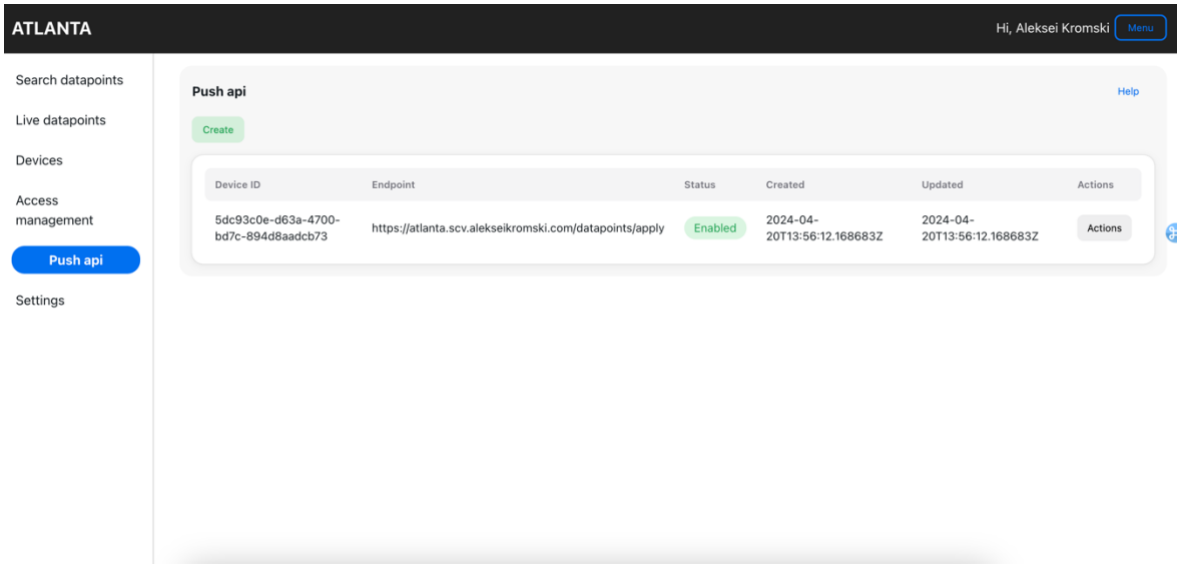



*Figure 37 Web application, push api management system*

Editing a logged-in user

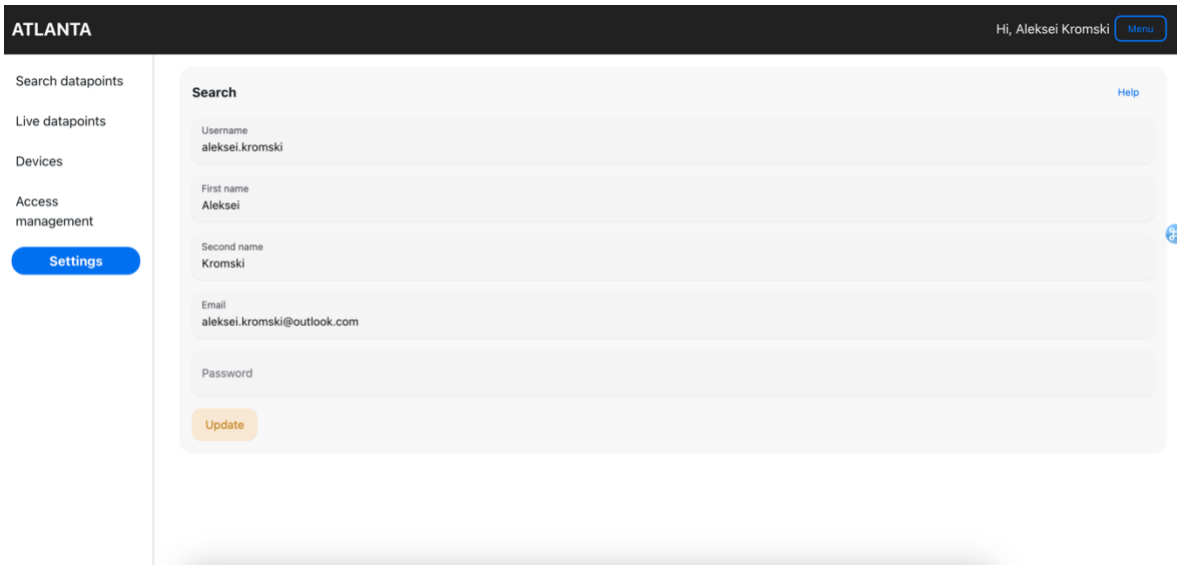



*Figure 38 Web application, logged-in user settings*

# FUTURE OF THE PROJECT

In the future, this project may have constant development and distribution by using the «fork» functionality on Github, namely the cloning of the main repository and the addition of new functionality in its repository, for example, a large number of companies uses this GitHub function using it on any open-source software for its improvement and optimization for the needs of your own system. The advantage of my project is that it is very easy to expand with new modules using the above-described module system in the core. Thus, small companies can take a ready-made project as a basis and start developing it if necessary, while large companies can gradually implement modules for a smooth transition to a new system. For example, a large company with some functionality can send a copy of incoming data from the device and see what will be at the output, this approach will reduce the risks when introducing a new standard

# CONCLUSION

The purpose of this thesis is to develop affordable Internet of Things (IoT) device and server software to improve supply chain visibility in logistics for large and small businesses. This IoT device and server software was designed to collect and analyze critical environmental and geographic data for use by logistics companies.

The IoT device was successfully built using NEO-6M GPS module, GY-68 BMP180 sensor, SIM800L cellular network module, DHT11 sensor, MT3608 converter and Arduino UNO R3 with Wi-Fi module. An important part is that a comparative analysis of existing IoT devices and server software on the market from various companies around the world was carried out. The results of the benchmarking analysis showed that the developed IoT device and server application were competitive, offering extensive functionality and ease of scalability

During the development of this project, unexpected problems arose, namely the limitation of the use of the IoT device by the legacy 2G network. However, this was corrected by the presence of a Wi-Fi module in the board, which provided more stable indicative results. One of the main goals of the project was to bridge the gap between small and large businesses in the logistics industry, who can easily apply the new open standard to fulfill their needs

In the end, I can say that the project has successfully achieved its given goal of developing a cost-effective IoT device and logistics backend application. The solution has been able to outperform itself in benchmarking, solving problems faced by small and large businesses by offering the best functionality and resource optimization at a relatively low cost in development and project support. The main discovery was that this project can be easily integrated into other areas of activity, for example: analysis of weather conditions, construction and compliance with norms, smart cities and agriculture. Since its flexible system does not prevent it from being used as originally intended, which makes it a universal device

The application could be found on Github:

- Server application - https://github.com/AlekseiKromski/atlanta-server
- Firmware - https://github.com/AlekseiKromski/atlanta-firmware

# REFERENCES


[1] Interface ublox NEO-6M GPS Module with Arduino, https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/ (19.04.2024)

[2] Siregar, B., Andayani, U., Bahri, R. P., & Fahmi, F. (2018, March). Real-time monitoring system for elderly people in detecting falling movement using accelerometer and gyroscope. In *Journal of Physics: Conference Series* (Vol. 978, No. 1, p. 012110). IOP Publishing.

[3] Interface BMP180 Barometric Pressure & Temperature Sensor with Arduino, https://lastminuteengineers.com/bmp180-arduino-tutorial/ (19.04.2024)

[4] Send Receive SMS & Call with SIM800L GSM Module & Arduino, https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/ (19.04.2024)

[5] Sugiyanti, I. (2019). Design of ATM crime monitoring system based on MQTT protocol using SIM800L and Arduino Mega 2560.

[6] Interface DHT11 Module With Arduino, https://lastminuteengineers.com/dht11-module-arduino-tutorial/ (19.04.2024)

[7] Kuria, K. P., Robinson, O. O., & Gabriel, M. M. (2020). Monitoring temperature and humidity using Arduino Nano and Module-DHT11 sensor with real time DS3231 data logger and LCD display.

[8] MT3608 Max DC-DC 2V-24V 2A Step Up Converter Booster Power Module, https://www.robotbanao.com/products/mt3608-max-dc-dc-2v-24v-2a-step-up-converter-booster-power-module (19.04.2024)

[9] ATmega328p + ESP8266 WIFI tutorial, https://forum.arduino.cc/t/atmega328p-esp8266-wifi-tutorial/944380 (19.04.2024)

[10] Bradley, L. J., & Wright, N. G. (2020). Optimising SD saving events to maximise battery lifetime for Arduino™/Atmega328P data loggers. *IEEE Access*, *8*, 214832-214841.

[11] Mehta, M. (2015). ESP8266: A Breakthrough in wireless sensor networks and internet of things. *International Journal of Electronics and Communication Engineering & Technology*, *6*(8), 7-11.

[12] 5/10/20M UL1007 Single Core Copper Wire 26 24 22 20 18 16 14 AWG PVC Insulation Solid Tinned Plating Line Electric Cable Line, https://www.aliexpress.com/item/1005006106330815.html?spm=a2g0o.cart.0.0.522c38dazVHzgM&mp=1 (19.04.2024)

[13] Handy Portable Resistor Kit for Arduino Starter Kit UNO R3 LED potentiometer tact switch pin header, https://www.aliexpress.com/item/32238063369.html?spm=a2g0o.cart.0.0.522c38dazVHzgM&mp=1 (19.04.2024)

[14] Makettplaat 2*(45*45mm) 228+213 auku eraldi, https://www.oomipood.ee/product/pc_02lam_makettplaat_2_45_45mm_426_auku_eraldi (19.04.2024)

[15] Brand new 100% original 18650 MJ1 3.7V 3500mAh lithium-ion rechargeable battery, https://www.aliexpress.com/item/1005006494150346.html?spm=a2g0o.productlist.main.1.a10fAOVrAOVrwb&algo_pvid=b5dba5be-1e6c-4ee8-8724-5cbbc2f91e6e&algo_exp_id=b5dba5be-1e6c-4ee8-8724-5cbbc2f91e6e-0&pdp_npi=4%40dis%21EUR%218.98%214.49%21%21%2167.65%2133.82%21%402101efab17133727303154869ef726%211200003740872032 7%21sea%21EE%21215154699 2%21&curPageLogUid=DDnrbqI27rbI&utparam-url=scene%3Asearch%7Cquery_from%3A (19.04.2024)

[16] Send Receive SMS & Call with SIM800L GSM Module & Arduino, https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/ (19.04.2024)

[17] Dynamic global network infrastructure with self-configuration and interoperable communication (Ali, Z. H., Ali, H. A., & Badawy, M. M., 2015, p. 37)

[18] Freichel, S. L., Rütten, P., & Wörtge, J. K. (2022). Challenges of supply chain visibility in distribution logistics–a literature review. Ekonomski vjesnik: Review of Contemporary Entrepreneurship, Business, and Economic Issues, 35(2), 453-466.

# APPENDICES

**Appendix 1.**

| Task name | Due |
|---|---|
| Server: Prepare project deployment env | November 13, 2023 |
| Server: Create basic modules: tcp_consumer / http server / storage | November 21, 2023 |
| Server: Create basic core system | November 22, 2023 → November 24, 2023 |
| Firmware: to do the desoldering of modules on the board | November 22, 2023 → November 24, 2023 |
| Server: Create requirement tree for modules in core | November 27, 2023 → November 28, 2023 |
| Firmware: connect all modules to the Arduino Board | November 27, 2023 → November 28, 2023 |
| Server: Create storage interface | November 28, 2023 → November 29, 2023 |
| Server: Create postgres implementation of storage interface | November 29, 2023 → December 14, 2023 |
| Server: Add migration system for postgress | December 15, 2023 |
| Server: Create adapter to parse incoming string in tcp_consumer over TCP | December 18, 2023 → December 22, 2023 |
| Firmware: write code for getting all data (Atmega chip) | December 18, 2023 → December 22, 2023 |
| Server: Create search telemetry endpoint | December 18, 2023 → December 29, 2023 |
| Server: Create device management endpoint | December 29, 2023 → January 3, 2024 |
| Server: Create user permissions management endpoint | January 3, 2024 → January 8, 2024 |
| Firmware: write code for transmit data to server (WiFi) | January 3, 2024 → January 8, 2024 |
| Firmware: write code for transmit data to server (SIM800L) | January 4, 2024 → January 8, 2024 |
| Firmware: connect 2 microcontroller and testing | January 4, 2024 → January 25, 2024 |
| Server: Create history management endpoint | January 8, 2024 → January 13, 2024 |
| Server: Create Guard system to auth on server | January 16, 2024 → January 25, 2024 |
| Server: Create login / logout enpoint | January 26, 2024 → January 30, 2024 |
| Server: Implement at-socket-server go lib to have WebSocket functionality | January 31, 2024 → February 13, 2024 |

| | |
|---|---|
| Server: Create basic front-end functionality | February 14, 2024 → February 21, 2024 |
| Front-end: Search page | February 21, 2024 → February 28, 2024 |
| Front-end: Live preview page | February 28, 2024 → March 7, 2024 |
| Front-end: device management | March 8, 2024 → March 12, 2024 |
| Front-end: users & permissions management | March 12, 2024 → March 19, 2024 |
| Front-end: current user management | March 19, 2024 → March 27, 2024 |
| Front-end: login page | March 27, 2024 → April 4, 2024 |
| Server: Push-api: receive data from tcp_consumer and send it to external api | March 27, 2024 → April 4, 2024 |
| Server: Create push api endpoint | March 27, 2024 → April 4, 2024 |
| Front-end: create push api management system | March 27, 2024 → April 4, 2024 |

*Figure 39 A table that interprets all completed tasks for Firmware, Server Software, Web Application*

**Appendix 2.**

| Product | Cost (EUR) | Shipping (EUR) | Date | Link |
|---|---|---|---|---|
| GY-NEO6MV2 - GPS module | 2.62 | 1.13 | 13.11.23 | https://www.aliexpress.com/item/1005001635722164.html?spm=a2g0o.order_list.order_list_main.5.2d2d1802FMTGVO |
| GY-68 BMP180 - Pressure module | 0.54 | 0.11 | 13.11.23 | https://www.aliexpress.com/item/1005004860318570.html?spm=a2g0o.order_list.order_list_main.10.2d2d1802FMTGVO |
| SIM800L | 3.40 | free | 13.11.23 | https://www.aliexpress.com/item/1005002532480516.html?spm=a2g0o.order_list.order_list_main.9.69bb1802s85Q8q |
| DHT11 - Temperature sensor | 0.76 | 0.50 | 13.11.23 | https://www.aliexpress.com/item/32840892862.html?spm=a2g0o.order_list.order_list_main.20.2d2d1802FMTGVO |
| Arduino UNO R3 + WiFi module | 5.67 | 1.64 | 13.11.23 | https://www.aliexpress.com/item/32848546164.html?spm=a2g0o.order_list.order_list_main.34.2d2d1802FMTGVO |
| Box | 4.82 | free | 30.01.24 | https://www.aliexpress.com/item/1005005622148025.html?spm=a2g0o.cart.0.0.522c38dazVHzgM&mp=1 |
| MT3608 DC-DC Step Up | 10.24 | free | 30.01.24 | https://www.aliexpress.com/item/1005006035788311.html?spm=a2g0o.cart.0.0.522c38dazVHzgM&mp=1 |
| Resistor Kit | 2.17 | 1.70 | 30.01.24 | https://www.aliexpress.com/item/32238063369.html?spm=a2g0o.cart.0.0.522c38dazVHzgM&mp=1 |
| Cooper wire | 9.87 | 0.42 | 30.01.24 | https://www.aliexpress.com/item/1005006106330815.html?spm=a2g0o.cart.0.0.522c38dazVHzgM&mp=1 |
| Total (eur) | 40.09 | 5.5 | | |

*Figure 40 Table of modules with price and shipping price (with tax)*

# LICENCE

**Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks**

Mina, **Aleksei Kromski**

Annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **An empirical comparison and construction of iot appliances for supply chain visibility in logistics**

mille juhendaja on **Chen-Wan Lin**

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

- Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
- Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
- Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

*Aleksei Kromski*

***16.05.2024***