Security tracker for cars

João Almeida nº87583
Afonso Costa do vale nº87648

January 2022

Abstract

This paper describes the implementation and development of a solution for automobile-based tracking and security that was developed using Arduino and Lora-Wan.

The device has 2 modes - Tracking and Security mode - the first allows the user to locate his automobile in real time, storing it for visualisations and analysis, the second alarms the user if his car moves which is potentially helpful in the case of theft.

Keywords: lora-wan arduino iot gps tracking node-red dashboard

Contents

1 Introduction

IoT can help us solve many problems that exist nowadays. One of them is car burglary.

According to Motor24, 36 cars are stolen every day in Portugal[6] and 125 per year per 100,00 inhabitants. In comparison to the rest of Europe, Portugal appears to be relatively safe in 12th place.

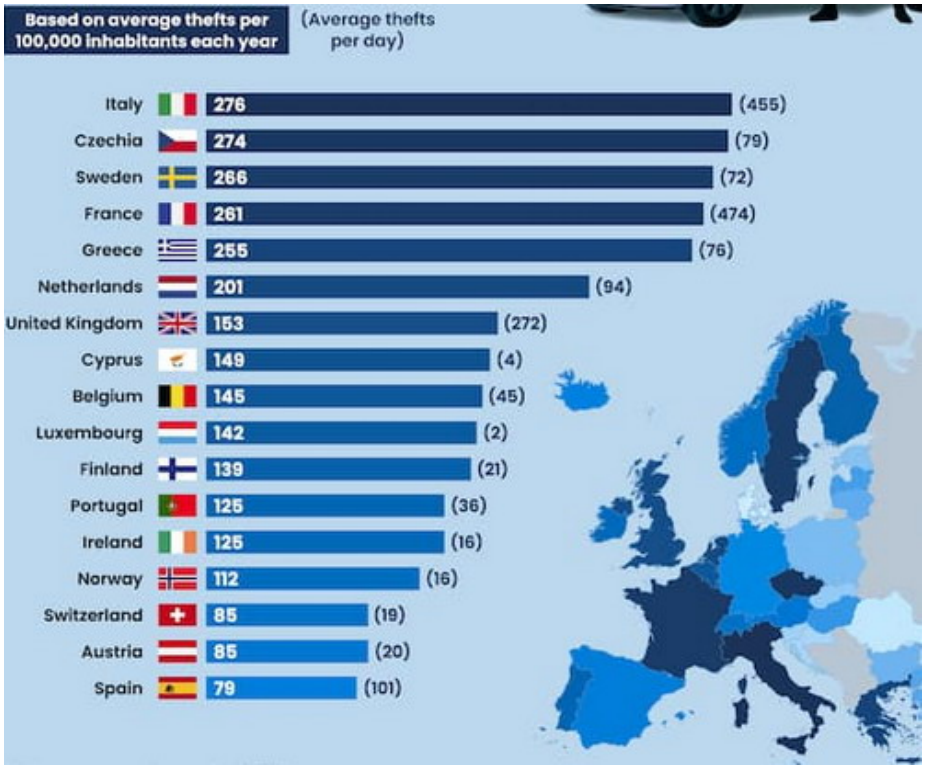


Figure 1: Stolen cars per country in Europe

Even though Portugal occupies the bottom half of the table, it's still a problem worth considering in Portugal and in the rest of Europe.

According to Medium, the average cost of a new car in Portugal in recent years is 29529€ [7].

This is a very substantial amount, meaning that in Portugal, on average around 531,522€ are stolen as cars everyday. (Calculated using 50% of the value of newer cars, to roughly account for depreciation. Calculation is as follows: $36 * (29529/2)$).

That's why we developed a solution to track and detect this kind of burglary to try to help solving this problem. Our device aids in the location of the vehicles

and alerts us to any unusual movements.

Our GPS module helps us track the vehicles, and our acceleration sensor helps us notice any unwanted movements from them.

2 Related Work

GPS trackers have many possible usages in many fields of interest such as transportation, military and health care. It is of core importance for location based services.

Therefore, many articles and projects have been done regarding this subject because of its major importance nowadays.

GPS trackers are something that is very much covered in the scientific literature currently.

In a project[4], a solution for tracking car speeds with a GPS tracker and the Internet-of-Things platform was developed.

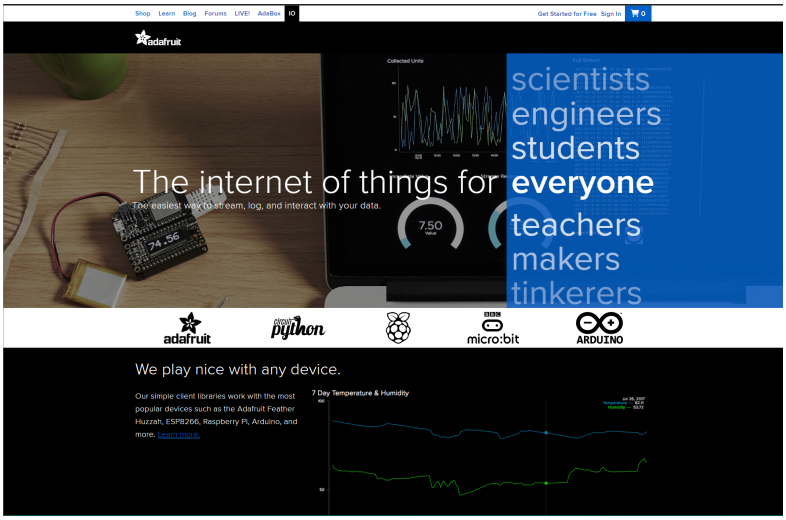
In this solution, the speed was obtained using the difference between the actual position and the last recorded position within that time span.

$$speed = \frac{distance}{deltaTime}$$

For example, if an object moved from point A to point B in 1s and assuming $dist(A, B) = 2m$, then the speed would be:

$$speed = \frac{2}{1}$$

The position and speed would be sent to a dedicated IoT platform(Adafruit IO[1]) which processes the information to be displayed *"in the monitor of personal computer or Android based mobile phone."*[4].



Another project involving GPS tracking[3] uses this technology to track the real time position of a patient to provide emergency services.

3 System Description

3.1 General Overview

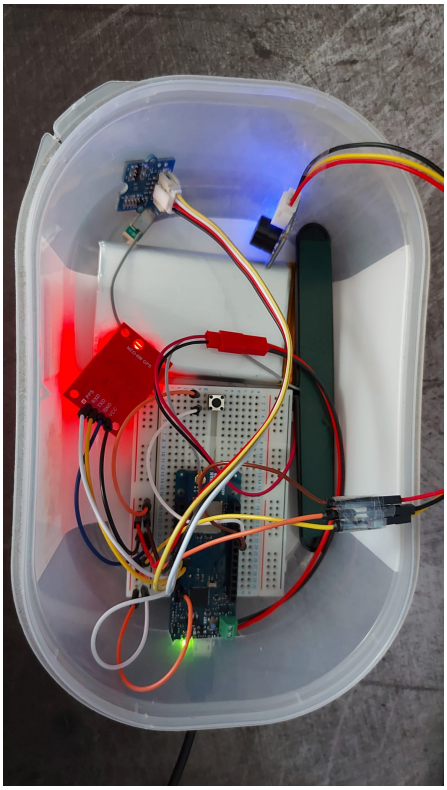


Figure 2: Top down view of implementation

This project is meant to serve as a protection and data gathering device for a personal car which it achieves using two different operational modes that the user switches manually pressing a button on the device, using the on board led to indicate what is the current state of the device, led On for tracking mode and led OFF for Alert mode.

Tracking mode In this mode, the device activates its tracking module, periodically registering its current position and communicating said data with our server, where that data is stored and used for visualizations, statistics or even machine learning algorithms.

Alert mode In Alert mode the device activates its Gyroscopic sensor and detects any sudden movement, upon which it will communicate this information to the server. Said server will send email notifications to the configured address, making sure the owner of the valuable item knows that the item is in motion.

3.2 "The things network"

is a Lora-WAN Network Server which enables a free and easy to use solution for Lora in any project. [8] Although it is not without its limita-

tions to bandwidth and rate at which the messages are sent but, for a free offering it is very useful for prototyping projects and developing quick solutions.

By registering our device on "The Things Network" website it is able to send and receive messages, called up-link and down-link messages.

Our device is coded to periodically listen for down-link messages coming from the Lora network (this frequency is customizable in a single constexpr unsigned long variable), which ultimately come from our back-end on node-red. It will also send up-link messages, the frequency and types of messages depend on the state that the device finds itself on.

The code used for sending and receiving messages was heavily inspired from the code provided in the examples from the Arduino mkr wan 1300 board. Some simple changes were made, such as returning the received message, if the was one, so that the main loop of the Arduino code can parse the received message and manage the state of the app.

This allows us to send up-link messages at will in our code, with more flexibility and only treat received down-link messages in a streamlined way.

Downlink The things network, not only allows us to receive messages from the device (UpLink) but also send data to it. This is what is called a Downlink message, and there are three different classes of Lora devices, A , B and C, the majority of devices being of class A, which is the default and most energy efficient, due to it only being able to read messages from the network after sending an uplink message. Due to this, every uplink message the device sends, it also reads the Downlink queue (where the downlink messages are stored) and if there is any message, it is read and parsed to change the state of the app.

3.3 Hardware

Micro-controller As a micro - controller we are using the Arduino mkr wan 1300 due to its "out of the box" communication features.

Since the project was, from the beginning, envisioned to use LoRa Wan as the means of communicating between the device and the server this model was an easy choice, as it comes prepared with a Lora wan pin where an antenna can simply be plugged in to communicate with the network.

Grove 3-axis digital compass The Grove 3-axis digital compass v2.0 module is activated, allowing for measurements of the magnetic field in three perpendicular axes. These readings are then read through the SCL and SDA pins of the Arduino mkr wan 1300 board, and using the bmm150 library for Arduino we read the data from the module.

Security / Alert Mode In The Alert mode, the 3-axis digital compass is active, each reading that is done with the module updates the "previous" variables, the values read in the last iteration are stored (they are named with "previous_" as a prefix).

With the following code we compare the values read in the last iteration with the ones in the latest iteration to detect sudden changes in movement, if these changes are too abrupt and they pass the threshold defined in "movement_variation_limit" an alarm variable will be set to true which will then cause the alarm to be activated.

When the alarm is set off, an alarm message will be sent through Lora to our node-red "back-end"

```
if (
  abs(abs(previous_headingDegrees) - abs ( headingDegrees ) )
      > movement_variation_limit ||
  abs(abs(previous_xyHeadingDegrees) - abs ( xyHeadingDegrees ) )
      > movement_variation_limit||
  abs(abs(previous_zxHeadingDegrees) - abs ( zxHeadingDegrees ) )
      > movement_variation_limit
  ) {
  should_alarm = true;
}
```

Buzzer An active Buzzer module is used as a physical alarm when the Grove 3-axis module detects sudden movement and starts the alarm state.

This buzzer is fairly easy to connect to the Arduino. It only needs a VCC, a GND and a Sig connection, VCC and GND are simple, vcc for energy and gnd for ground, Sig is connected to a digital pin on the arduino, where or code will interact with the module.

To make the buzzer play a sound, the "tone" method must be used, it received the pin where the Sig inpt of the buzzer is connected to, a frequency and a duration in milliseconds, where the duration is optional. If no duration is specified, the buzzer will continue playing until the noTone() method is called on the same pin.

With this in mind, many popular songs can easily be played with the device, as some of them being freely available as open source code, for example on robsoncouto's github [2].

Tracking For tracking, a Neo-6M GPS module is being connected to the board through the TX and RX pins, where serial communication can happen between the Arduino and the module.

The module itself has a small *"U.FL"* connector for an antenna that the user can replace, being able to easily invest in a more powerful antenna for more precision in the latitude and longitude readings.

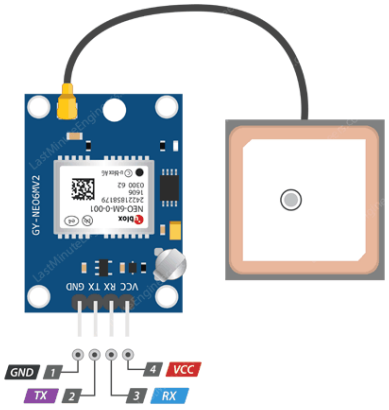


Figure 3: neo-6m gps module

This antenna is what makes the module work. It detects and connects to nearby satellites, calculating its current position using theirs. A process that is called Trilateration. The more connected satellites the better for the precision of the read coordinates.

To visually give some feedback on its current state, the module features a red led light that, if is completely off means that the module is searching for satellites and if it is blinking every one second means that a "*Position Fix*" is found, in other words, enough satellites were found and calculated position is accurate. Information inspired from: lastminuteengineers tutorial on neo6m gps with arduino [5].

We created a method that is used to read the raw serial data from the TX and RS pins that come from the neo-6m module, encoding that data into a gps object from the "TinyGPS++" library which provides many useful functions like getting the latitude, longitude, number of satelites currently connected and more information.

With all this information available, it still needs to be transmitted to the server, to do that we aggregate that data into a String that obeys json rules:

{"s":0,"l": {"lat":38.70451,"lng":-9.20228,"alt":78.20000}}

We have two keys, "s" and "l", "s" being the number of satelites and "l" being a nestes json which represents location, inside it has "lat" for latitude, "lng" for longitude" and "alt" for altitude.

This message is converted into bytes, represented below as hexadecimal values and sent through the lora network.

7B 22 73 22 3A 30 2C 22 6C 22
3A 20 7B 22 6C 61 74 22 3A 33
38 2E 37 30 34 35 31 2C 22 6C
6E 67 22 3A 2D 39 2E 32 30 32
32 38 2C 22 61 6C 74 22 3A 37
38 2E 32 30 30 30 30 7D 7D

3.4 Software

3.4.1 Node-red

Node red is a low code platform built using JavaScript, justifying its web based interface.

It's low code approach using nodes that connect to each other, each with different functionalities provide great flexibility and speed of prototyping.

What improves it even further are it's packages. With them, added functionality can be plugged in.

The following is a list of all the external packages the project depends on:

Dependencies

- node-red-node-sqlite
- node-red-dashboard
- node-red-contrib-web-worldmap
- node-red-node-email
- node-red-contrib-whatsapp-cmb

Main Flow

MQTT integration The Things network website, can relay the messages received from the deployed devices through the LoraWan network into a Mqtt web based channel that can then be read with any device using an internet connection.

Node-red has multiple mqtt nodes readily available, where messages can be read and sent to topics

Dashboard The "node-red-dashboard" module can be used to easily create a web dashboard consisting of buttons, charts, gauges and more.

The nodes from this module are divided between ones for acting such as buttons, drop-downs, sliders, and others for displaying information, such as gauges, charts, text and more.

Each node can belong to a **group** inside a **Tab**. These tabs give a way to have multiple pages inside our dashboard, each page with its own set of groups.

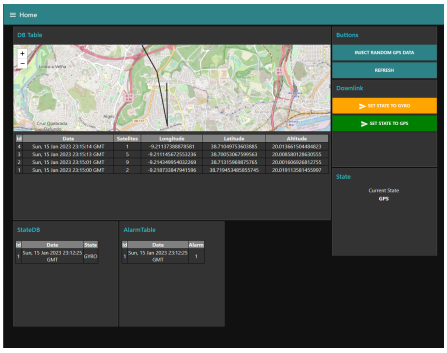


Figure 4: node-red map example

Each group can hold multiple "widgets", a widget being a button or a chart, or a gauge, etc.

When a tab has multiple groups inside, the space is divided vertically and the widgets are shown linearly in a "column-style" layout.

Our Implementation of the dashboard Is organised in two separate tabs, a main tab for seeing all the new data coming, visualising the map and also buttons to send **Downlink** messages to change the current device's state.

Worldmap The "node-red-contrib-web-worldmap" module integrates very well with the previously mentioned "dashboard" module to insert interactive maps into the created dashboard.

A map is being integrated in order to show the movements of the target. While the locations are being received through the mqtt channel, the map is updated with each new record, showing a trail that indicates the last positions of the device.

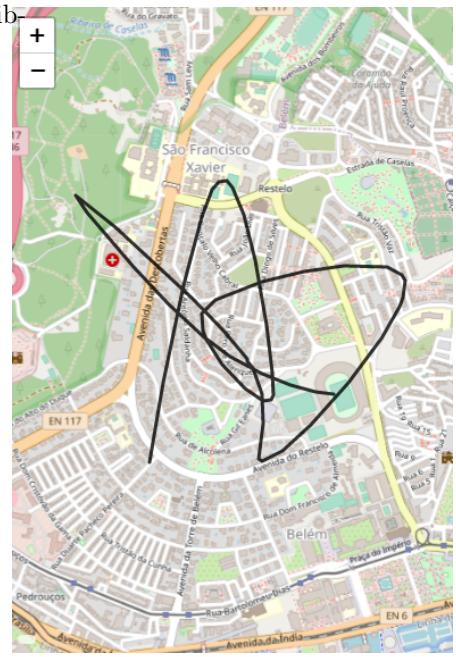


Figure 5: node-red map example

SQLite A persistent storage option is crucial for any solution.

In this project SQLite is being used, due to our familiarity with the technology and its strict structure on the data that it receives. This strict approach may lead to more development time but leads to a more robust project in the end with more potential to grow and increase in features.

Database Architecture

The project has three different types of data that it collects:

- Gps location data;
- App state data;
- Alarm data;

Since these three can happen in different times and totally asynchronous, one SQL table was created for each one of these categories. All of the tables feature an Auto Increment Integer ID that serves as the primary key and a date field that registers the date at which the message was received.

The **gps** table has three columns besides the ones mentioned, latitude, longitude and altitude. The **alarm** table has an additional column for storing if the alarm was happening. The **state** table has an additional column for storing if the state of the app, true for Gyro and false for GPS

Email Notification An email notification will be sent when the hardware detects an intruder. This detection is done by the 3 axis digital model when a certain acceleration is detected and then a message is sent via Lora to Node-red. Then the email notification is triggered on node-red.

Whats-app Notification Similarly to our email notification, a message through "Whats-app" will be sent via a bot to the configured phone number when an alert reaches Node Red An email notification will be sent when the hardware detects an intruder. This detection is done by the 3 axis digital model when a certain acceleration is detected.

3.4.2 Arduino Code

We divided our code into multiple files. Our main file is called gps_lora_project.ino. We also have a file called arduino_secrets.h where our APP Key and EUI are saved. Each Arduino component has its on header(.h) and implementation file(_inl.h).

A **header file** is where the declaration of the methods lies. The **implementation** counter part is where the methods declaration lies. Components files:

1. **3-Axis Digital Compass v2** - axis.h and axis_inl.h
2. **Active Buzzer** - buzzer.h and buzzer_inl.h
3. **Nneo-6m gps** - gps.h abd gps_inl.h
4. **LoRa-Wan** - lora.h and lora_inl.h

This keeps our code organised, clean and ready for future expansions.

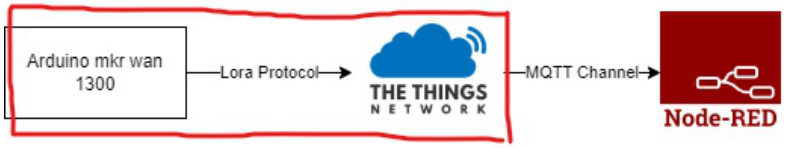
3.5 Testing

Each part of the tech stack of this project can be tested separately.

Testing Arduino software

Our project features a "testbed" folder for each different module where test scripts where used to test and implement each different module of the Arduino hardware. For example: The buzzer and the Gyroscope have these files separate.

Testing Arduino Lora connection

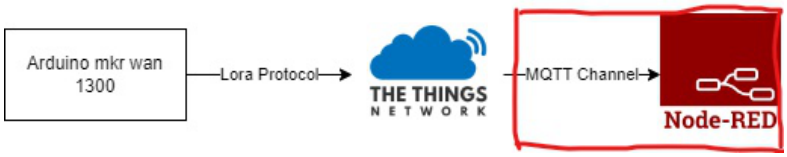


Since Lora communication was very crucial for the project as a whole, it was the first module to be tested. For this, various example scripts can be found online where simple messages are sent from the Arduino to the Lora network.

These scripts were kept all throughout the development of the project, being used as a basis or what the final project turned into.

As some modules had their challenges, namely the GPS module, test messages were sent instead, to not slow the development of the project and to test the whole architecture and flow of data. From Arduino to Lora, to MQTT, to Node-red (and sometimes to email).

Testing Node Red back-end



Using the native MQTT nodes from node-red and some very valuable tools from The Things Network, tools where up-link and down-link messages can be simulated into the network, the node-red can read these messages and act as if they were sent from the device itself, allowing the testing of the visualisations and database.

4 Conclusion

The project served as a very insightful learning experience, about the limitations of gps and lora technology and how to organize code in the iot and embedded systems ecosystem.

The shortcomings of Lora and its inconsistency make it a free and low power network but in cases where reliability and speed are crucial, it is simply overshadowed by much more powerful technology like sms and mobile data (ex: 4g).

Node-red proved to be a reliable tool to make small full-stack prototypes but for large systems, the connections between nodes can become messy and unreadable.

4.1 Future Work

Modes of the device Currently the user switches modes manually through the press of a button but this could be improved upon, for example, the node-red backend could use a downlink message to communicate to the device that it should switch modes, this way the user could operate the device remotely. Even turning on GPS mode in the case of theft, having the possibility of knowing the location of the item.

Communication The project suffered from the LoRa's unreliability, testing the whole app was tedious, as many times the messages would get blocked from the network.

Battery Getting a good battery to power the device even when not connected to a computer would be a very good and simple addition, as it would allow for easier use of the device as a whole. This was not implemented during development due to time constraints and issues with on of the batteries that we had for testing.

References

[1] *Adafruit*. English. URL: https://io.adafruit.com/.

[2] *Arduino-songs*. English. URL: https://github.com/robsoncouto/arduino-songs.

[3] Pratik Kanani and Mamta Padole. "Real-time Location Tracker for Critical Health Patient using Arduino, GPS Neo6m and GSM Sim800L in Health Care". In: *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*. 2020, pp. 242–249. DOI: 10.1109/ICICCS48265.2020.9121128.

[4] Asep Najmurrokhman et al. "Design and Implementation of Vehicle Speed Recorder using GPS Tracker and Internet-of-Things Platform". In: *2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST)*. 2021, pp. 152–156. DOI: 10.1109/ICAICST53116.2021.9497797.

[5] *Neo6m-gps-arduino-tutorial*. English. URL: https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/.

[6] *Os países onde se roubam mais carros na Europa*. Portuguese. URL: https://www.motor24.pt/noticias/os-paises-onde-se-roubam-mais-carros-na-europa/1544299/.

[7] *The Real Cost of Owning a Car — EU*. English. URL: https://roadzen.medium.com/the-real-cost-of-owning-a-car-eu-efa814dde33a/.

[8] *Thethingsnetwork*. English. URL: https://www.thethingsnetwork.org/.

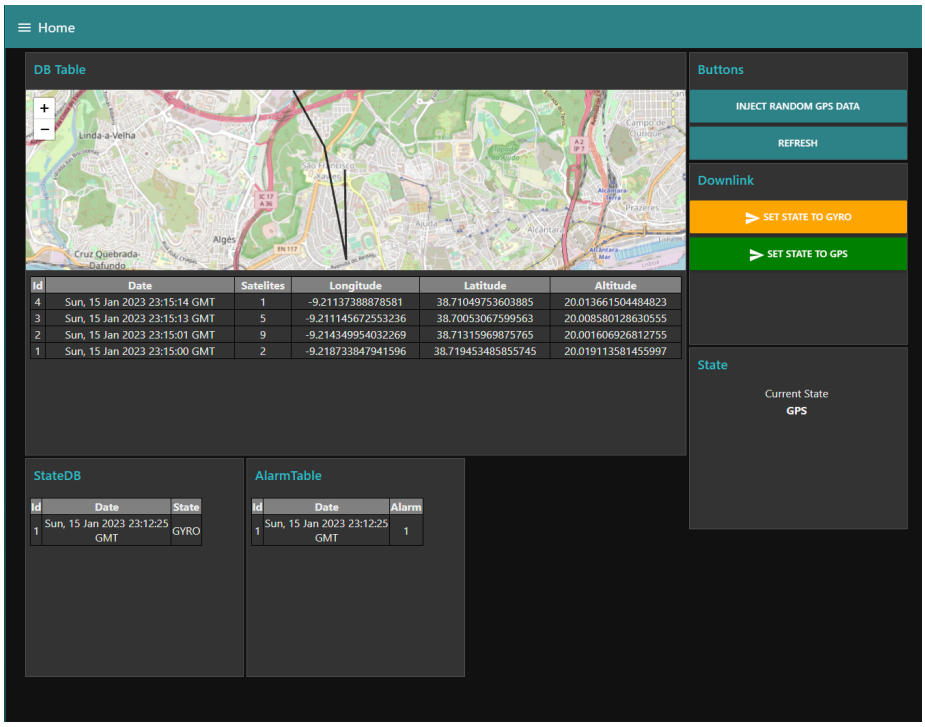


Figure 6: node-red dashboard

A Appendix

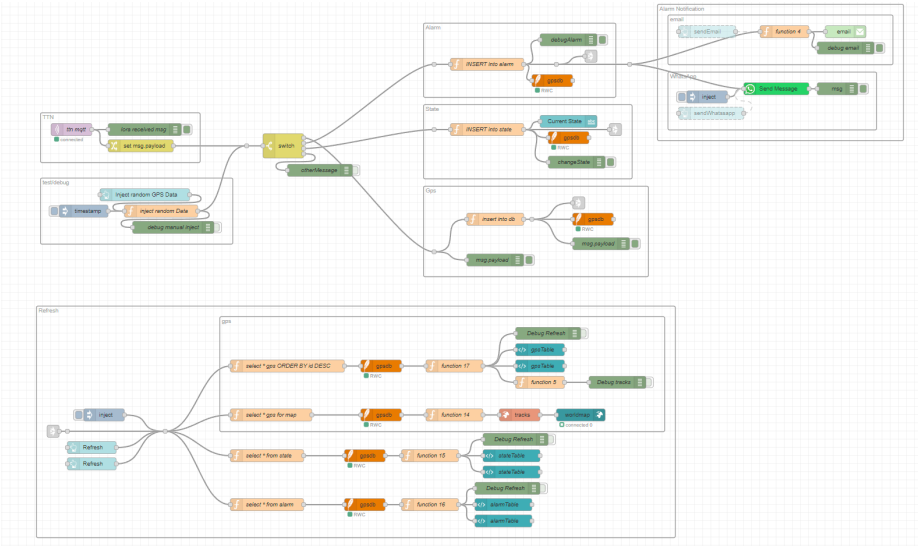


Figure 7: Flow 1 from Node Red, dealing with lora and inserting data into database

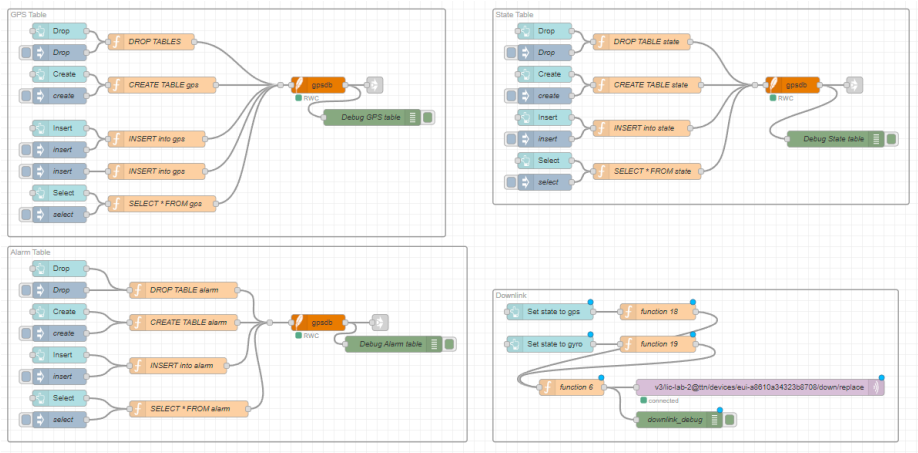


Figure 8: Flow 2 from Node Red, creating, dropping and testing tables in sqlite database