Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

LoRaWAN Lamb Link

Robert Muldoon

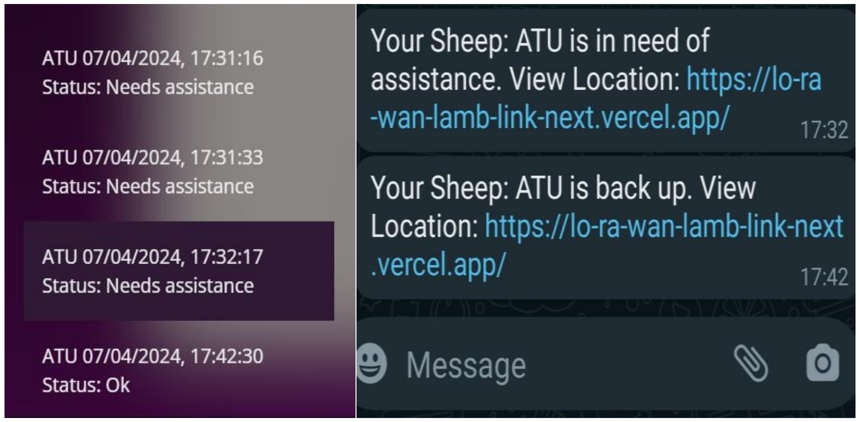
G00395612

BEng(H) in Software & Electronic Engineering

Project Engineering

Atlantic Technological University

2023/2024



ATU 07/04/2024, 17:31:16
Status: Needs assistance

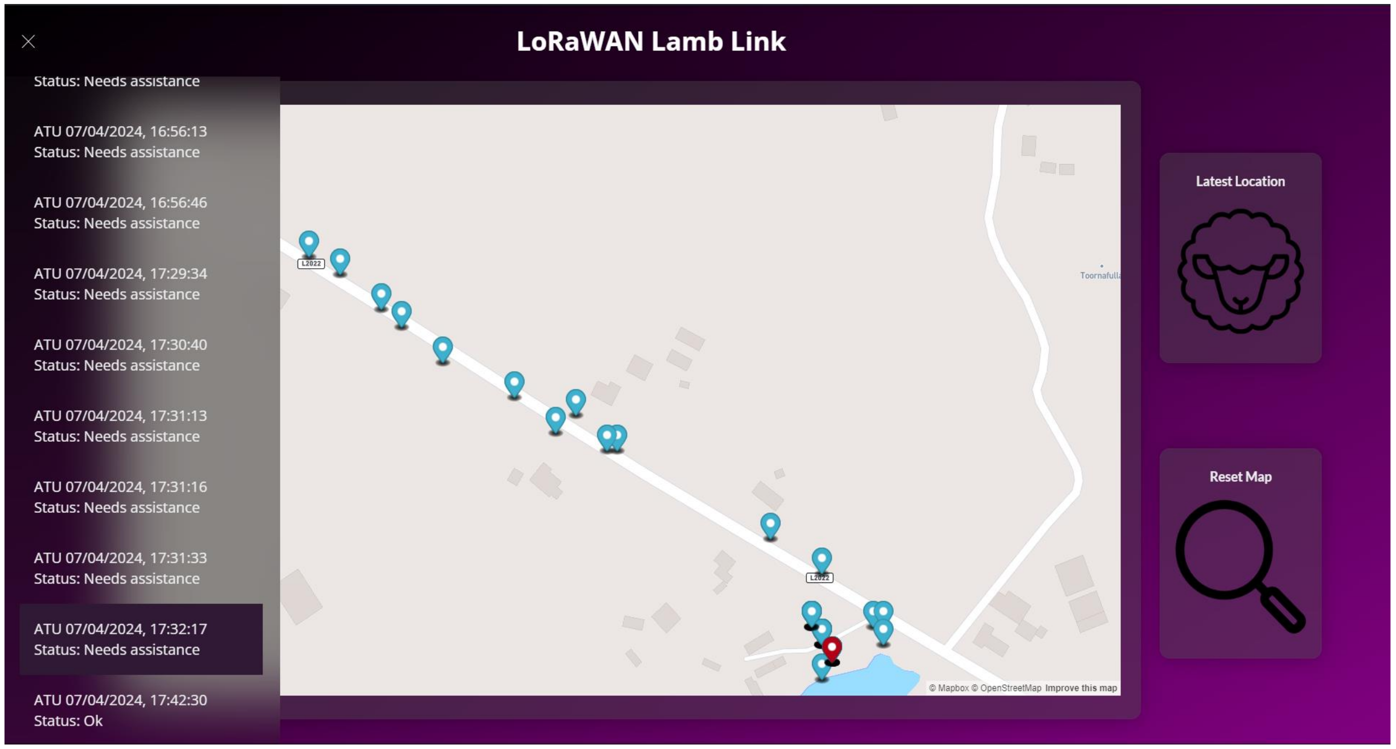
ATU 07/04/2024, 17:31:33
Status: Needs assistance

ATU 07/04/2024, 17:32:17
Status: Needs assistance

ATU 07/04/2024, 17:42:30
Status: Ok

Your Sheep: ATU is in need of assistance. View Location: https://lo-ra-wan-lamb-link-next.vercel.app/ 17:32

Your Sheep: ATU is back up. View Location: https://lo-ra-wan-lamb-link-next.vercel.app/ 17:42



Status: Needs assistance

ATU 07/04/2024, 16:56:13
Status: Needs assistance

ATU 07/04/2024, 16:56:46
Status: Needs assistance

ATU 07/04/2024, 17:29:34
Status: Needs assistance

ATU 07/04/2024, 17:30:40
Status: Needs assistance

ATU 07/04/2024, 17:31:13
Status: Needs assistance

ATU 07/04/2024, 17:31:16
Status: Needs assistance

ATU 07/04/2024, 17:31:33
Status: Needs assistance

ATU 07/04/2024, 17:32:17
Status: Needs assistance

ATU 07/04/2024, 17:42:30
Status: Ok

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Atlantic Technological University.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

Signed: Robert Muldoon

Acknowledgements

I would like to acknowledge and thank the academic staff at ATU who have provided me with support and guidance for the duration of my studies. In particular, I would like to thank Niall O'Keefe, who as my supervisor has continuously provided me with technical feedback and support. I received the lend of Robustel LG5100 from TITAN Group and received helpful Robustel LG5100 information directly from Robustel employees through email.

Table of Contents

1 Summary

LoRaWAN Lamb Link aims to support European farmers in preventing forest fires by making use of LoRaWAN and GNSS tracking technology on sheep to monitor grazing areas. The project's primary goal is to track sheep movement using a GNSS tracker and visualise areas where they might have grazed with a map component. This assists farmers and potentially governments in determining areas that may be prone to fires by identifying neglected regions. Additionally, accelerometer data is used to alert the farmer to a sheep in need of assistance.

The project utilises a Robustel LG5100 LoRaWAN gateway which features a ChirpStack server to establish a LoRaWAN network. GNSS and accelerometer data from the LoRaWAN node are transmitted to the gateway, which communicates with ChirpStack. ChirpStack processes the data and sends it to the Spring backend hosted on an EC2 AWS server. The backend using RESTful API calls adds, retrieves and deletes data from a MongoDB database. Twilio integration enables the system to send WhatsApp alerts to the farmer when a sheep is detected to need assistance.

The frontend, developed with Next.js and hosted on Vercel, provides a user-friendly interface for viewing and managing sheep data.

Key Features:

- **GNSS tracking of sheep.**
- **Visualisation of data on a map.**
- **Integration of Twilio for WhatsApp alerts when an accelerometer detects sheep in distress.**
- **Robustel LG5100 LoRaWAN gateway and ChirpStack server for LoRaWAN network infrastructure.**
- **Backend hosted on EC2 AWS server with MongoDB database and Frontend developed with Next.js and hosted on Vercel.**

This project successfully achieved all that it was set out to do. It established a functional LoRaWAN network and utilises both GNSS and accelerometer data to provide practical assistance and provides a user-friendly frontend to visualise and manage the data.

2 Poster

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

LoRaWAN Lamb Link

Robert Muldoon

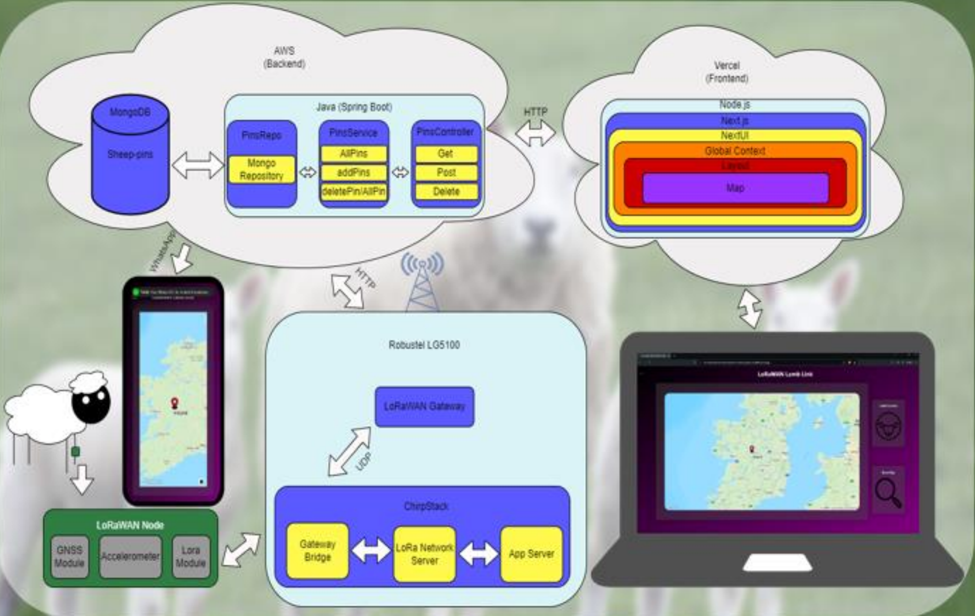
B.Eng Software and Electronic Engineering

Introduction

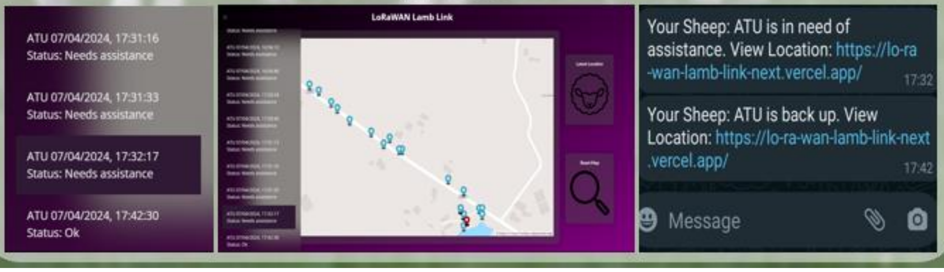
LoRaWAN Lamb Link aims to map out the location sheep have grazed in by using a device that features GNSS as well as LoRaWAN, a low power and long-range wireless technology. An accelerometer will detect a sheep in need of assistance and will alert the farmer through a WhatsApp message.

The STEVAL ASTRA1B is the LoRaWAN node used in this project. It will send the data collected from the GNSS and accelerometer components to the LoRaWAN Gateway at set intervals.

A Robustel LG5100 LoRaWAN gateway hosts a ChirpStack server which will receive data from the gateway and send it to a Spring Boot backend hosted on an EC2 AWS server. A WhatsApp alert is sent using Twilio when frames sent to the backend contain consecutive indication of a sheep in a compromised position. This backend features a MongoDB database for storing the data. The Next.js frontend hosted on Vercel uses RESTful API calls to access data from the backend and display it on a map component.



Results



ATU 07/04/2024, 17:31:16
Status: Needs assistance

ATU 07/04/2024, 17:31:33
Status: Needs assistance

ATU 07/04/2024, 17:32:17
Status: Needs assistance

ATU 07/04/2024, 17:42:30
Status: Ok

Your Sheep: ATU is in need of assistance. View Location: https://lo-ra-wan-lamb-link-next.vercel.app/ 17:32

Your Sheep: ATU is back up. View Location: https://lo-ra-wan-lamb-link-next.vercel.app/ 17:42

Technologies

- **LoRaWAN** to send data in an energy efficient manner.
- **C** to programme the LoRaWAN node.
- **Next.js** to develop the frontend.
- **Node.js** to execute frontend code.
- **Java** to develop Spring Boot backend.
- **AWS** to host the backend.
- **Vercel** to host the frontend.
- **MongoDB** for storing and retrieving sheep data.
- **Twilio** to send WhatsApp messages.

Features

- Complete LoRaWAN Network.
- User-friendly frontend scaled for PC monitor and mobile phone screens.
- Sheep location and well-being tracking.
- WhatsApp messaging including link to animal location when a sheep in need of assistance is detected.

3 Introduction

As global warming continues to warm the Earth, forest fires are becoming more and more prevalent. European countries such as France and Spain in particular are being affected. More than 119,000 hectares had already been impacted by the 18th of June according to the European Forest Fire Information System [1]. One potential solution to this problem that is currently being implemented by farmers in Spain is to allow herds of sheep and goats to graze in areas that are potentially at risk. By consuming all the dry vegetation in the area, the risk of a fire is greatly reduced along with other benefits such as carrying seed and fertilising the terrain as they move through it. The LoRaWAN Lamb Link aims to aid farmers in effectively managing grazing areas to reduce fire hazards.

The project aims to utilise LoRaWAN, accelerometer and GNSS tracking technology to monitor sheep movements and detect instances of distress. By tracking grazing patterns and identifying neglected areas, farmers can strategically manage areas that, as a result of neglect may have elevated levels of vegetation and therefore minimise fire risks.

The scope of the project encompasses the development of a comprehensive system comprising hardware, software, and cloud infrastructure. This includes implementation of a LoRaWAN network with a Robustel LG5100 gateway and ChirpStack server for data transmission, a backend on an EC2 AWS server for data processing and storage using Spring Boot and MongoDB, a user-friendly frontend interface developed with Next.js and hosted on Vercel for data visualisation and the integration of WhatsApp messaging using Twilio to alert the farmer of sheep in need of assistance.

This report will provide insight into the technical details of the LoRaWAN Lamb Link project. This includes the architecture, functionality, implementation process, challenges faced, and outcomes achieved.

4 Project Architecture

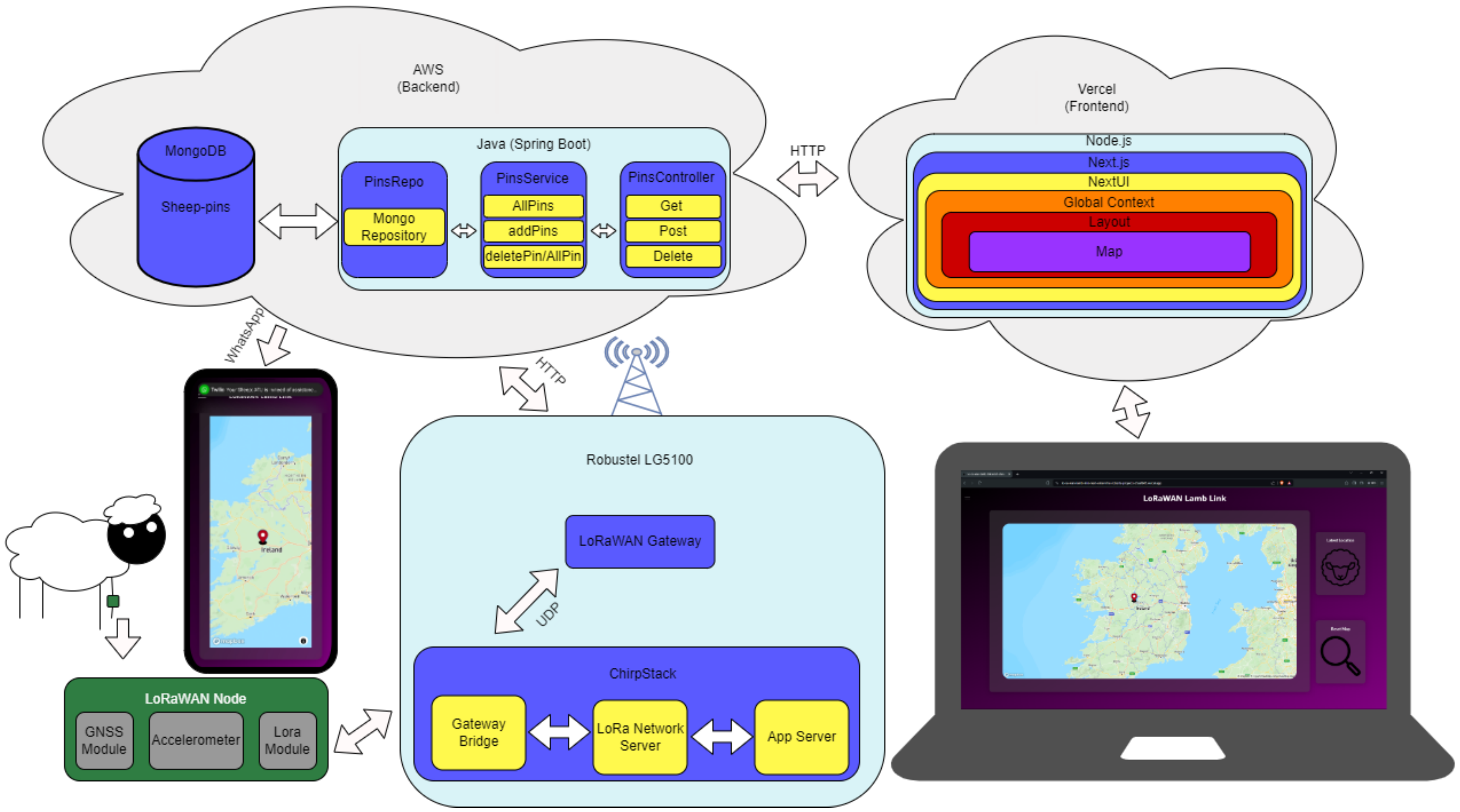


Figure 4-1 Architecture

5 Background (LoRa/LoRaWAN)

This project relies extensively on LoRaWAN technology. LoRaWAN is an ultra-low power wide area network built on top of Lora modulation. LoRa is a wireless modulation technique derived from Chirp Spread Spectrum technology. It encodes information on radio waves using chirp pulses [2], enabling long range communication with minimal power. This makes it well-suited for applications that transmit small data packets at low bit rates over large distances, consuming significantly less energy compared to traditional wireless technologies like Wi-Fi or cellular data.

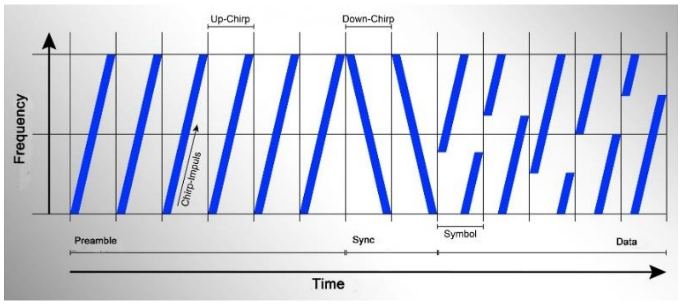


Figure 5-1 Chirp Spread Spectrum [3]

LoRaWAN serves as the Media Access Control (MAC) layer built on top of LoRa modulation. It is the software layer that defines how devices utilises the LoRa hardware, specifying parameters such as transmission timing and message formats. LoRaWAN gateways are capable of transmitting and receiving signals over a distance of over 10 kilometres in rural areas and 3 kilometres in dense urban areas [2]. This will enable the project to efficiently transmit location and accelerometer data while conserving energy.

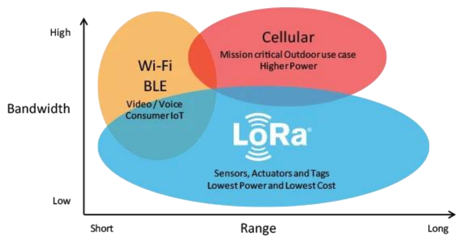

High
Bandwidth
Low
Cellular
Mission critical Outdoor use case
Higher Power
Wi-Fi
BLE
Video / Voice
Consumer IoT
LoRa
Sensors, Actuators and Tags
Lowest Power and Lowest Cost
Short
Range
Long

Figure 5-2 LoRa Range/Bandwidth [2]

6 Background (ChirpStack)

ChirpStack is an open-source LoRaWAN Network Server that enables communication between LoRaWAN devices and applications [4] . It features multiple components that are essential in the creation and operation of a LoRaWAN network. These include the Gateway Bridge, Network Server, Application Server and the ChirpStack Concentratord.

The ChirpStack Concentratord functions as the intermediary link between nodes and the gateway hardware, optimising data reception and routing to ensure efficient communication.

The Gateway Bridge ensures seamless communication between the gateway hardware and the Network Server. It plays an essential role in proficiently managing and transmitting data across the network.

The Network Server serves as the central hub overseeing the entire LoRaWAN network, responsible for tasks such as device activation, efficient routing and maintaining reliable two-way communication with nodes.

Finally, the Application Server serves as a crucial link connecting the network server to the external applications and services. This includes managing data processing, storage, and facilitating integration with databases and dashboards, ensuring seamless interaction between the LoRaWAN network and external entities. The Application servers HTTP integration feature will enable this project to send data from the LoRaWAN network to the Spring Backend.

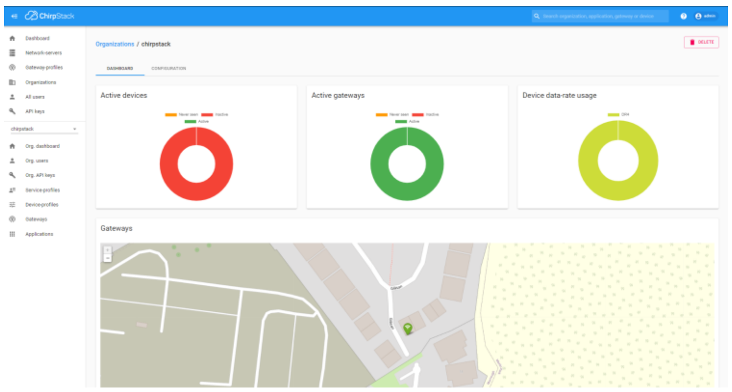


Figure 6-1 ChirpStack Application Server [4]

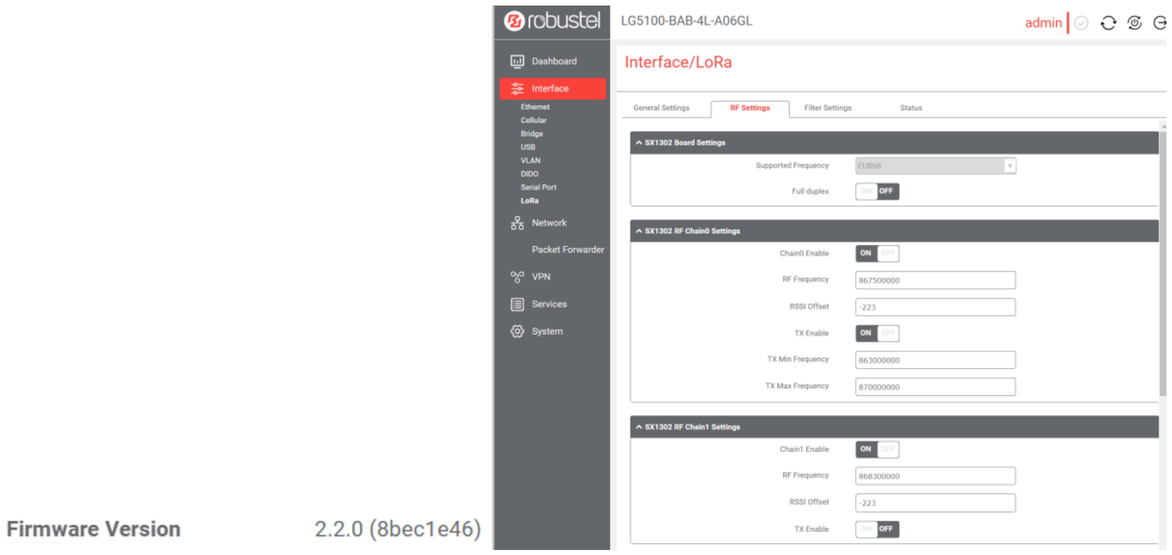
7 Background (Robustel LG5100)

The Robustel LG5100 industrial LoRaWAN gateway [5] plays a key role in the projects LoRaWAN network. It functions as not only a gateway to receive LoRaWAN frames but also features a Debian 11 based operating system running on an ARM Cortex-A7 based CPU. This in turn allows for the running of a ChirpStack server. Alternatively a ChirpStack could be setup on any machine running Debian or Ubuntu and a separate gateway could be configured for it, however this gives the project the benefit of unifying the collection, processing and later sending on of data into one device, simplifying the maintenance of the network.

Given that the Robustel LG5100 is a new and industry focused device online information on setting up the gateway was minimal and information on how to configure up the ChirpStack server was non-existent. In order to progress I got into direct contact with a Robustel employee who supplied useful information that I could interpret and adapt to meet the need of the project as well as an updated firmware that featured an integrated ChirpStack server. This integration has greatly simplified the setup from the original process I was working through.

Once the correct firmware is confirmed to be installed the correct LoRa radio frequency must be configured for the region the gateway is operating in. In the case of this project it is 868MHz.

This is done in Interface -> LoRa -> RF Settings.



SX1302 Board Settings

Supported Frequency	EU868
Full duplex	OFF

SX1302 RF Chain0 Settings

Chain0 Enable	ON
RF Frequency	867500000
RSSI Offset	-223
TX Enable	ON
TX Min Frequency	863000000
TX Max Frequency	870000000

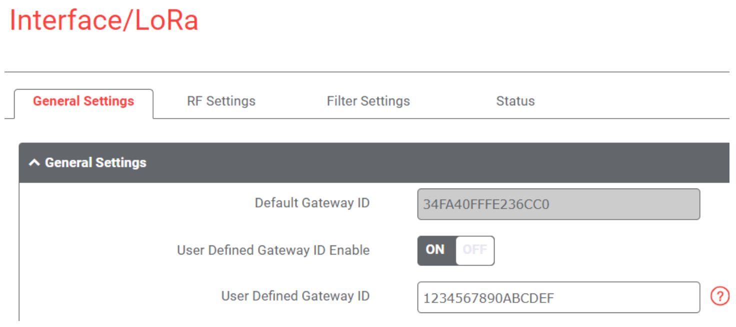
SX1302 RF Chain1 Settings

Chain1 Enable	ON
RF Frequency	868300000
RSSI Offset	-223
TX Enable	OFF

Firmware Version 2.2.0 (8bec1e46)

Figure 7-1 LoRa RF Settings

Optionally the Gateway ID can also be manually defined in the General settings tab.



General Settings

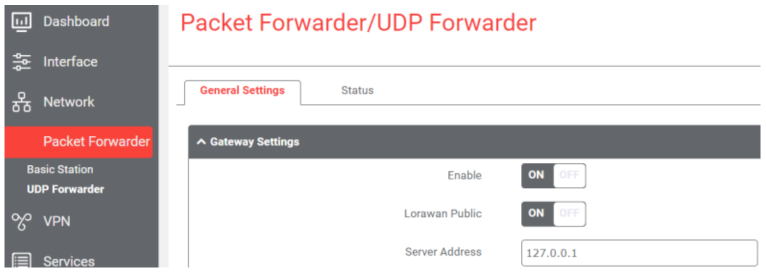
Default Gateway ID: 34FA40FFFE236CC0

User Defined Gateway ID Enable: ON

User Defined Gateway ID: 1234567890ABCDEF

Figure 7-2 LoRa General Settings

Next in Packet Forwarder -> UDP Forwarder make sure the gateway is enabled. Note the server address: 127.0.0.1



Gateway Settings

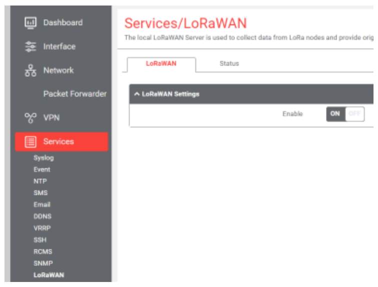
Enable: ON

Lorawan Public: ON

Server Address: 127.0.0.1

Figure 7-3 UDP General Settings

Finally within Services -> LoRaWAN enable LoRaWAN settings.



The local LoRaWAN Server is used to collect data from LoRa nodes and provide orig

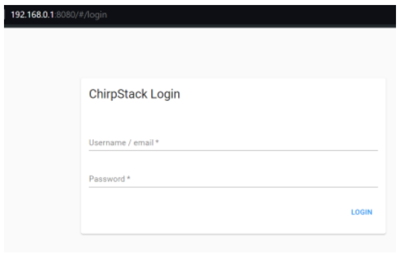
LoRaWAN Settings

Enable: ON

Figure 7-4 Services LoRaWAN

Now that the gateway is operating on the correct frequency and has been enabled the ChirpStack server can be accessed on http://192.168.0.1:8080/ with the default login.

Username: admin Password: admin


192.168.0.1:8080/#/login

ChirpStack Login

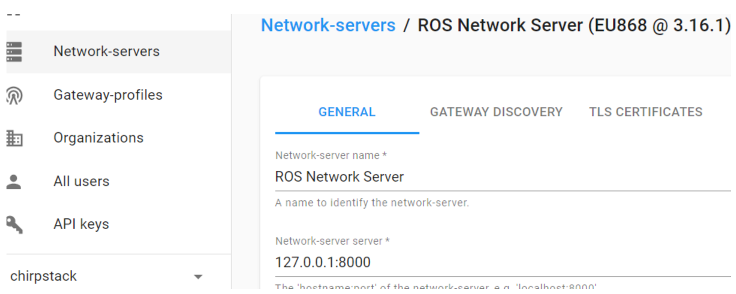
Username / email *

Password *

LOGIN

Figure 7-5 ChirpStack Login

Once logged in a Network-server can be added using the server address previously noted.



Network-servers / ROS Network Server (EU868 @ 3.16.1)

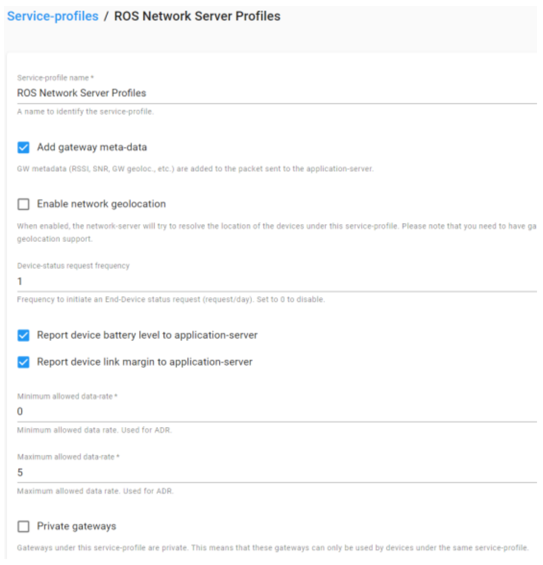
GENERAL | GATEWAY DISCOVERY | TLS CERTIFICATES

Network-server name *
ROS Network Server
A name to identify the network-server.

Network-server server *
127.0.0.1:8000
The 'hostname:port' of the network-server, e.g. 'localhost:8000'.

Figure 7-6 Network-servers

Next a server profile must be configured. Options such as gateway-data would be helpful in a larger network made up of multiple gateways.


Service-profiles / ROS Network Server Profiles

Service-profile name *
ROS Network Server Profiles
A name to identify the service-profile.

[✓] Add gateway meta-data
GW metadata (RSSI, SNR, GW geoloc., etc.) are added to the packet sent to the application-server.

[ ] Enable network geolocation
When enabled, the network-server will try to resolve the location of the devices under this service-profile. Please note that you need to have ga
geolocation support.

Device-status request frequency
1
Frequency to initiate an End-Device status request (request/day). Set to 0 to disable.

[✓] Report device battery level to application-server
[✓] Report device link margin to application-server

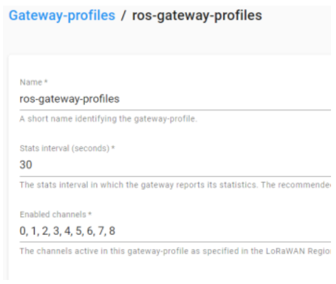
Minimum allowed data-rate *
0
Minimum allowed data rate. Used for ADR.

Maximum allowed data-rate *
5
Maximum allowed data rate. Used for ADR.

[ ] Private gateways
Gateways under this service-profile are private. This means that these gateways can only be used by devices under the same service-profile.

Figure 7-7 Service-profiles

Next a gateway profile must be configured. The number of channels the gateway supports is usually specified in the specifications within the user manual.


Gateway-profiles / ros-gateway-profiles

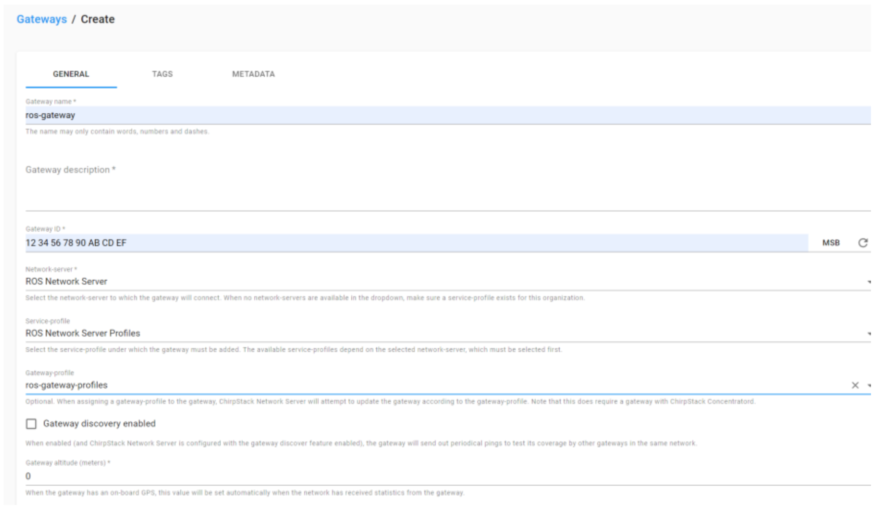
Name *
ros-gateway-profiles
A short name identifying the gateway-profile.

Stats interval (seconds) *
30
The stats interval in which the gateway reports its statistics. The recommende

Enabled channels *
0, 1, 2, 3, 4, 5, 6, 7, 8
The channels active in this gateway-profile as specified in the LoRaWAN Regio

Figure 7-8 Gateway-profiles

Following that the Robustel can be added to the ChirpStack server using the previously configured profiles. The gateway is identified using the Gateway ID which was manually defined in a previous step.



Gateway name *
ros-gateway
The name may only contain words, numbers and dashes.

Gateway description *

Gateway ID *
12 34 56 78 90 AB CD EF
MSB

Network-server *
ROS Network Server
Select the network-server to which the gateway will connect. When no network-servers are available in the dropdown, make sure a service-profile exists for this organization.

Service-profile
ROS Network Server Profiles
Select the service-profile under which the gateway must be added. The available service-profiles depend on the selected network-server, which must be selected first.

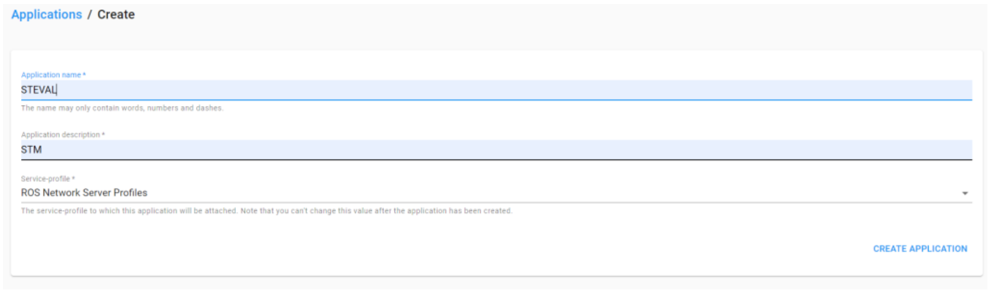
Gateway-profile
ros-gateway-profiles
Optional. When assigning a gateway-profile to the gateway, ChirpStack Network Server will attempt to update the gateway according to the gateway-profile. Note that this does require a gateway with ChirpStack Concentratord.

☐ Gateway discovery enabled
When enabled (and ChirpStack Network Server is configured with the gateway discover feature enabled), the gateway will send out periodical pings to test its coverage by other gateways in the same network.

Gateway altitude (meters) *
0
When the gateway has an on-board GPS, this value will be set automatically when the network has received statistics from the gateway.

Figure 7-9 Gateway Create

Next an application is required to contain the devices that will be sending data. The service profile is set to the previously created server profile to attach it with the projects server.



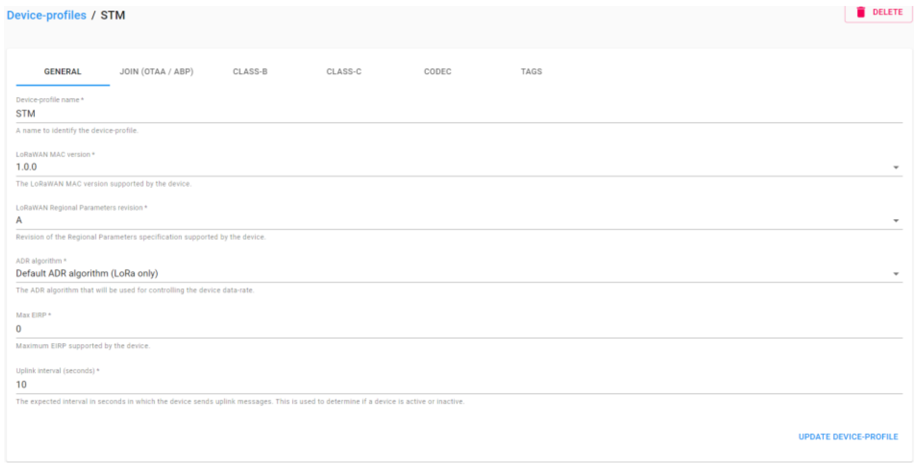
Application name *
STEVAL
The name may only contain words, numbers and dashes.

Application description *
STM

Service-profile *
ROS Network Server Profiles
The service-profile to which this application will be attached. Note that you can't change this value after the application has been created.

Figure 7-10 Application Create

Next the device profile must be created. These parameters can be adjusted to suit the requirements of the device. The codec this device uses to decode lora frames is also defined in the Device-profiles CODEC tab. The process of creating this codec is described in a later section.



Device-profile name *
STM
A name to identify the device-profile.

LoRaWAN MAC version *
1.0.0
The LoRaWAN MAC version supported by the device.

LoRaWAN Regional Parameters revision *
A
Revision of the Regional Parameters specification supported by the device.

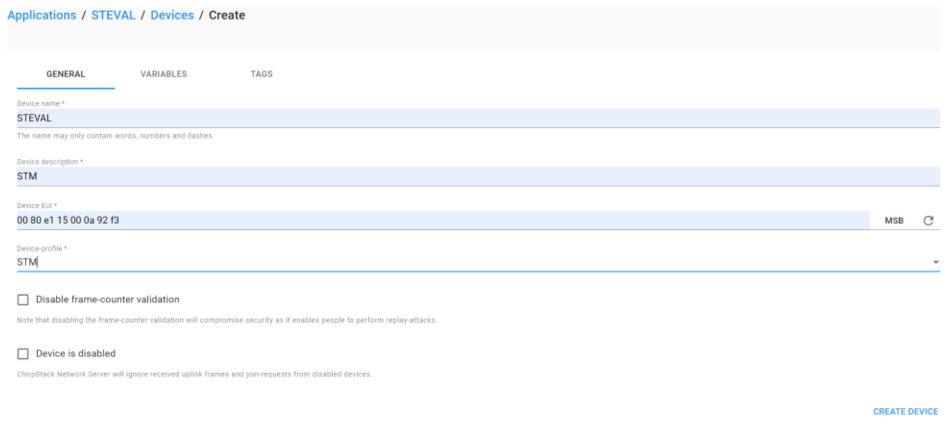
ADR algorithm *
Default ADR algorithm (LoRa only)
The ADR algorithm that will be used for controlling the device data-rate.

Max EIRP *
0
Maximum EIRP supported by the device.

Uplink interval (seconds) *
10
The expected interval in seconds in which the device sends uplink messages. This is used to determine if a device is active or inactive.

Figure 7-11 Device-profiles

Then the specific device is added to the Application section. To link the specific device to the Application server a Device EUI is required. Depending on the device this key may be provided by the manufacturer or it may require interfacing with the device through USB or wirelessly.



Applications / STEVAL / Devices / Create

GENERAL VARIABLES TAGS

Device name *
STEVAL
The name may only contain words, numbers and dashes.

Device description *
STM

Device EUI *
00 80 e1 15 00 0a 92 f3 MSB

Device-profile *
STM

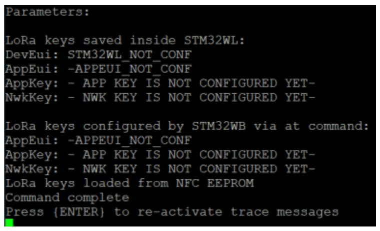
☐ Disable frame-counter validation
Note that disabling the frame-counter validation will compromise security as it enables people to perform replay-attacks.

☐ Device is disabled
ChirpStack Network Server will ignore received uplink frames and join-requests from disabled devices.

CREATE DEVICE

Figure 7-12 Devices Create

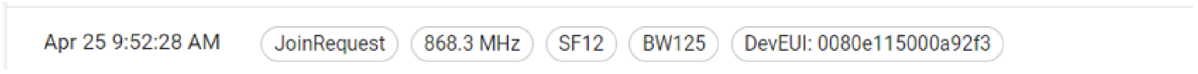
In the case of the LoRaWAN node selected for this project, the Device EUI is displayed upon interfacing through USB and reading what is displayed through a putty terminal. However as will be explained in the next section the original firmware had to wiped causing the Device EUI to not display.



```
Parameters:

LoRa keys saved inside STM32WL:
DevEui: STM32WL_NOT_CONF
AppEui: -APPEUI_NOT_CONF
AppKey: - APP KEY IS NOT CONFIGURED YET-
NwkKey: - NWK KEY IS NOT CONFIGURED YET-

LoRa keys configured by STM32WB via at command:
AppEui: -APPEUI_NOT_CONF
AppKey: - APP KEY IS NOT CONFIGURED YET-
NwkKey: - NWK KEY IS NOT CONFIGURED YET-
LoRa keys loaded from NFC EEPROM
Command complete
Press (ENTER) to re-activate trace messages
```

Figure 7-13 Undefined DevEui

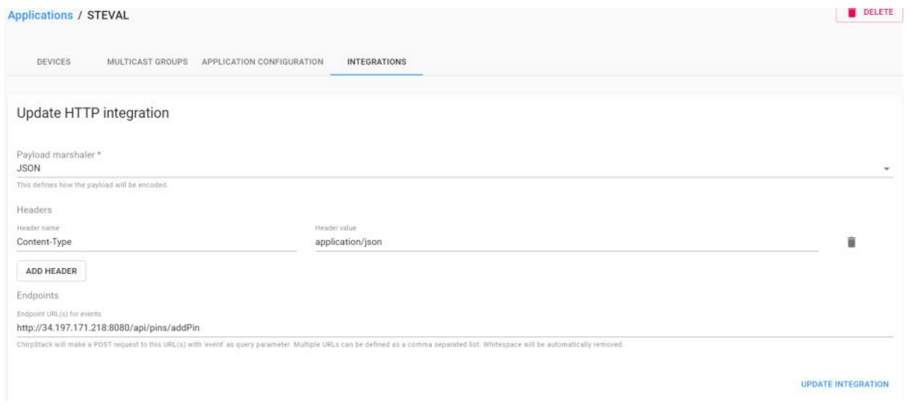
Luckily because of how LoRaWAN works there is a solution to this issue. A LoRaWAN gateway will indiscriminately pick up all transmitted LoRa frames within the gateways range and send it to the ChirpStack server where the Application server will decide if the node is a part of the network and decode the information if so. Because of this it is a very simple process to discover a single nodes Device EUI by going into the LoRaWAN frames tab in the Gateway section and observing the DevEUI of the device sending out Join requests.



Apr 25 9:52:28 AM JoinRequest 868.3 MHz SF12 BW125 DevEUI: 0080e115000a92f3

Figure 7-14 ChirpStack Frame Join Request

Finally in order to send data from the ChirpStack server to the backend HTTP integrations can be used to send a RESTFul API Post request using a route that is defined in the AWS EC2 hosted backend.



Update HTTP integration

Payload marshaler *
JSON
This defines how the payload will be encoded.

Headers

Header name: Content-Type
Header value: application/json

Endpoints

Endpoint URL(s) for events: http://34.197.171.218:8080/api/pins/addPin
ChirpStack will make a POST request to this URL(s) with 'event' as query parameter. Multiple URLs can be defined as a comma separated list. Whitespace will be automatically removed.

Figure 7-15 ChirpStack HTTP Integration

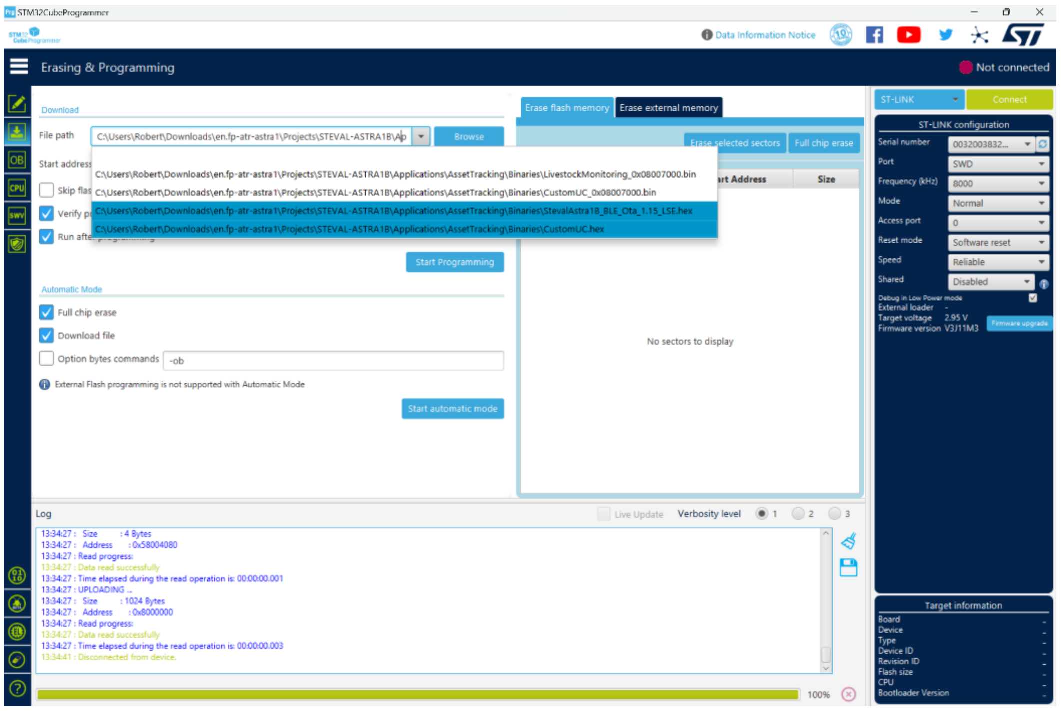
8 Background (STEVAL-ASTRA1B)

The LoRaWAN node selected for this project is the STEVAL-ASTRA1B. It is a development kit and reference design that simplifies prototyping, testing, and evaluating advanced asset tracking applications such as livestock monitoring, fleet management and logistics [6]. Key features essential to the project include a LoRa supported STM32WL55JC SOC, LIS2DTW12 3-axis accelerometer and a Teseo-LIV3F GNSS receiver. Two things of note in the demonstration of this devices functionality within the project is that the GNSS module is relatively weak and as about four satellites are required to pick up coordinates it must be used outdoors in a relatively open area. Secondly regardless of the sending interval the device has for lora frames a frame can be manually sent at any time by pressing the button on the front of the device.

As the device features STM SOC's it can be programmed with software provided by st.com [7]. The primary software used to program this device was STMCubeIDE however STMCubeProgramer was also required in fixing an issue. Also provided be st.com is by FP-ATR-ASTRA1 function pack [8]. This function pack for the STEVAL-ASTRA1B contains example projects that adjust key variables to suit multiple scenarios. These scenarios include Fleet management, Livestock monitoring, Goods monitoring, Logistics and Custom. While the Livestock monitoring sounds like a good fit for this project I opted to start with the Custom project as I wanted the most neutral starting point.

In order to program the device an external board the STLINK-V3MINIE is required as the integrated USB C port on the board cannot be used for this purpose. On the initial attempt to flash the example firmware there was an error which prevented any firmware from being flashed and also removed the original firmware effectively leaving the device unusable. After further research on online forums I came across recommendations on how to fix this issue. First as previously mentioned STMCubeProgramer must be installed. Once the software is installed and the device is connected a full chip erase must be performed. This can be done within the Erasing & Programming section.

Once the chip has been erased, under Download the .hex file located within the en.fp-atr-astra1 folder that was previously downloaded as part of the FP-ATR-ASTRA1 function pack can be selected and installed. This will return the device to a functioning state and allow for further flashing of firmware.


File path: C:\Users\Robert\Downloads\en.fp-atr-astra1\Projects\STEVAL-ASTRA1B\Ap

C:\Users\Robert\Downloads\en.fp-atr-astra1\Projects\STEVAL-ASTRA1B\Applications\AssetTracking\Binaries\LivestockMonitoring_0x08007000.bin
C:\Users\Robert\Downloads\en.fp-atr-astra1\Projects\STEVAL-ASTRA1B\Applications\AssetTracking\Binaries\CustomUC_0x08007000.bin
C:\Users\Robert\Downloads\en.fp-atr-astra1\Projects\STEVAL-ASTRA1B\Applications\AssetTracking\Binaries\StevalAstra1B_BLE_Ota_1.15_LSE.hex
C:\Users\Robert\Downloads\en.fp-atr-astra1\Projects\STEVAL-ASTRA1B\Applications\AssetTracking\Binaries\CustomUC.hex

Automatic Mode
- [✓] Full chip erase
- [✓] Download file
- [ ] Option bytes commands -ob

External Flash programming is not supported with Automatic Mode

No sectors to display

ST-LINK configuration

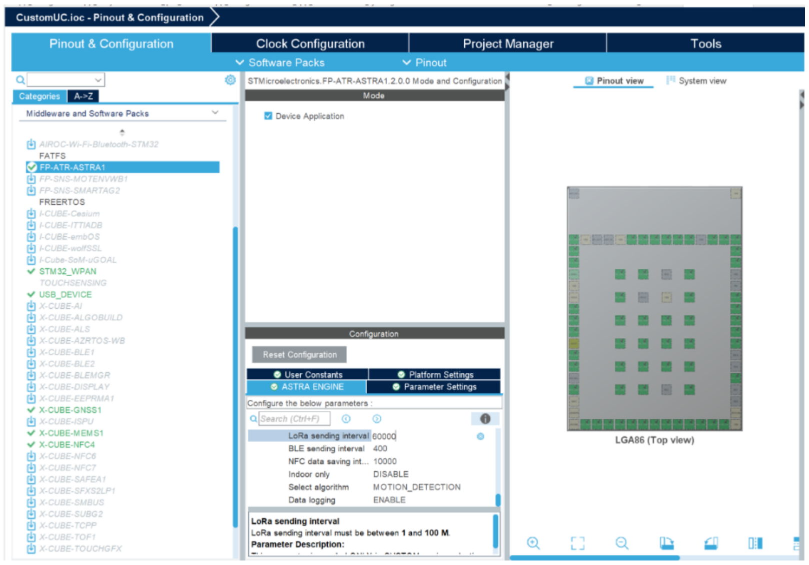
Serial number	0032003832...
Port	SWD
Frequency (kHz)	8000
Mode	Normal
Access port	0
Reset mode	Software reset
Speed	Reliable
Shared	Disabled
Debug in Low Power mode	✓
External loader	-
Target voltage	2.95 V
Firmware version	V3J11M3

Log

```
13:34:27 :   Size         : 4 Bytes
13:34:27 :   Address      : 0x58004080
13:34:27 : Read progress:
13:34:27 : Data read successfully
13:34:27 : Time elapsed during the read operation is: 00:00:00.001
13:34:27 : UPLOADING ...
13:34:27 :   Size         : 1024 Bytes
13:34:27 :   Address      : 0x8000000
13:34:27 : Read progress:
13:34:27 : Data read successfully
13:34:27 : Time elapsed during the read operation is: 00:00:00.003
13:34:41 : Disconnected from device.
```

Figure 8-1 STEVAL-ASTRA1B Fix

Upon thorough examination of the example project code and observation on the changes I made to code on the device I concluded that a few changes to the CustomUC example project would be sufficient to achieve what the project requires of it. Once the CustomUC or any other example project has been opened in the IDE open the .ioc file. In Middleware and Software Packs -> FP-ATR-ASTRA1 -> ASTRA Engine the LoRa sending interval can be configured. In real world examples this interval can be kept fairly large as constant updates on sheep locations is not necessary but for the sake of presenting and demonstrating the project it is set to one minute. Once this change has been made the code can be generated by the IDE



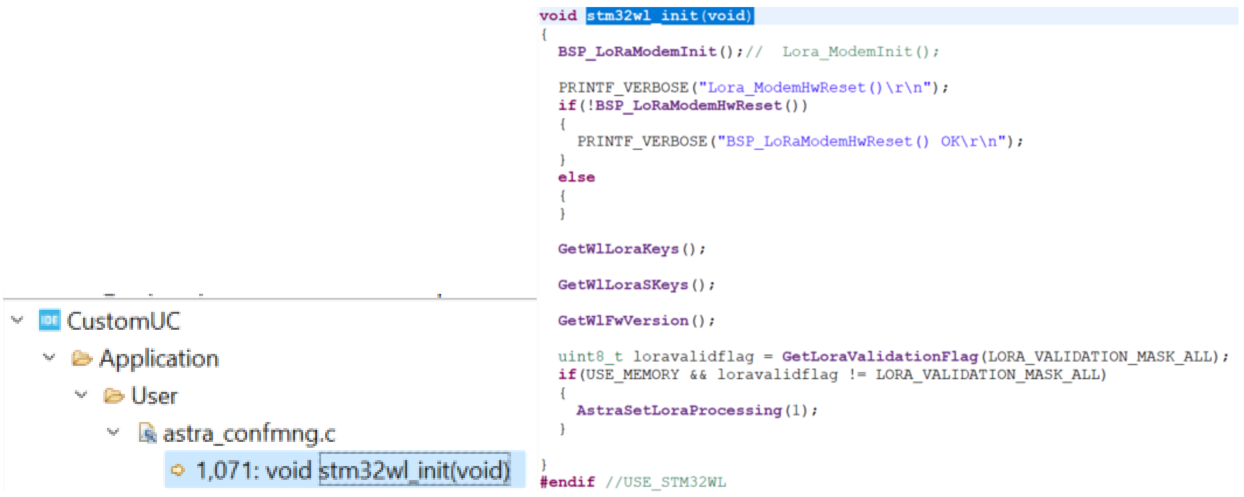
Configure the below parameters :

LoRa sending interval	60000
BLE sending interval	400
NFC data saving int...	10000
Indoor only	DISABLE
Select algorithm	MOTION_DETECTION
Data logging	ENABLE

LoRa sending interval
LoRa sending interval must be between **1 and 100 M**.
Parameter Description:

Figure 8-2 Setting LoRa sending interval

One issue with the auto generated code is the lora functionality is disabled by default. To get around this, AstraSetLoraProccessing called at the end of the stm32wl_init function can be edited so that it passes a 1 instead of its default of 0.



```
CustomUC
  Application
    User
      astra_confmng.c
        1,071: void stm32wl_init(void)
```

```
void stm32wl_init(void)
{
  BSP_LoRaModemInit();//  Lora_ModemInit();

  PRINTF_VERBOSE("Lora_ModemHwReset()\r\n");
  if(!BSP_LoRaModemHwReset())
  {
    PRINTF_VERBOSE("BSP_LoRaModemHwReset() OK\r\n");
  }
  else
  {
  }

  GetWlLoraKeys();

  GetWlLoraSKeys();

  GetWlFwVersion();

  uint8_t loravalidflag = GetLoraValidationFlag(LORA_VALIDATION_MASK_ALL);
  if(USE_MEMORY && loravalidflag != LORA_VALIDATION_MASK_ALL)
  {
    AstraSetLoraProcessing(1);
  }

}
#endif //USE_STM32WL
```

Figure 8-3 Initialise LoRa on startup

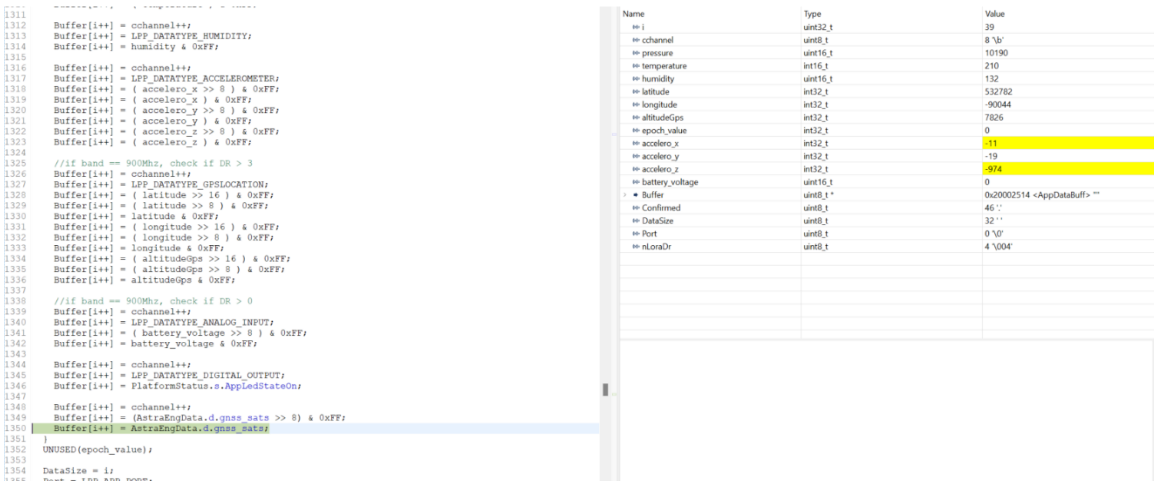
The final edit to the devices code is described in the LoRaWAN decoder section of this report.

9 LoRaWAN Decoder

For any LoRaWAN node a LoRaWAN frame is sent containing all the collected data which it parses into a hexadecimal message to the gateway. When this data reaches the Application server it must be then decoded to reveal the data in a format that is understandable. This part of the report will detail the process taken to decode the hexadecimal message sent by the STEVAL-ASTRA1B.

First, in order to figure out how the hexadecimal message is constructed I had to find where this takes place. Since I used a slightly modified version of the CustomUC solution provided in the FP-ATR-ASTRA1 function pack, I had already familiarised myself with where it happened. Application -> User -> astra_datamng.c contains the LoraSendPacket function. In this function the variables of interest longitude, latitude and accelero_x are assigned a value. Then as seen below these values are inserted into an array called buffer. One of the modifications I made to this code is I added the number of GNSS Satellites to the array. This information is useful when viewing the LoRaWAN frames as if there is too few satellites in view the device will not be able to provide valid data.

Observing this code and running it through the debugger gave me an idea of not only how the data in the hexadecimal message would be structured but also what data to expect when that message is decoded.

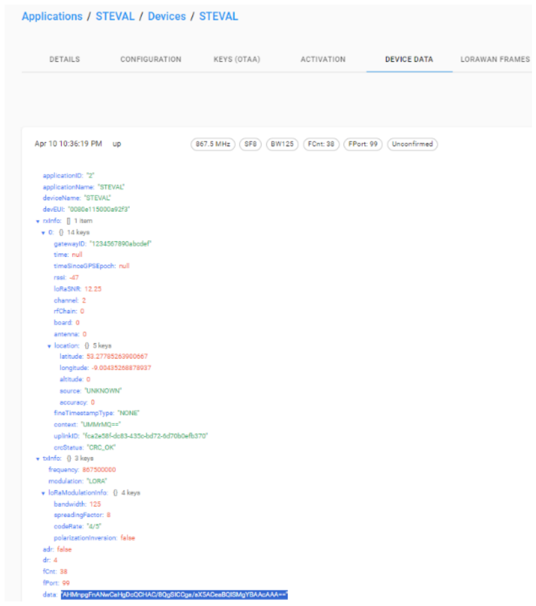


```
    Buffer[i++] = cchannel++;
    Buffer[i++] = LPP_DATATYPE_HUMIDITY;
    Buffer[i++] = humidity & 0xFF;

    Buffer[i++] = cchannel++;
    Buffer[i++] = LPP_DATATYPE_ACCELEROMETER;
    Buffer[i++] = ( accelero_x >> 8 ) & 0xFF;
    Buffer[i++] = ( accelero_x ) & 0xFF;
    Buffer[i++] = ( accelero_y >> 8 ) & 0xFF;
    Buffer[i++] = ( accelero_y ) & 0xFF;
    Buffer[i++] = ( accelero_z >> 8 ) & 0xFF;
    Buffer[i++] = ( accelero_z ) & 0xFF;

    //if band == 900Mhz, check if DR > 3
    Buffer[i++] = cchannel++;
    Buffer[i++] = LPP_DATATYPE_GPSLOCATION;
    Buffer[i++] = ( latitude >> 16 ) & 0xFF;
    Buffer[i++] = ( latitude >> 8 ) & 0xFF;
    Buffer[i++] = latitude & 0xFF;
    Buffer[i++] = ( longitude >> 16 ) & 0xFF;
    Buffer[i++] = ( longitude >> 8 ) & 0xFF;
    Buffer[i++] = longitude & 0xFF;
    Buffer[i++] = ( altitudeGps >> 16 ) & 0xFF;
    Buffer[i++] = ( altitudeGps >> 8 ) & 0xFF;
    Buffer[i++] = altitudeGps & 0xFF;

    //if band == 900Mhz, check if DR > 0
    Buffer[i++] = cchannel++;
    Buffer[i++] = LPP_DATATYPE_ANALOG_INPUT;
    Buffer[i++] = ( battery_voltage >> 8 ) & 0xFF;
    Buffer[i++] = battery_voltage & 0xFF;

    Buffer[i++] = cchannel++;
    Buffer[i++] = LPP_DATATYPE_DIGITAL_OUTPUT;
    Buffer[i++] = PlatformStatus.s.AppLedStateOn;

    Buffer[i++] = cchannel++;
    Buffer[i++] = (AstraEngData.d.gnss_sats >> 8) & 0xFF;
    Buffer[i++] = AstraEngData.d.gnss_sats;
  }
  UNUSED(epoch_value);

  DataSize = i;
```

Name	Type	Value
i	uint32_t	39
cchannel	uint8_t	8 '\b'
pressure	uint16_t	10190
temperature	int16_t	210
humidity	uint16_t	132
latitude	int32_t	532782
longitude	int32_t	-90044
altitudeGps	int32_t	7826
epoch_value	int32_t	0
accelero_x	int32_t	-11
accelero_y	int32_t	-19
accelero_z	int32_t	-974
battery_voltage	uint16_t	0
Buffer	uint8_t *	0x20002514 <AppDataBuff> ""
Confirmed	uint8_t	46 '.'
DataSize	uint8_t	32 ' '
Port	uint8_t	0 '\0'
nLoraDr	uint8_t	4 '\004'

Figure 9-1 LoRa packet structure

Then on the application server in the Device Data section I looked for the base64 string which contains the data.

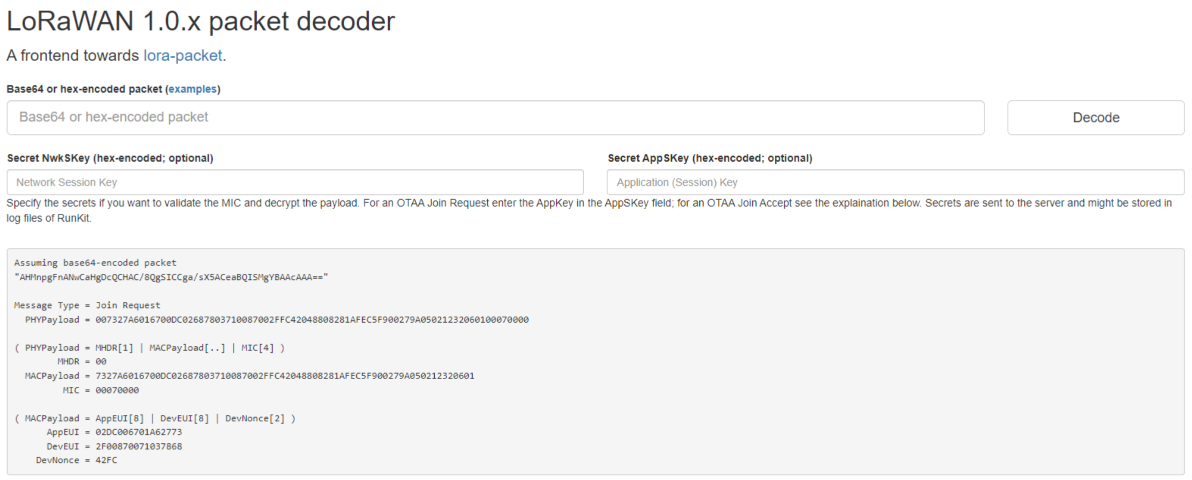


Apr 10 10:36:19 PM up 867.5 MHz SF8 BW125 FCnt: 38 FPort: 99 Unconfirmed

```
applicationID: "2"
applicationName: "STEVAL"
deviceName: "STEVAL"
devEUI: "0080e115000a92f3"
▾ rxInfo: [] 1 item
  ▾ 0: {} 14 keys
      gatewayID: "1234567890abcdef"
      time: null
      timeSinceGPSEpoch: null
      rssi: -47
      loRaSNR: 12.25
      channel: 2
      rfChain: 0
      board: 0
      antenna: 0
    ▾ location: {} 5 keys
        latitude: 53.27785263900667
        longitude: -9.00435268878937
        altitude: 0
        source: "UNKNOWN"
        accuracy: 0
      fineTimestampType: "NONE"
      context: "UMMrMQ=="
      uplinkID: "fca2e58f-dc83-435c-bd72-6d70b0efb370"
      crcStatus: "CRC_OK"
▾ txInfo: {} 3 keys
    frequency: 867500000
    modulation: "LORA"
  ▾ loRaModulationInfo: {} 4 keys
      bandwidth: 125
      spreadingFactor: 8
      codeRate: "4/5"
      polarizationInversion: false
adr: false
dr: 4
fCnt: 38
fPort: 99
data: "AHMnpgFnANwCaHgDcQCHAC/8QgSICCga/sX5ACeaBQISMgYBAAcAAA=="
```

Figure 9-2 Raw data

Following that the string can be pasted into the online LoRaWAN packet decoder [9] to confirm that the data is in fact the same as constructed in the LoraSendPacket function by checking the Hexadecimal values in the PHYPayload.


LoRaWAN 1.0.x packet decoder
A frontend towards lora-packet.
Base64 or hex-encoded packet (examples)
Base64 or hex-encoded packet
Decode
Secret NwkSKey (hex-encoded; optional)
Network Session Key
Secret AppSKey (hex-encoded; optional)
Application (Session) Key

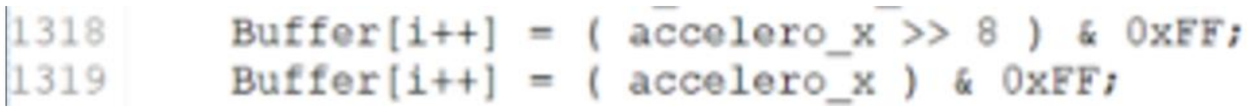
Specify the secrets if you want to validate the MIC and decrypt the payload. For an OTAA Join Request enter the AppKey in the AppSKey field; for an OTAA Join Accept see the explaination below. Secrets are sent to the server and might be stored in log files of RunKit.

```
Assuming base64-encoded packet
"AHMnpgFnANwCaHgDcQCHAC/8QgSICCga/sX5ACeaBQISMgYBAAcAAA=="

Message Type = Join Request
  PHYPayload = 007327A6016700DC02687803710087002FFC42048808281AFEC5F900279A05021232060100070000

( PHYPayload = MHDR[1] | MACPayload[..] | MIC[4] )
        MHDR = 00
  MACPayload = 7327A6016700DC02687803710087002FFC42048808281AFEC5F900279A050212320601
         MIC = 00070000

( MACPayload = AppEUI[8] | DevEUI[8] | DevNonce[2] )
      AppEUI = 02DC006701A62773
      DevEUI = 2F00870071037868
    DevNonce = 42FC
```

Figure 9-3 LoRa packet decoder

Now with an idea of how exactly the payload is structured and confirmation that it contains the expected data, I can now go to the codec section of the ChirpStack device profiles and start to write the decoder.

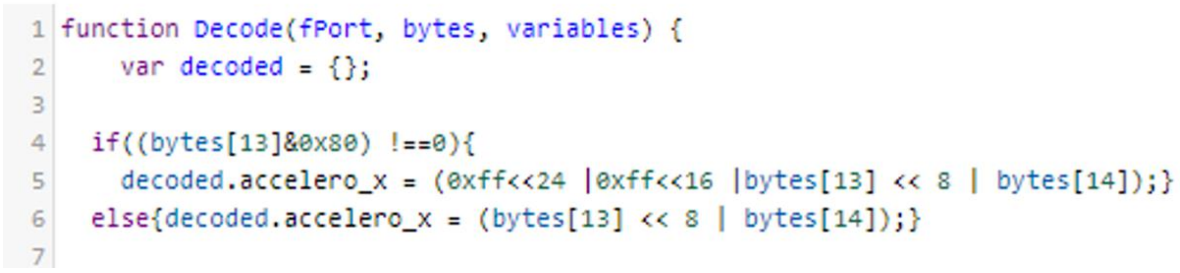
In order to contain the device data within a JSON object an empty object “decoded” is declared. Next with previous knowledge from the LoraSendPacket function of where in the payload the relevant data appears I can start to extract and assign the values to variables that will be displayed in the JSON object. The first variable relevant is accelero_x which starts at the thirteenth hexadecimal value. I also know from referencing the code that accelero_x will get passed as a 16-bit signed value SXXX XXXX XXXX that will use the MSB (Most Significant Bit) “S” to define whether the data is a positive or negative value.



```
        Buffer[i++] = ( accelero_x >> 8 ) & 0xFF;
        Buffer[i++] = ( accelero_x ) & 0xFF;
```

Figure 9-4 16-bit accelerometer data

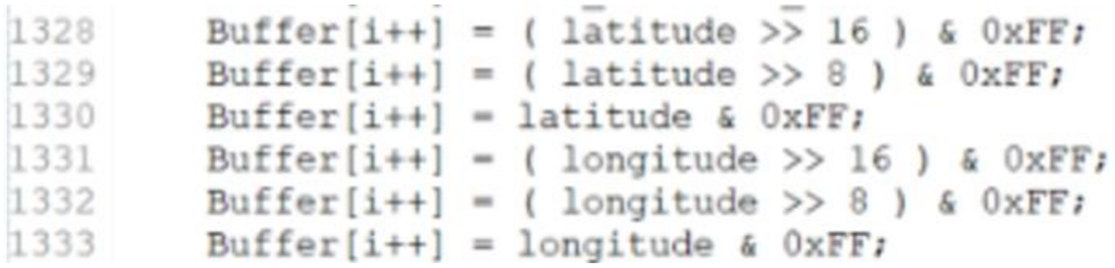
The first step in decoding this is to check if the value is a positive or negative value. This is done using a Bitwise And (&) operator on the MSB as previously mentioned. If the MSB is a 1 it means the value that has been collected from the LoRaWAN node is a negative. Alternatively, if the MSB is a 0 the data is a positive value. If the value is determined by the if statement to be negative it must then be converted to a 32-bit signed number and the left most bits are filled with ones in order to assign it a negative value within the decoded JSON object. This step is skipped if a positive value is determined. Then position 13 and 14 are used to assign the rest of the value by bit shifting position 13 by 8 bits to the left and having position 14 placed at the least significant bits.



```
function Decode(fPort, bytes, variables) {
    var decoded = {};

  if((bytes[13]&0x80) !==0){
    decoded.accelero_x = (0xff<<24 |0xff<<16 |bytes[13] << 8 | bytes[14]);}
  else{decoded.accelero_x = (bytes[13] << 8 | bytes[14]);}

```

Figure 9-5 Converted to 32-bit

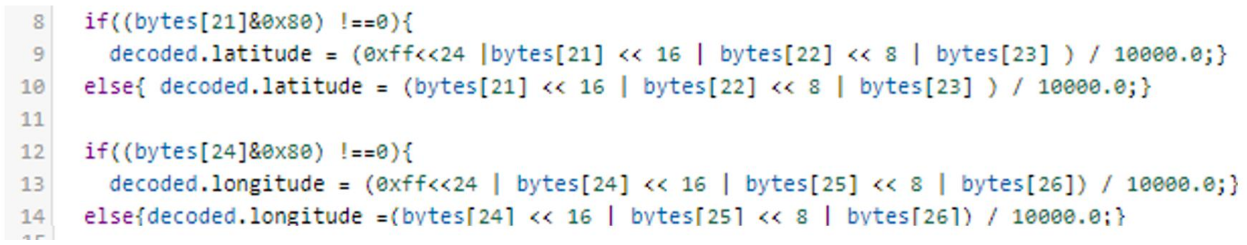
For both the longitude and latitude similar steps are taken, however longitude and latitude are 24-bit signed values and earlier in the code they were multiplied by 1000 as they contain decimal points.



```
Buffer[i++] = ( latitude >> 16 ) & 0xFF;
Buffer[i++] = ( latitude >> 8 ) & 0xFF;
Buffer[i++] = latitude & 0xFF;
Buffer[i++] = ( longitude >> 16 ) & 0xFF;
Buffer[i++] = ( longitude >> 8 ) & 0xFF;
Buffer[i++] = longitude & 0xFF;
```

Figure 9-6 24-bit longitude/latitude data

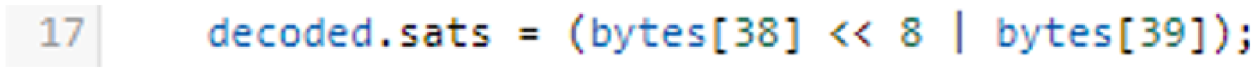
So, in order to decode these values the MSB of position 21 for latitude and 24 longitude are used to find the value of the MSB. Then for latitude position 21 will be bit shifted by 16 bits to the left followed by position 22 being shifted by 8 bits to the left leaving position 23 to occupy the least significant bits. This value then needs to be divided by 1000 in order to revert it back to its original decimal value. The same steps are taken for longitude.



```
if((bytes[21]&0x80) !==0){
  decoded.latitude = (0xff<<24 |bytes[21] << 16 | bytes[22] << 8 | bytes[23] ) / 10000.0;}
else{ decoded.latitude = (bytes[21] << 16 | bytes[22] << 8 | bytes[23] ) / 10000.0;}

if((bytes[24]&0x80) !==0){
  decoded.longitude = (0xff<<24 | bytes[24] << 16 | bytes[25] << 8 | bytes[26]) / 10000.0;}
else{decoded.longitude =(bytes[24] << 16 | bytes[25] << 8 | bytes[26]) / 10000.0;}
```

Figure 9-7 Converted to 32-bit

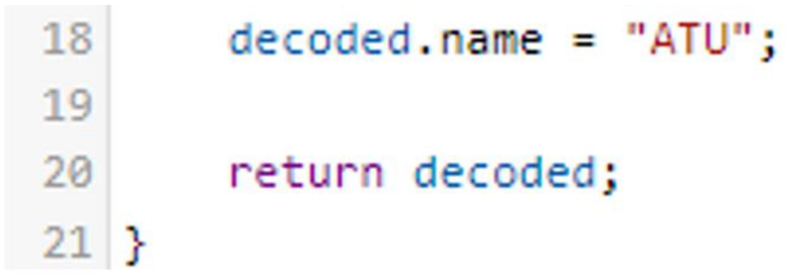
As I also added the number of satellites to the end of the original code it would be particularly useful information to decode. No if statement is needed in this case as there cannot be negative numbers of satellites.



```
decoded.sats = (bytes[38] << 8 | bytes[39]);
```

Figure 9-8 Satellite data

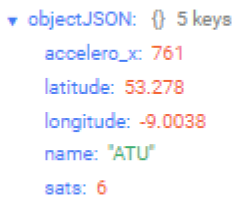
Finally for the sake of determining the identity of the device on the frontend I assigned a name. This would be particularly useful if there were more nodes that needed to be differentiated.



```
    decoded.name = "ATU";

    return decoded;
}
```

Figure 9-9 Device name

The result can then be seen in any proceeding LoRaWAN frames contained within the objectJSON:



```
▾ objectJSON:  {} 5 keys
    accelero_x: 761
    latitude: 53.278
    longitude: -9.0038
    name: "ATU"
    sats: 6
```

Figure 9-10 Decoded LoRaWAN frame

10 Backend

The Backend for this project allows for the creating, deleting, and retrieving of pins featuring the location and accelerometer data. It was created using Spring Boot which is an open-source tool that makes it easier to use Java-based frameworks to create microservices and web apps [10]. It also connects to a MongoDB backend where these pins are stored. The structure of the project's backend can be viewed as a that of a singular microservice that could be potentially expanded on to feature in an even bigger project. Like any microservice it features a main Data class, a Controller class, a Service class, and a Repo class.

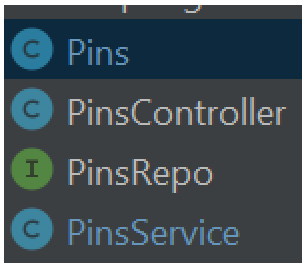
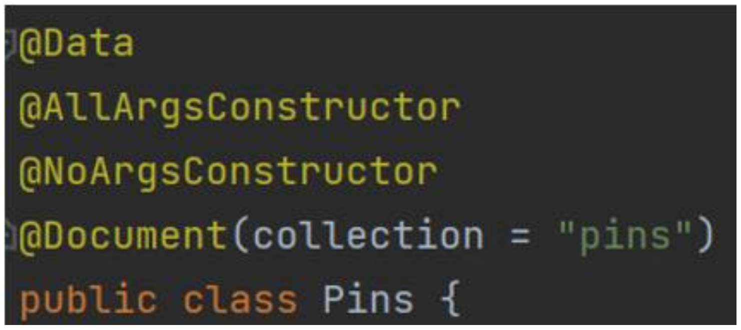


Figure 10-1 Backend Structure

The main data class featured in this backend is Pins. The Pins class is first annotated as a Data class so Spring Boot can identify its purpose and then create all its constructors. It is also given the Document annotation so it can identify in which collection within the database the pins will be stored.



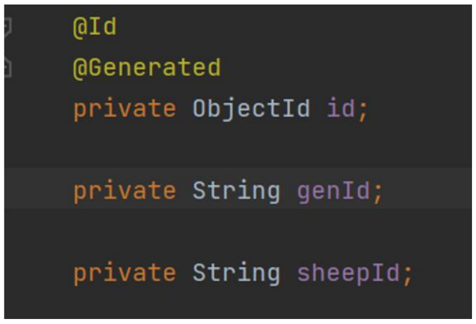
```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Document(collection = "pins")
public class Pins {
```

Figure 10-2 Pins Class

Next within the variables there are three separate ID's given. The first ObjectID as seen by the annotation above it is generated by MongoDB when the listing is created in the database. This

ID however is an object containing both an id and a timestamp instead of a string which makes it more difficult to work with when trying to use it to identify which pin is requested to be deleted. To get around this the addPins function will generate its random ID and assign it to genId.

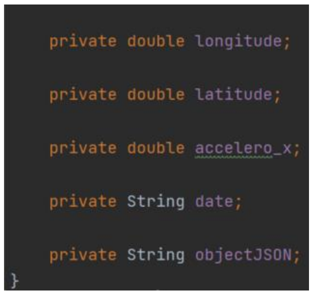
The sheepId is a more readable ID that is assigned the sheep name that has been read by the decoder. In the case of this project, it will always be "ATU" however with more lora nodes featuring different sheep names this ID would be important in quickly identifying the sheep.



```
@Id
@Generated
private ObjectId id;

private String genId;

private String sheepId;
```

Figure 10-3 Pins Class Id's

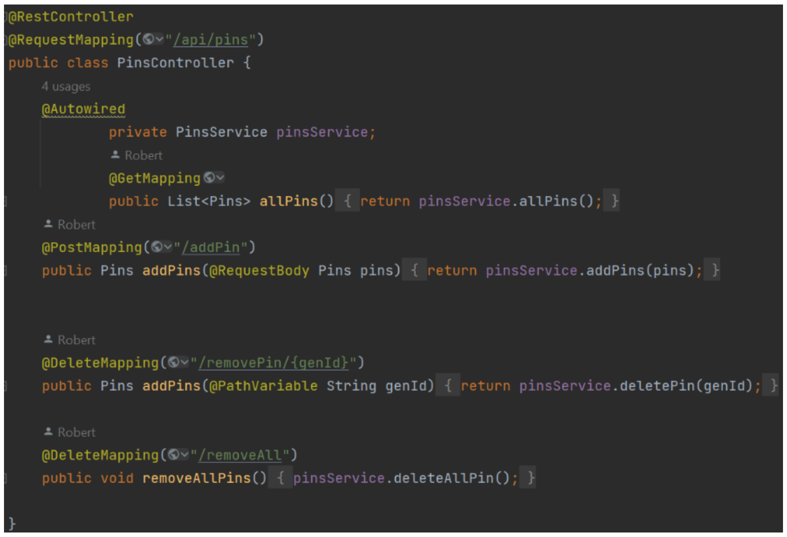
Finally, there is the values that will be displayed on the frontend. The date value is assigned using a formatted version of the timestamp within the MongoDB generated ObjectID. The rest of the values all originate from objectJSON which is the JSON Object that gets sent using a REST API call from the ChirpStack Network servers HTTP integrations every time a LoRaWAN frame is received from the gateway and decoded. The longitude, latitude and accelero_x are all parsed out of the object and assigned to their corresponding variables.



```
    private double longitude;

    private double latitude;

    private double accelero_x;

    private String date;

    private String objectJSON;
}
```

Figure 10-4 Pins Class data variables

The controller class features the mapping for the REST API calls used to interact with the backends database. All these mappings will call a function that is defined in the service class. The get map is called on load up and set intervals by the frontend. It retrieves a list of all the pins from the database. The post map will add additional pins to the database, this is the mapping used by ChirpStack's HTTP integration and will be called every time the gateway receives a new LoRaWAN frame from the node. There are also two delete mappings. One to delete a specific pin based on its ID, this will be called when a user selects a pin on the frontend they want to delete. The second delete mapping to remove all pins is not called from the frontend or ChirpStack. It is only used to clear the map for demonstration purposes and therefore must be accessed with a tool like Postman or Talend API.

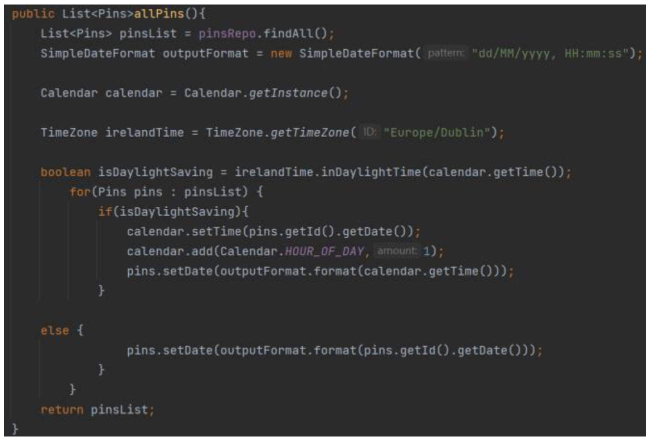


```
@RestController
@RequestMapping("/api/pins")
public class PinsController {
    4 usages
    @Autowired
            private PinsService pinsService;
            Robert
            @GetMapping
            public List<Pins> allPins() { return pinsService.allPins(); }
    Robert
    @PostMapping("/addPin")
    public Pins addPins(@RequestBody Pins pins) { return pinsService.addPins(pins); }


    Robert
    @DeleteMapping("/removePin/{genId}")
    public Pins addPins(@PathVariable String genId) { return pinsService.deletePin(genId); }

    Robert
    @DeleteMapping("/removeAll")
    public void removeAllPins() { pinsService.deleteAllPin(); }

}
```

Figure 10-5 Pins Controller Class

Once one of the REST API mappings are called, they will call their corresponding function within the service class where their interaction with the repository and any additional process to achieve the desired result is defined. The allPins function will first retrieve a list of all the pins stored in the database. When they are displayed on the frontend, they must feature a timestamp to signify to the user when the data was sent to the network. They be automatically given a timestamp within the generated objectId when they are entered into the MongoDB database. However, this time is format in a non-user-friendly manner e.g.2023-12-

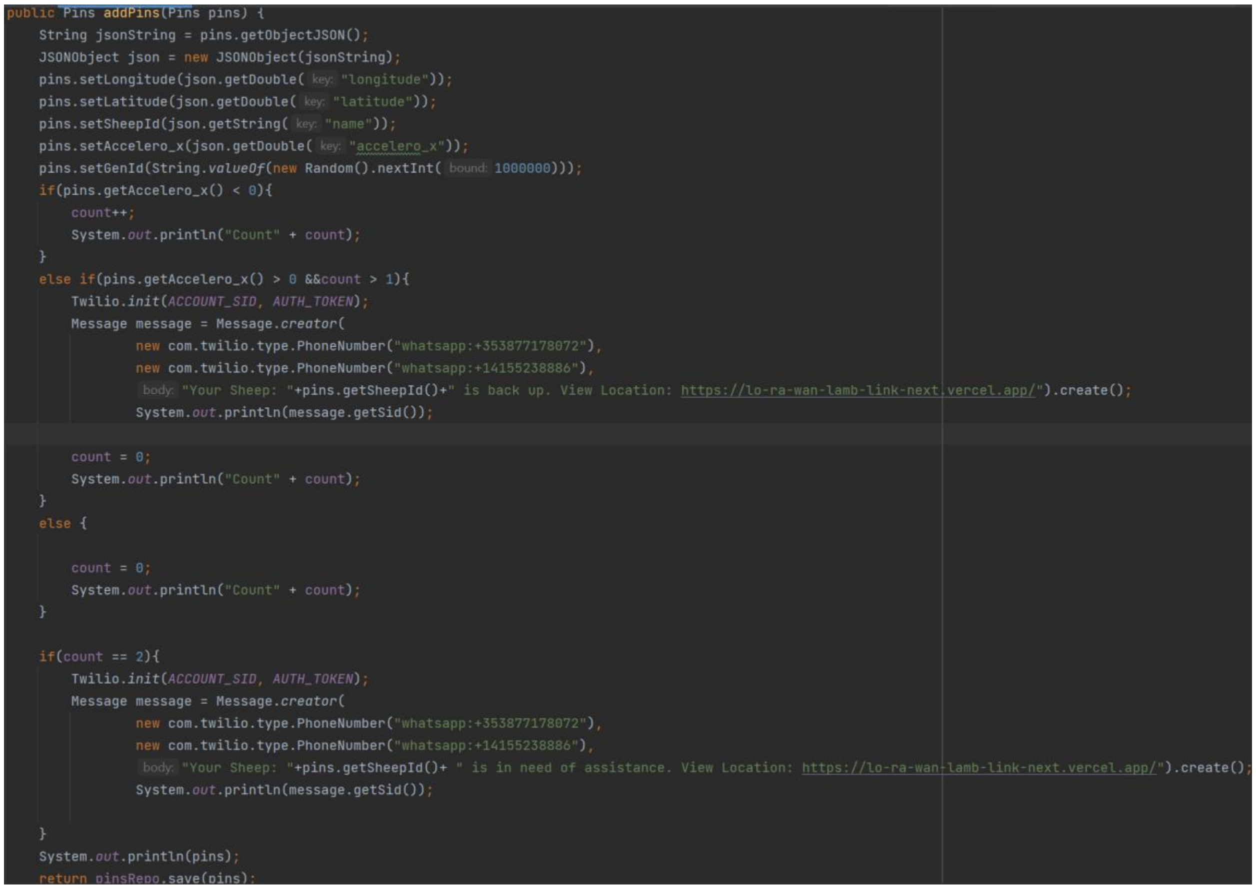
03T12:49:39.428+00:00. Before the pins are sent to the frontend this date is formatted to a much more readable dd/mm/yyyy, hh:mm: ss format. The automatically generated timestamp is also in UTC so to make sure the correct GMT time is displayed. TimeZone is used to check whether Ireland is currently within Daylight time and adjust the time accordingly.



```java
public List<Pins>allPins(){
    List<Pins> pinsList = pinsRepo.findAll();
    SimpleDateFormat outputFormat = new SimpleDateFormat( pattern: "dd/MM/yyyy, HH:mm:ss");

    Calendar calendar = Calendar.getInstance();

    TimeZone irelandTime = TimeZone.getTimeZone( ID: "Europe/Dublin");

    boolean isDaylightSaving = irelandTime.inDaylightTime(calendar.getTime());
        for(Pins pins : pinsList) {
            if(isDaylightSaving){
                calendar.setTime(pins.getId().getDate());
                calendar.add(Calendar.HOUR_OF_DAY, amount: 1);
                pins.setDate(outputFormat.format(calendar.getTime()));
            }

    else {
                pins.setDate(outputFormat.format(pins.getId().getDate()));
            }
        }
    return pinsList;
}
```

Figure 10-6 allPins Function

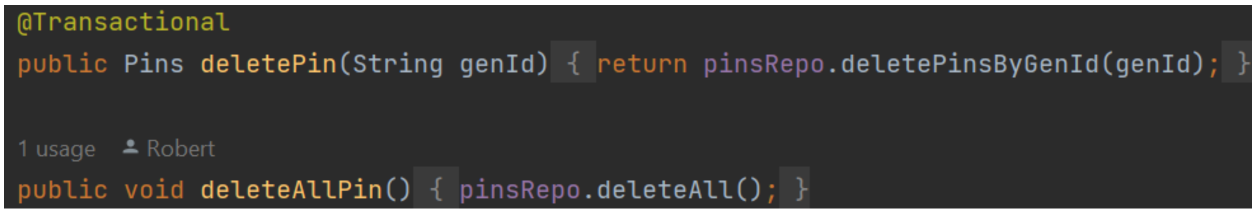
The addPins function is responsible for assigning all the collected data to the variables of the pin object before they are stored in the database through the repository. When this function is called by ChirpStack HTTP integration using the corresponding REST mapping it is sent a pin object that has one defined variable the objectJSON. This objectJSON contains all the relevant data need to assign the rest of the variable with their data. Next it will assign the genId by generating a random number. After the accelero_x value will be checked to determine if the sheep needs assistance. If the value is a negative the sheep has been determined to be upside down and a count variable will be incremented. If two concurrent messages have negative values the sheep is determined to need assistance and a WhatsApp message is constructed and sent using Twilio. This message features a message alerting the farmer of the sheep's status, sheepId and a link to view the pins on the frontend. A similar message will be sent when the accelero_x next gets a positive value after the alert has been sent out. This message has a modified message to inform the farmer that the sheep is now okay.



```java
public Pins addPins(Pins pins) {
    String jsonString = pins.getObjectJSON();
    JSONObject json = new JSONObject(jsonString);
    pins.setLongitude(json.getDouble( key: "longitude"));
    pins.setLatitude(json.getDouble( key: "latitude"));
    pins.setSheepId(json.getString( key: "name"));
    pins.setAccelero_x(json.getDouble( key: "accelero_x"));
    pins.setGenId(String.valueOf(new Random().nextInt( bound: 1000000)));
    if(pins.getAccelero_x() < 0){
        count++;
        System.out.println("Count" + count);
    }
    else if(pins.getAccelero_x() > 0 &&count > 1){
        Twilio.init(ACCOUNT_SID, AUTH_TOKEN);
        Message message = Message.creator(
                new com.twilio.type.PhoneNumber("whatsapp:+353877178072"),
                new com.twilio.type.PhoneNumber("whatsapp:+14155238886"),
                 body: "Your Sheep: "+pins.getSheepId()+" is back up. View Location: https://lo-ra-wan-lamb-link-next.vercel.app/").create();
                System.out.println(message.getSid());

        count = 0;
        System.out.println("Count" + count);
    }
    else {

        count = 0;
        System.out.println("Count" + count);
    }

    if(count == 2){
        Twilio.init(ACCOUNT_SID, AUTH_TOKEN);
        Message message = Message.creator(
                new com.twilio.type.PhoneNumber("whatsapp:+353877178072"),
                new com.twilio.type.PhoneNumber("whatsapp:+14155238886"),
                 body: "Your Sheep: "+pins.getSheepId()+ " is in need of assistance. View Location: https://lo-ra-wan-lamb-link-next.vercel.app/").create();
                System.out.println(message.getSid());

    }
    System.out.println(pins);
    return pinsRepo.save(pins);
```

Figure 10-7 addPins Function

The deletePin function will be called from the frontend and will use the repository to delete a certain pin from the MongoDB database. The deleteAllPin will use the repository to remove all pins from the database. This can only be called using an external tool like Postman or Talend API



```java
@Transactional
public Pins deletePin(String genId) { return pinsRepo.deletePinsByGenId(genId); }

1 usage   Robert
public void deleteAllPin() { pinsRepo.deleteAll(); }
```

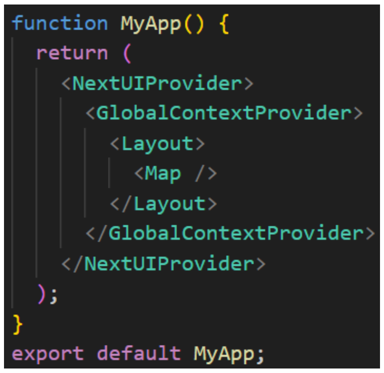
Figure 10-8 Delete Functions

11 Frontend

11.1 Frontend Structure

The frontend for this project provides a user-friendly interface that allows for the creation, retrieval, and deletion of pins. It is built using Next.js a React framework that provides building blocks such as routing to build fully interactive dynamic web applications[11] .

The entry point of the frontend is the MyApp component, this encapsulates the entire application. Next the project is wrapped with a NextUIProvider. NextUI is a UI library that combines the power of TailwindCSS with React Aria to provide complete components (logic and styles) for building accessible and customisable interfaces. The project is then wrapped with a GlobalContextProvider to manage global state and context across the application. Inside that is the Layout component which is mostly responsible for the visual structure of the app the application. Finally, within the layout there is the map component which is responsible for providing the application with a map that a user can use to visualise and interact with pins representing the location of sheep.

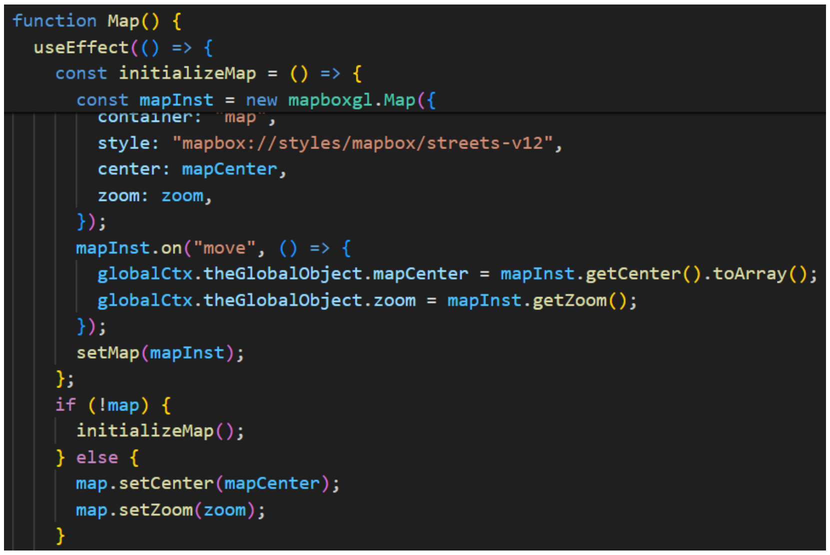


```
function MyApp() {
  return (
    <NextUIProvider>
      <GlobalContextProvider>
        <Layout>
          <Map />
        </Layout>
      </GlobalContextProvider>
    </NextUIProvider>
  );
}
export default MyApp;
```

Figure 11-1-1 Frontend Structure

11.2 Map Component

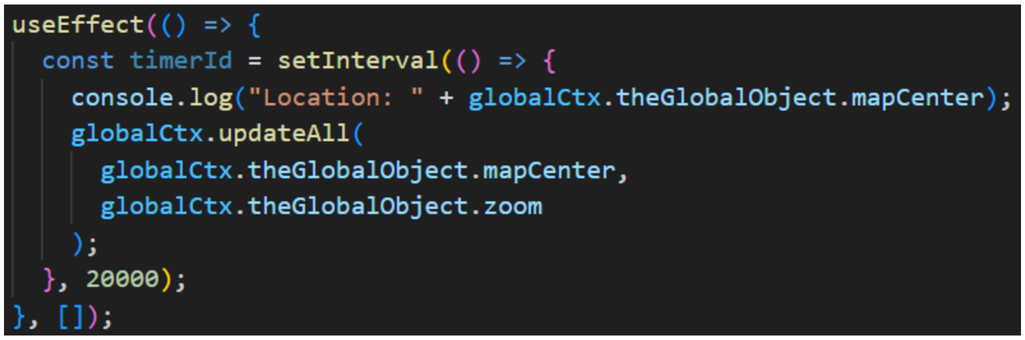
Central to the whole application, the map component utilises Mapbox GL, a client-side JavaScript library for building web maps and web applications [12]. In this project mapbox is used to provide an interactive map that a user can view as well as delete the past and present location and status of sheep as represented by pin markers. First an instance of mapboxgl is initialised with a style, position of the map will be centred at by default and level of zoom of the map. Next the position is kept track of on interaction with the map. As this map is periodically updated with new pin data it is important to keep track of where the user's current position is to prevent the map jumping back to the default position. The map will only be initialised if there is currently no map otherwise the centre and zoom levels will be updated.



```
function Map() {
  useEffect(() => {
    const initializeMap = () => {
      const mapInst = new mapboxgl.Map({
        container: "map",
        style: "mapbox://styles/mapbox/streets-v12",
        center: mapCenter,
        zoom: zoom,
      });
      mapInst.on("move", () => {
        globalCtx.theGlobalObject.mapCenter = mapInst.getCenter().toArray();
        globalCtx.theGlobalObject.zoom = mapInst.getZoom();
      });
      setMap(mapInst);
    };
    if (!map) {
      initializeMap();
    } else {
      map.setCenter(mapCenter);
      map.setZoom(zoom);
    }
```

Figure 11-2-1 Map initialisation

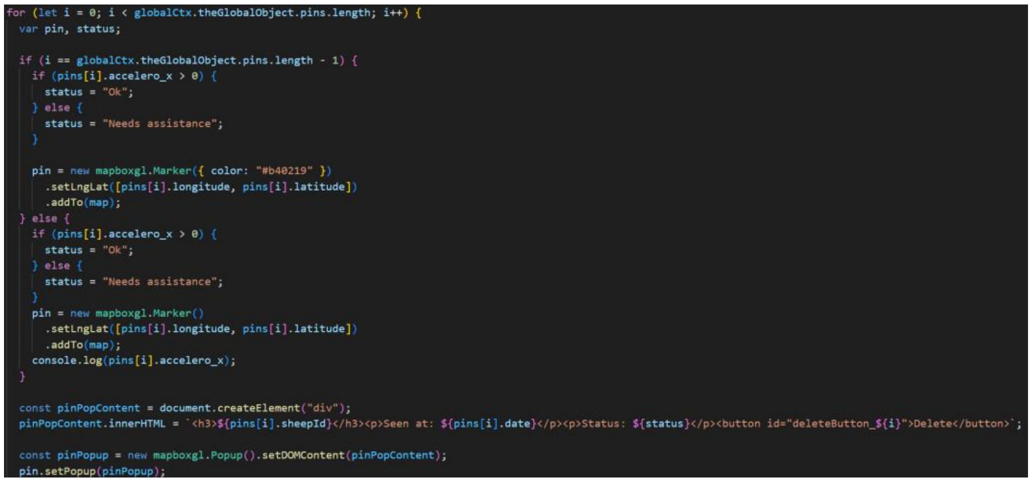
Next a timer is used to keep the map pins updated. The current centre and zoom levels will be updated to prevent the map from jumping back to the default position. This is important in providing a good user experience.



```
useEffect(() => {
  const timerId = setInterval(() => {
    console.log("Location: " + globalCtx.theGlobalObject.mapCenter);
    globalCtx.updateAll(
      globalCtx.theGlobalObject.mapCenter,
      globalCtx.theGlobalObject.zoom
    );
  }, 20000);
}, []);
```

Figure 11-2-2 Auto update pins

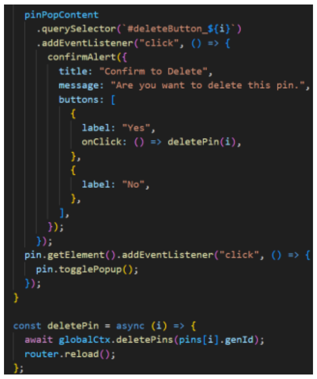
After that to create the pin markers that will display the location data, within a for loop that will run for as many entries as are in the database. The pins are first checked for their accelero_x value and will be designated with a status of “Ok” or “Needs assistance”. If this pin is the last entry within the globalcontext it is the latest pin and is given a separate colour to differentiate it from the other pins. Its longitude and latitude are then assigned, and it is added to the map. After that, the pin is given an interactive popup that will display when the pin is clicked containing the sheep name, date the pin was added, the status and a delete button which features a second popup for confirmation of deletion.



```
for (let i = 0; i < globalCtx.theGlobalObject.pins.length; i++) {
  var pin, status;

  if (i == globalCtx.theGlobalObject.pins.length - 1) {
    if (pins[i].accelero_x > 0) {
      status = "Ok";
    } else {
      status = "Needs assistance";
    }

    pin = new mapboxgl.Marker({ color: "#b40219" })
      .setLngLat([pins[i].longitude, pins[i].latitude])
      .addTo(map);
  } else {
    if (pins[i].accelero_x > 0) {
      status = "Ok";
    } else {
      status = "Needs assistance";
    }
    pin = new mapboxgl.Marker()
      .setLngLat([pins[i].longitude, pins[i].latitude])
      .addTo(map);
    console.log(pins[i].accelero_x);
  }

  const pinPopContent = document.createElement("div");
  pinPopContent.innerHTML = `<h3>${pins[i].sheepId}</h3><p>Seen at: ${pins[i].date}</p><p>Status: ${status}</p><button id="deleteButton_${i}">Delete</button>`;

  const pinPopup = new mapboxgl.Popup().setDOMContent(pinPopContent);
  pin.setPopup(pinPopup);
```

Figure 11-2-3 Create pins on map

Below is the delete confirmation popup. If deletion is confirmed the pins position within the array it was read in is sent to a deletePins function that will use a route called for the globalcontext to call interact with the backend using a REST API call to remove the pin from the database.



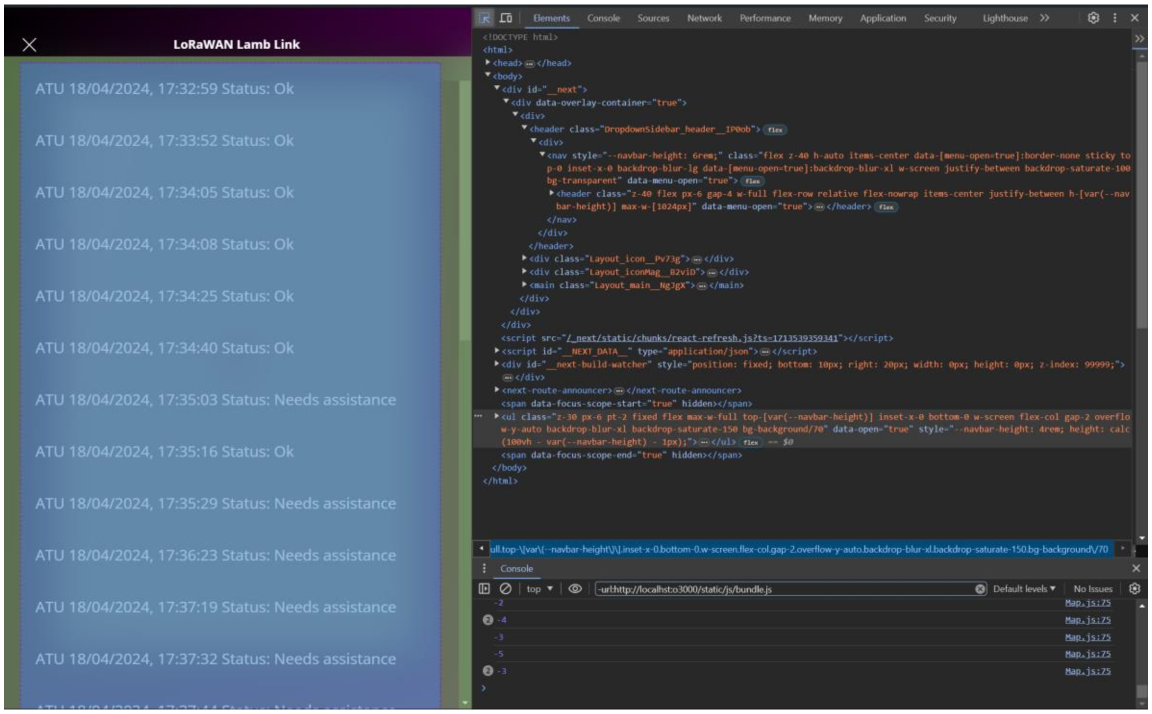
```
    pinPopContent
      .querySelector(`#deleteButton_${i}`)
      .addEventListener("click", () => {
        confirmAlert({
          title: "Confirm to Delete",
          message: "Are you want to delete this pin.",
          buttons: [
            {
              label: "Yes",
              onClick: () => deletePin(i),
            },
            {
              label: "No",
            },
          ],
        });
      });
    pin.getElement().addEventListener("click", () => {
      pin.togglePopup();
    });
  }

  const deletePin = async (i) => {
    await globalCtx.deletePins(pins[i].genId);
    router.reload();
  };
```

Figure 11-1-4 Delete pins

11.3 Dropdown Sidebar (NextUI)

The Dropdown Sidebar component complements the map component by providing a sidebar containing a list of the pins on the map allowing the user to not only quickly view all the pins data such as sheepId, date and status all listed in chronological order, but also provides the ability to quickly jump to that position on the map component.

To create this component the Navbar component from NextUI is modified [13] . While the Navbar components provide visually pleasing toggle buttons and a menu that can be used to display content in its default state it will not completely fit the aesthetic this project requires. for instance the menu component will by default take up the entire screen, this is not ideal as it would be preferred that the map pins are visible while viewing the pins. Viewing the NextUI docs for this component these are not attributes that are specifically documented to allow editing. To get around this issue browser developer tools can be used to select the component. With the component selected the elements of the component can be edited.



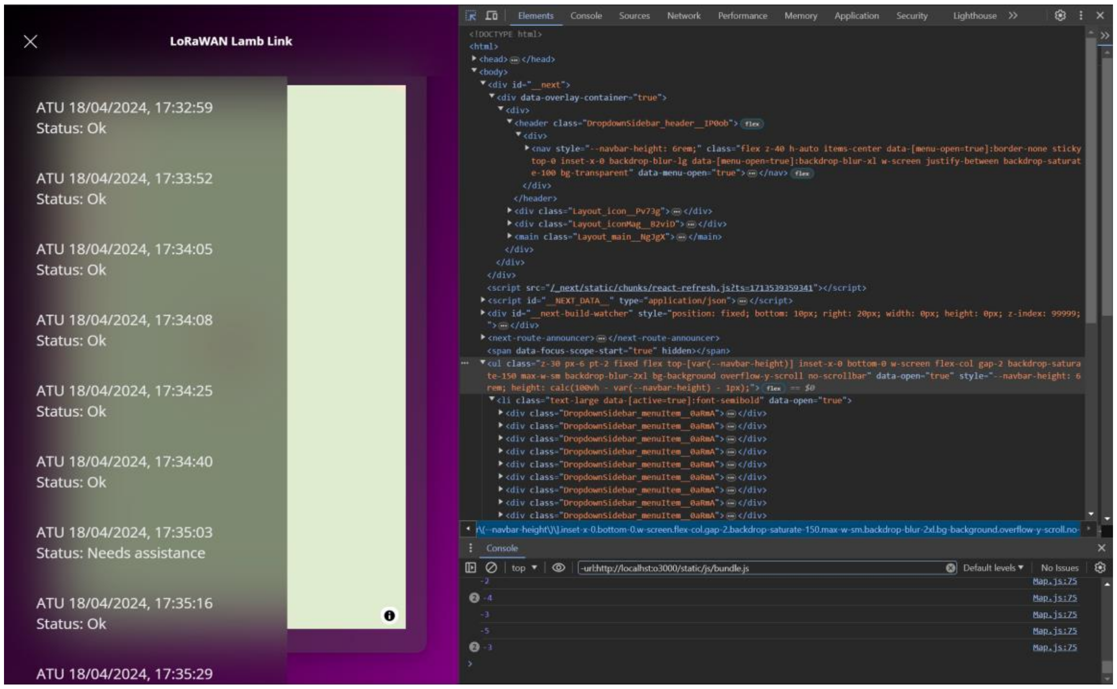
```
ATU 18/04/2024, 17:32:59 Status: Ok
ATU 18/04/2024, 17:33:52 Status: Ok
ATU 18/04/2024, 17:34:05 Status: Ok
ATU 18/04/2024, 17:34:08 Status: Ok
ATU 18/04/2024, 17:34:25 Status: Ok
ATU 18/04/2024, 17:34:40 Status: Ok
ATU 18/04/2024, 17:35:03 Status: Needs assistance
ATU 18/04/2024, 17:35:16 Status: Ok
ATU 18/04/2024, 17:35:29 Status: Needs assistance
ATU 18/04/2024, 17:36:23 Status: Needs assistance
ATU 18/04/2024, 17:37:19 Status: Needs assistance
ATU 18/04/2024, 17:37:32 Status: Needs assistance
```

```
<!DOCTYPE html>
<html>
 ▸<head> ⋯ </head>
 ▾<body>
   ▾<div id="__next">
     ▾<div data-overlay-container="true">
       ▾<div>
         ▾<header class="DropdownSidebar_header__IP0ob"> flex
           ▾<div>
             ▾<nav style="--navbar-height: 6rem;" class="flex z-40 h-auto items-center data-[menu-open=true]:border-none sticky to
              p-0 inset-x-0 backdrop-blur-lg data-[menu-open=true]:backdrop-blur-xl w-screen justify-between backdrop-saturate-100
              bg-transparent" data-menu-open="true"> flex
               ▸<header class="z-40 flex px-6 gap-4 w-full flex-row relative flex-nowrap items-center justify-between h-[var(--nav
                bar-height)] max-w-[1024px]" data-menu-open="true"> ⋯ </header> flex
              </nav>
            </div>
          </header>
         ▸<div class="Layout_icon__Pv73g"> ⋯ </div>
         ▸<div class="Layout_iconMag__B2viD"> ⋯ </div>
         ▸<main class="Layout_main__NgJgX"> ⋯ </main>
        </div>
      </div>
    </div>
    <script src="/_next/static/chunks/react-refresh.js?ts=1713539359341"></script>
   ▸<script id="__NEXT_DATA__" type="application/json"> ⋯ </script>
   ▸<div id="__next-build-watcher" style="position: fixed; bottom: 10px; right: 20px; width: 0px; height: 0px; z-index: 99999;">
    ⋯ </div>
   ▸<next-route-announcer> ⋯ </next-route-announcer>
    <span data-focus-scope-start="true" hidden></span>
⋯  ▸<ul class="z-30 px-6 pt-2 fixed flex max-w-full top-[var(--navbar-height)] inset-x-0 bottom-0 w-screen flex-col gap-2 overflo
    w-y-auto backdrop-blur-xl backdrop-saturate-150 bg-background/70" data-open="true" style="--navbar-height: 4rem; height: calc
    (100vh - var(--navbar-height) - 1px);"> ⋯ </ul> flex == $0
    <span data-focus-scope-end="true" hidden></span>
  </body>
</html>

ul.top-\[var\(--navbar-height\)\].inset-x-0.bottom-0.w-screen.flex-col.gap-2.overflow-y-auto.backdrop-blur-xl.backdrop-saturate-150.bg-background\/70
```

```
Console
top ▾   -url:http://localhst:o3000/static/js/bundle.js   Default levels ▾   No Issues
  -2                                                      Map.js:75
② -4                                                      Map.js:75
  -3                                                      Map.js:75
  -5                                                      Map.js:75
② -3                                                      Map.js:75
>
```

Figure 11-3-1 Browser Developer tools

Viewing the effects specific changes have on the component and referencing tailwind resources such as the TailwindCSS official documents[14] and TailwindCSS cheat sheets [15] the desired changes can be made and implemented in the components code.



ATU 18/04/2024, 17:32:59
Status: Ok

ATU 18/04/2024, 17:33:52
Status: Ok

ATU 18/04/2024, 17:34:05
Status: Ok

ATU 18/04/2024, 17:34:08
Status: Ok

ATU 18/04/2024, 17:34:25
Status: Ok

ATU 18/04/2024, 17:34:40
Status: Ok

ATU 18/04/2024, 17:35:03
Status: Needs assistance

ATU 18/04/2024, 17:35:16
Status: Ok

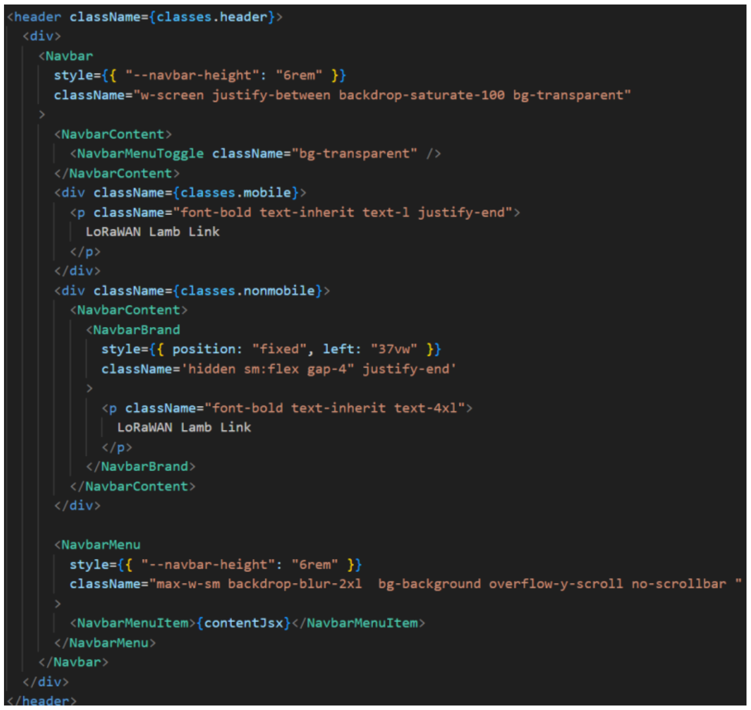
ATU 18/04/2024, 17:35:29

```
<!DOCTYPE html>
<html>
 <head> … </head>
 <body>
  <div id="__next">
   <div data-overlay-container="true">
    <div>
     <header class="DropdownSidebar_header__IP0ob"> flex
      <div>
       <nav style="--navbar-height: 6rem;" class="flex z-40 h-auto items-center data-[menu-open=true]:border-none sticky
       top-0 inset-x-0 backdrop-blur-lg data-[menu-open=true]:backdrop-blur-xl w-screen justify-between backdrop-saturat
       e-100 bg-transparent" data-menu-open="true"> … </nav> flex
      </div>
     </header>
     <div class="Layout_icon__Pv73g"> … </div>
     <div class="Layout_iconMag__82viD"> … </div>
     <main class="Layout_main__NgJgX"> … </main>
    </div>
   </div>
  </div>
  <script src="/_next/static/chunks/react-refresh.js?ts=1713539359341"></script>
  <script id="__NEXT_DATA__" type="application/json"> … </script>
  <div id="__next-build-watcher" style="position: fixed; bottom: 10px; right: 20px; width: 0px; height: 0px; z-index: 99999;
  "> … </div>
  <next-route-announcer> … </next-route-announcer>
  <span data-focus-scope-start="true" hidden></span>
  <ul class="z-30 px-6 pt-2 fixed flex top-[var(--navbar-height)] inset-x-0 bottom-0 w-screen flex-col gap-2 backdrop-satura
  te-150 max-w-sm backdrop-blur-2xl bg-background overflow-y-scroll no-scrollbar" data-open="true" style="--navbar-height: 6
  rem; height: calc(100vh - var(--navbar-height) - 1px);"> flex == $0
   <li class="text-large data-[active=true]:font-semibold" data-open="true">
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
    <div class="DropdownSidebar_menuItem__0aRmA"> … </div>
r\(--navbar-height\)\].inset-x-0.bottom-0.w-screen.flex-col.gap-2.backdrop-saturate-150.max-w-sm.backdrop-blur-2xl.bg-background.overflow-y-scroll.no-
```

```
-url:http://localhst:o3000/static/js/bundle.js
  -2                                                  Map.js:75
2 -4                                                  Map.js:75
  -3                                                  Map.js:75
  -5                                                  Map.js:75
2 -3                                                  Map.js:75
>
```

Figure 11-3-2 CSS successfully changed

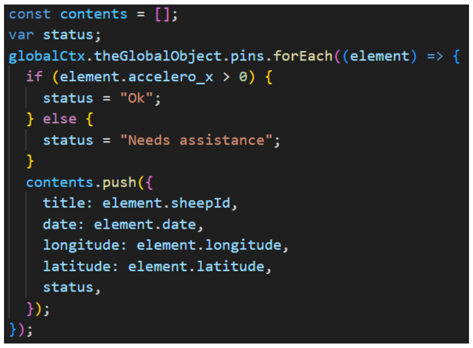
As previously mentioned once the desired changes made to the TailwindCSS components have been made they can be implemented in the code. Unlike making a standard CSS change using style = “{{xxxxxx }}” Tailwind is edited using “className=”xxxxxx”. Everything is contained within the Navbar. First there is the NavbarMenuToggle. This toggle will smoothly transition between a two-bar hamburger icon and an x icon when toggled. As the name suggests I will reveal the menu componant. As seen in the above images this menu has been modified to function as a sidebar. The menus content is defined by the array of values “contentJsx” holds. Two “LoRaWAN Lamb Link” titles can also be seen however only one is ever visible. This is because the website will dynamically adjust between a wider screen like a monitor where having the title within the “NavbarBrand” is more suitable and a narrower screen like a phone where it is not.



```
<header className={classes.header}>
  <div>
    <Navbar
      style={{ "--navbar-height": "6rem" }}
      className="w-screen justify-between backdrop-saturate-100 bg-transparent"
    >
      <NavbarContent>
        <NavbarMenuToggle className="bg-transparent" />
      </NavbarContent>
      <div className={classes.mobile}>
        <p className="font-bold text-inherit text-1 justify-end">
          LoRaWAN Lamb Link
        </p>
      </div>
      <div className={classes.nonmobile}>
        <NavbarContent>
          <NavbarBrand
            style={{ position: "fixed", left: "37vw" }}
            className='hidden sm:flex gap-4" justify-end'
          >
            <p className="font-bold text-inherit text-4xl">
              LoRaWAN Lamb Link
            </p>
          </NavbarBrand>
        </NavbarContent>
      </div>

      <NavbarMenu
        style={{ "--navbar-height": "6rem" }}
        className="max-w-sm backdrop-blur-2xl  bg-background overflow-y-scroll no-scrollbar "
      >
        <NavbarMenuItem>{contentJsx}</NavbarMenuItem>
      </NavbarMenu>
    </Navbar>
  </div>
</header>
```

Figure 11-3-3 Dropdown Sidebar

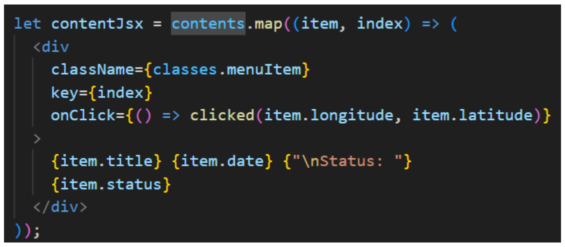
In order set the content to display within the NavbarMenuItem first like how it was done in the map component, pins from the globalcontext have their accelero_x values checked and are assigned the corresponding status to be displayed. Next the variables from those pins including the newly assigned value is pushed into a content array.



```
const contents = [];
var status;
globalCtx.theGlobalObject.pins.forEach((element) => {
  if (element.accelero_x > 0) {
    status = "Ok";
  } else {
    status = "Needs assistance";
  }
  contents.push({
    title: element.sheepId,
    date: element.date,
    longitude: element.longitude,
    latitude: element.latitude,
    status,
  });
});
```

Figure 11-3-4 Menu data read in

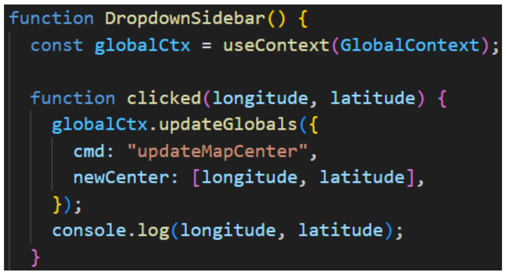
Next the content array is used to map the title, date, and status of the sheep to be displayed into the contextJsx variable. The longitude and latitude are not displayed instead it is sent to the “clicked” function that each of the items are allocated. This function is called when an item is clicked as defined by onClick.



```
let contentJsx = contents.map((item, index) => (
  <div
    className={classes.menuItem}
    key={index}
    onClick={() => clicked(item.longitude, item.latitude)}
  >
    {item.title} {item.date} {"\nStatus: "}
    {item.status}
  </div>
));
```

Figure 11-3-5 Menu items created

The clicked function uses the longitude and latitude of the menu item of the pin that is clicked to centre and zoom the map in on the corresponding pin by calling the updateGlobals function within the globalcontext, passing in the command updateMapCentre as well as the longitude and latitude values.

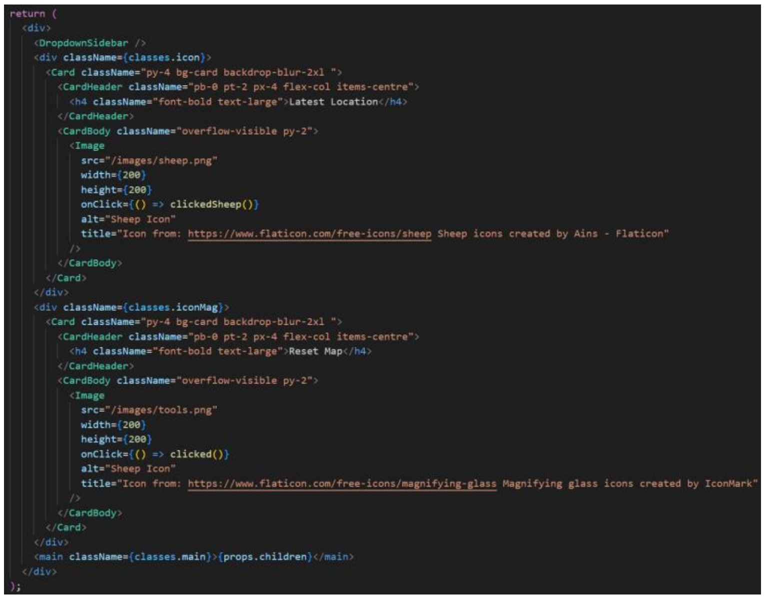


```
function DropdownSidebar() {
  const globalCtx = useContext(GlobalContext);

  function clicked(longitude, latitude) {
    globalCtx.updateGlobals({
      cmd: "updateMapCenter",
      newCenter: [longitude, latitude],
    });
    console.log(longitude, latitude);
  }
```

Figure 11-3-6 Menu items functionality

11.4 Layout

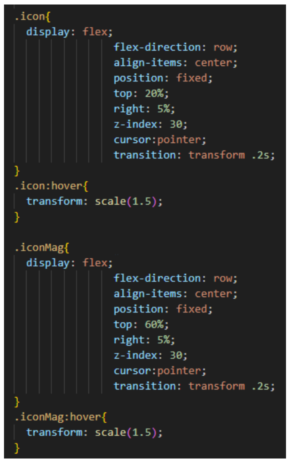
As the name suggest the layout component is the component primarily responsible for how the webapp is laid out. The previously detailed DropdownSidebar component is positioned at the top of the page. Then underneath that and positioned to the right of the page is the interactable icon buttons that will allow the user to either quickly centre the map to either the “Latestest Location” or to “Reset Map” respectively. As is seen in the code or by hovering over these icons they were sourced from flaticon.com [16] Theses icons like the map component are contained within NextUI Card components with blurred backgrounds to give a frosted glass appearance.



```
return (
  <div>
    <DropdownSidebar />
    <div className={classes.icon}>
      <Card className="py-4 bg-card backdrop-blur-2xl ">
        <CardHeader className="pb-0 pt-2 px-4 flex-col items-centre">
          <h4 className="font-bold text-large">Latest Location</h4>
        </CardHeader>
        <CardBody className="overflow-visible py-2">
          <Image
            src="/images/sheep.png"
            width={200}
            height={200}
            onClick={() => clickedSheep()}
            alt="Sheep Icon"
            title="Icon from: https://www.flaticon.com/free-icons/sheep Sheep icons created by Ains - Flaticon"
          />
        </CardBody>
      </Card>
    </div>
    <div className={classes.iconMag}>
      <Card className="py-4 bg-card backdrop-blur-2xl ">
        <CardHeader className="pb-0 pt-2 px-4 flex-col items-centre">
          <h4 className="font-bold text-large">Reset Map</h4>
        </CardHeader>
        <CardBody className="overflow-visible py-2">
          <Image
            src="/images/tools.png"
            width={200}
            height={200}
            onClick={() => clicked()}
            alt="Sheep Icon"
            title="Icon from: https://www.flaticon.com/free-icons/magnifying-glass Magnifying glass icons created by IconMark"
          />
        </CardBody>
      </Card>
    </div>
    <main className={classes.main}>{props.children}</main>
  </div>
);
```

Figure 11-4-1 Layout

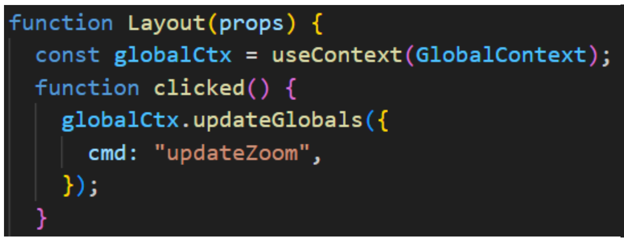
In the CSS file it can not only be seen how these icons are given their position on the right side of the screen but how they respond to being hovered over by smoothly increasing in size. This helps give the page an even more interactive experience.



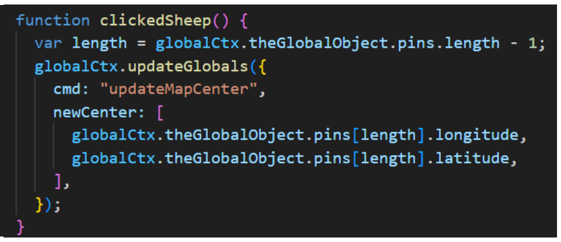
```
.icon{
  display: flex;
                flex-direction: row;
                align-items: center;
                position: fixed;
                top: 20%;
                right: 5%;
                z-index: 30;
                cursor:pointer;
                transition: transform .2s;
}
.icon:hover{
  transform: scale(1.5);
}

.iconMag{
  display: flex;
                flex-direction: row;
                align-items: center;
                position: fixed;
                top: 60%;
                right: 5%;
                z-index: 30;
                cursor:pointer;
                transition: transform .2s;
}
.iconMag:hover{
  transform: scale(1.5);
}
```

Figure 11-4-2 Layout CSS

Going back to the main Layout.js file these icons use a method similar to how the DropdownSidbars menu contents to achieve the desired position and zoom level.



```
function Layout(props) {
  const globalCtx = useContext(GlobalContext);
  function clicked() {
    globalCtx.updateGlobals({
      cmd: "updateZoom",
    });
  }
```



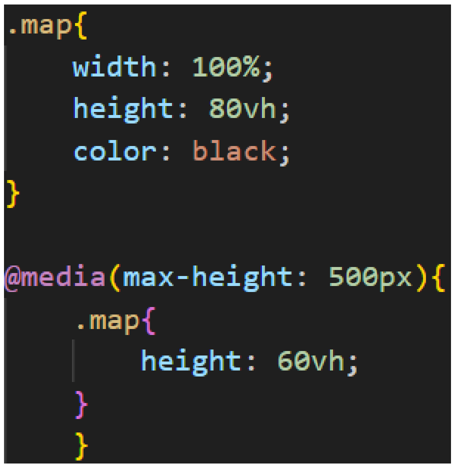
```
function clickedSheep() {
  var length = globalCtx.theGlobalObject.pins.length - 1;
  globalCtx.updateGlobals({
    cmd: "updateMapCenter",
    newCenter: [
      globalCtx.theGlobalObject.pins[length].longitude,
      globalCtx.theGlobalObject.pins[length].latitude,
    ],
  });
}
```

Figure 11-4-3 Layout icon functionality

11.5 Responsive Web Design

"Responsive Web Design is about creating web pages that look good on all devices[17] This is important to the LoRaWAN Lamb Link project as it features not the ability to open the web app from a WhatsApp link that has been sent to a mobile device as well as on a standard laptop or PC.

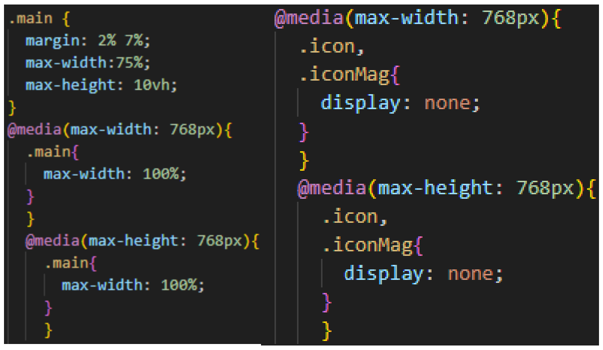
Map- To properly scale the map component for a phone screen in landscape the height of the map is reduced since there is little vertical height to work with.



```
.map{
    width: 100%;
    height: 80vh;
    color: black;
}

@media(max-height: 500px){
    .map{
        height: 60vh;
    }
}
```

Figure 11-5-1 Responsive Map

Layout – The max-width of the Layout component will in turn affect the max-width of the container the map component is in so when a monitor screen is detected it will be at 75% to accommodate the icon buttons to the right-hand side and when a mobile phone screen is detected, either landscape or portrait, the icons will be hidden and the map will be given the full screen. As there is little room to work with it is more important to prioritise the map as it is the main focus of the web application.

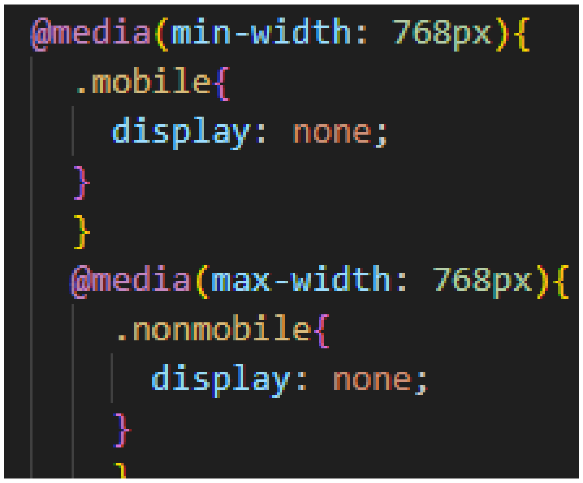


```
.main {
  margin: 2% 7%;
  max-width:75%;
  max-height: 10vh;
}
@media(max-width: 768px){
  .main{
    max-width: 100%;
  }
}
  @media(max-height: 768px){
    .main{
      max-width: 100%;
    }
  }
```

```
@media(max-width: 768px){
  .icon,
  .iconMag{
    display: none;
  }
  }
  @media(max-height: 768px){
    .icon,
    .iconMag{
      display: none;
    }
    }
```

Figure 11-5-2 Responsive Layout

DropdownSidebar- The Dropdown componant features two different types of titles. The NavbarBrand tile suits a larger monitor and simple text within NavbarContent suits a mobile screen so one is always hidden.



```
@media(min-width: 768px){
  .mobile{
    display: none;
  }
  }
  @media(max-width: 768px){
    .nonmobile{
      display: none;
    }
```

Figure 11-5-3 Responsive DropdownSidebar

12 Project Management

Throughout this project in order to plan out and keep track of progress I used Jira. I first broadly planned out what had to be done and gave a general timeframe of when I expected to get the job done using the timeline section. While I was generally able to stick to the timeline I would occasionally have to make adjustments. For instance I originally planned on first working on the STEVAL-ASTRA1B, however it took longer than expected to deliver so instead I worked on the backend and frontend of the project first.

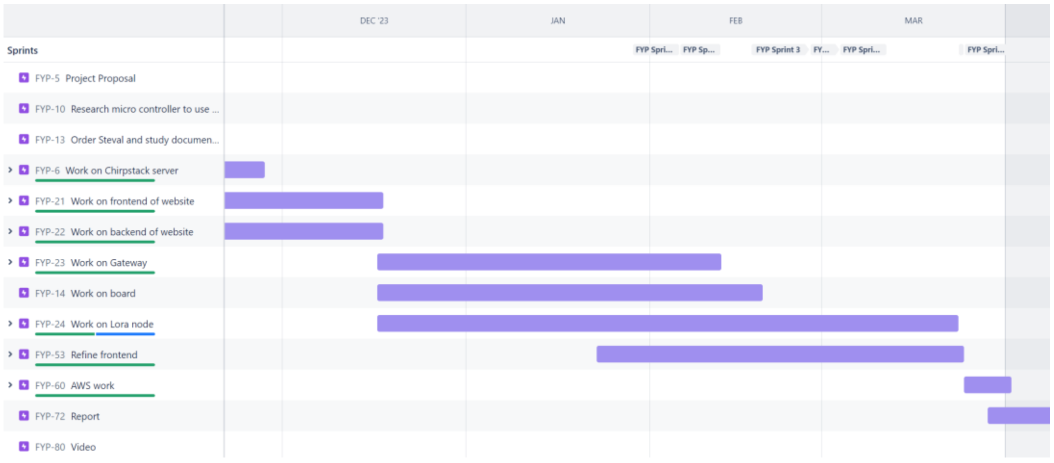


Figure 12-1 Timeline

When it comes to management of my weekly tasks I experimented with both scrum and kanban methods. [18]

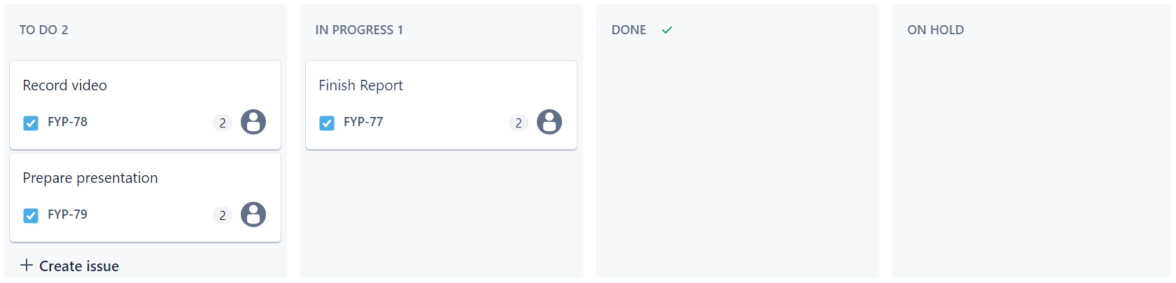
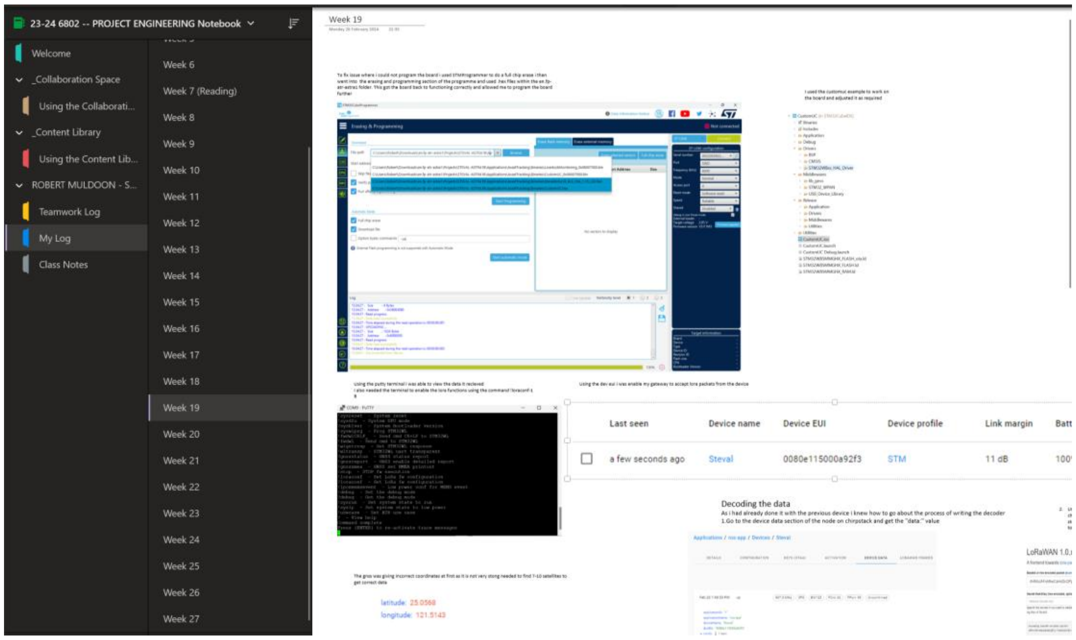


Figure 12-2 Jira

Overall I think scrum is a very useful tool for teams as the sprints deadlines will ensure specific tasks will get done. However in a solo project like this one the more flexible kanban allows more freedom as something I just learned in another module might apply to a part of the project that is currently not being worked on but I can pivot to. The strict deadline of a scrum sprint would not allow this.

I also kept weekly logs of my progress and issues I was facing during that time in my Project Engineering notebook on OneNote.



Week 19

To fix issue where i could not program the board i used STMProgrammer to do a full chip erase i then went into the erasing and programming section of the programme and used .hex files within the en.fp-atr-astrac folder. This got the board back to functioning correctly and allowed me to program the board further

using the putty terminal i was able to view the data it recieved
i also needed the terminal to enable the lora functions using the command !loraconf-1

Using the dev eui i was enable my gateway to accept lora packets from the device

	Last seen	Device name	Device EUI	Device profile	Link margin	Batt
☐	a few seconds ago	Steval	0080e115000a92f3	STM	11 dB	100

Decoding the data
As i had already done it with the previous device i knew how to go about the process of writing the decoder
1.Go to the device data section of the node on chirpstack and get the "data:" value

Applications / rss app / Devices / Steval

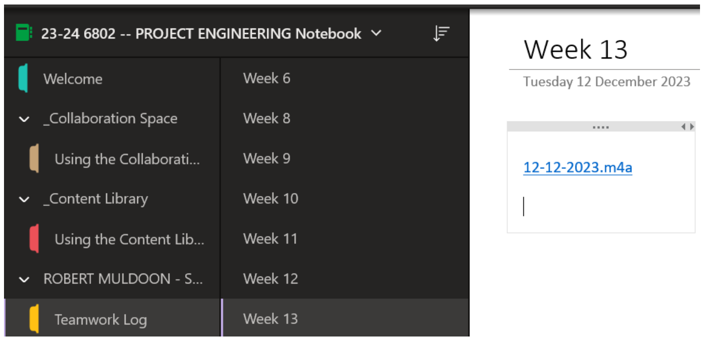
The gnss was giving incorrect coordinates at first as it is not very stong needed to find 7-10 satellites to get correct data

latitude: 25.0568
longitude: 121.5143

LoRaWAN 1.0.

Figure 12-3 Weekly logs

As well as weekly team logs where I had weekly standup meetings to keep myself and others accountable for progressing in our own projects.



Week 13
Tuesday 12 December 2023

12-12-2023.m4a

Figure 12-4 Team logs

13 Ethics

Ethically the capabilities this projects provides can have both positive and negative consequences. As with all tracking technologies there is always a potential for the abuse and misuse of the technology for malicious reasons. These may include the tracking of expensive items or even the tracking of people. The release of the Apple AirTag being a popular example as many people voiced concern over such issues [19].

There is also however the positive effect this project can have ethically which was the original inspiration for this project. By successfully tracking and mapping out areas sheep have grazed, in farmers have a better chance in successfully preventing forest fires as they can easily view what areas may have been neglected by sheep, and would therefore have more vegetation that may potentially catch fire.

The accelerometers ability to detect if a sheep is on its back and sending a WhatsApp alert to the phone of its farmer may will also mean less sheep deaths. This means the project can be a massive benefit to both the environment and animal welfare.

14 Conclusion

In conclusion the project successfully achieved all that it was set out to do. It established a functional LoRaWAN network, utilises GNSS and accelerometer data to provide practical assistance and provides a user-friendly frontend to visualise and manage the data. As well as this core feature, I also expanded it with the added the feature of sending WhatsApp alerts when a sheep is detected in need of assistance. This is the result of me being able to quickly learn and develop my skills in areas I had experience in, as well as areas with completely new technologies.

In researching and developing the features of this project I learned a lot and enhanced my full stack skills in both the backend using Java, Spring Boot and MongoDB and also in the frontend using Next.js, NextUI, TailwindCSS. I was happy to be able to learn and successfully implement all the features of the LoRaWAN network. This provided a great challenge for me as there is little to no information available for the specific devices I used and I had to research and adapt information available for other devices to suit the needs of my own. For the case of the Robustel LG5100 I had get in contact with employees from the company itself to acquire information not available online.

I also got the chance to develop my project management skills by using Jiras timeline features as well as trying out both kanban and scrum project management methods.

Overall I found that not only did this project provide me with valuable experience I can apply to both personal and professional projects in the future. I believe I created a valuable and accessible tool that if expanded to use multiple nodes to track multiple animals, as well as expanded to use multiple gateways placed in various areas in rural Spain or France, forest fires could be significantly reduced.

15 Appendix

15.1 (Data Sheets/User Manuals)

Robustel LG5100: https://www.robustel.com/product/lg5100-industrial-edge-computing-lorawan-gateway/

STEVAL-ASTRA1B: https://www.st.com/en/evaluation-tools/steval-astra1b.html#documentation

15.2 (Bill of Materials)

Item	**Quantity**	**Manuf**	**Manuf No**	**ATU Stores**	**Sourced from**	**Order No**	**Cost Euros**
STEVAL-ASTRA1B	1	STMicroelectonics	STEVAL-ASTRA1B	n	Farnell	3993579	134.48
STLINK-V3MINIE	**1**	STMicroelectonics	STLINK-V3MINIE	**n**	Farnell	3953430	11.64
Robustel LG5100 (Loan)	**1**	Robustel			TITAN Group		
Total							**146.12**

16 References

[1] T. Alice, ‘Wildfire season has started. Here’s what Europe’s doing wrong | Euronews’, euronews. Accessed: Apr. 26, 2024. [Online]. Available: https://www.euronews.com/my-europe/2023/06/25/wildfire-season-has-started-heres-what-europes-doing-wrong

[2] ‘What are LoRa and LoRaWAN? | The Things Network’. Accessed: Apr. 26, 2024. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

[3] ‘What is the Technology Behind LoRa Frequency | antennas and accessories’, Interline. Accessed: Apr. 26, 2024. [Online]. Available: https://interline.pl/Information-and-Tips/What-Technology-Behind-LoRa-Frequency

[4] A. Jain, ‘What is Chirpstack?’, Engineers Garage. Accessed: Apr. 26, 2024. [Online]. Available: https://www.engineersgarage.com/what-is-chirpstack/

[5] ‘LG5100 | Industrial LoRaWAN Gateway | Robustel’, Robustel. Accessed: Apr. 26, 2024. [Online]. Available: https://www.robustel.com/product/lg5100-industrial-edge-computing-lorawan-gateway/

[6] ‘STEVAL-ASTRA1B - Multiconnectivity asset tracking reference design based on STM32WB5MMG and STM32WL55JC - STMicroelectronics’, STMicroelectronics. Accessed: Apr. 26, 2024. [Online]. Available: https://www.st.com/en/evaluation-tools/steval-astra1b.html

[7] ‘STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics’, STMicroelectronics. Accessed: Apr. 26, 2024. [Online]. Available: https://www.st.com/en/development-tools/stm32cubeide.html

[8] ‘FP-ATR-ASTRA1 - STM32Cube function pack for the STEVAL-ASTRA1B multiconnectivity asset tracking reference design - STMicroelectronics’, STMicroelectronics. Accessed: Apr. 26, 2024. [Online]. Available: https://www.st.com/en/embedded-software/fp-atr-astra1.html

[9] ‘LoRaWAN 1.0.x packet decoder’. Accessed: Apr. 26, 2024. [Online]. Available: https://lorawan-packet-decoder-0ta6puiniaut.runkit.sh/

[10] ‘What is Java Spring Boot?—Intro to Spring Boot | Microsoft Azure’, Azure. Accessed: Apr. 26, 2024. [Online]. Available: https://azure.microsoft.com/en-gb/resources/cloud-computing-dictionary/what-is-java-spring-boot

[11] ‘React Foundations: About React and Next.js | Next.js’, Vercel. Accessed: Apr. 26, 2024. [Online]. Available: https://nextjs.org/learn/react-foundations/what-is-react-and-nextjs

[12] ‘Mapbox GL JS | Mapbox’. Accessed: Apr. 26, 2024. [Online]. Available: https://docs.mapbox.com/mapbox-gl-js/guides/

[13] ‘Navbar | NextUI - Beautiful, fast and modern React UI Library’, Vercel. Accessed: Apr. 26, 2024. [Online]. Available: https://nextui.org/docs/components/navbar

[14] ‘Background Color - Tailwind CSS’. Accessed: Apr. 26, 2024. [Online]. Available: https://tailwindcss.com/docs/background-color

[15] ‘Tailwind CSS Cheat Sheet’, Tailwind Components. Accessed: Apr. 26, 2024. [Online]. Available: https://tailwindcomponents.com/cheatsheet/

[16] ‘Vector Icons and Stickers - PNG, SVG, EPS, PSD and CSS’, Freepik. Accessed: Apr. 26, 2024. [Online]. Available: https://www.flaticon.com/

[17] ‘HTML Responsive Web Design’. Accessed: Apr. 26, 2024. [Online]. Available: https://www.w3schools.com/html/html_responsive.asp

[18] R. Lynn, ‘Kanban vs. Scrum: What are the Differences? | Planview’, planview. Accessed: Apr. 26, 2024. [Online]. Available: https://www.planview.com/resources/guide/introduction-to-kanban/kanban-vs-scrum/

[19] J. Shen, ‘The Ethical Implications of the AirTag — HUMANITIES+ Journal’, HUMANITIES+ JOURNAL . Accessed: Apr. 27, 2024. [Online]. Available: https://www.hplusjournal.com/home/the-ethical-implications-of-the-airtag