DTU Fotonik
Department of Photonics Engineering

Technical University of Denmark

Bachelor of Science Project

Transmission Device Containing LoRaWAN & Sigfox

Author: Saqib Hameed, s164149

Supervisors: Sarah Renée Ruepp
Martin Nordal Petersen

27-05-2019

Abstract

Internet of Things (IoT) is growing rapidly and the need for Lower Power Wide Area Networks (LPWANs) is crucial for the future growth. IoT applications and services help different sectors of the society to optimise and improve everyday life. This report covers three of these technologies, Narrow-Band IoT, LoRaWAN and Sigfox. The fundamentals of the network architecture and specifications are covered. Moving on to an explanation of the general location technique for GPS and the technologies different localisation techniques. For this project a transmission device is created with GPS positioning containing LoRa and Sigfox chips on a PCB. This device is used to test and compare the two technologies. GPS coordinates are sent via the technologies to their backend server, from where a callback is made to Microsoft's online cloud called Azure. Here the data is fetched in real time, stored in a data set and visualised by pinning the coordinates on a map. The device can be used as a tool for companies to test which technology suits their needs.
The process from designing the device to the implementation is covered. Even a small guide to setting up the technologies and the integration with the cloud is given. Two tests are performed: One by walking and one by driving in a car. These tests are performed in Lyngby Denmark. Based on the results, it seems Sigfox has the advantage in terms of reliability when sending a data packet while walking with a success rate of 100% to LoRa's 66%. This can be due to the interference Sigfox creates for LoRa. While driving, LoRa has the edge based on the results from the test with 83.3 % to Sigfox's 75% succes rate. For future development it is ideal to add the NB-IoT technology to the device.

Project info

Project title: Transmission Device Containing LoRaWan & Sigfox

ECTS Points: 15

Start date: 04-02-2019

End date: 27-05-2019

Supervisors: Martin Nordal Petersen & Sarah Renée Ruepp

Problem statement:

- **Create a compact IoT transmission device with GPS positioning containing LoRa and Sigfox chips on the same PCB.**
- *Explain the technologies and compare them empirically.*
- *Analyse and discuss the results.*
- *Visualise the results on an online platform.*

Contents

1 Introduction **8**
1.1 IoT Use cases . . . 9
1.1.1 IoT Appliance . . . 9
1.1.2 IoT Industry . . . 9
1.1.3 IoT Personal . . . 9
1.1.4 IoT Public . . . 9

2 Theoretical overview **10**
2.1 Low Power Wide Area Network . . . 10
2.2 LoRa . . . 11
2.2.1 LoRa . . . 11
2.2.2 Network architecture & LoRaWAN . . . 13
2.2.3 Setting up an end device . . . 14
2.2.4 Regional differences . . . 16
2.3 Sigfox . . . 16
2.3.1 Network architecture . . . 16
2.3.2 Radio access . . . 17
2.3.3 Setting up a device . . . 18
2.4 LTE . . . 18
2.5 Narrow-band Internet-of-Things . . . 19
2.5.1 Network Architecture . . . 20
2.5.2 Radio Access . . . 21
2.5.3 Improvements in newer releases . . . 22
2.6 Specification comparison . . . 22
2.7 Localisation . . . 23
2.7.1 Trilateration . . . 23
2.7.2 Multilateration . . . 25
2.7.3 Localisation for IoT . . . 25

3 System Design **27**
3.0.1 Requirements & hardware design . . . 27
3.0.2 Solution . . . 30
3.0.3 Data flow . . . 30

4 Implementation **32**
4.1 LoRa connection . . . 32
4.1.1 Sigfox connection . . . 33
4.1.2 Cloud integration . . . 34
4.1.3 The Arduino Code/Functionality for the End Product . . . 39

5 Field tests **42**
5.1 Verification . . . 42
5.2 Tests . . . 42
5.2.1 Test 1: Testing the device in a car around campus . . . 44
5.2.2 Test 2: Testing the device walking around campus . . . 47

6 Future work & Discussion **50**
6.1 Hardware 50
6.2 Software 51
6.3 Tests 51

7 Conclusion **53**

8 Appendix **56**
8.1 Appendix A 56
8.2 Appendix B 58
8.3 Appendix C 60

Acronyms

3GPP 3rd Generation Partnership Project.

ABP Activation By Personlisation.

ADR Adaptive Data Rate.

AES Advanced Encryption Standard.

AOA Angle of Arrival.

API Application Programming Interface.

BPSK Binary Phase Shift Keying.

BW Bandwidth.

CIoT Cellullar Internet of Things.

CP Control Plane.

CP-CIoT Control Plane Cellular Internet of Things.

CR Code Rate.

CSS Chirp Spread Spectrum.

DSL Digital Subscriber Line.

E-CID Enhanced cell-ID.

e-NodeB Evolved-NodeB.

E-UTRAN Evolved UMTS Terrestrial Radio Access Network.

EPC Evolved Packet Core.

ETSI European Telecommunications Standards Institute.

FSK Frequency Shift Keying.

GFSK Gaussian Frequency Shift Keying.

GPS Global Gositioning Gystem.

GSM Global System for Mobile Communications.

HSPA High Speed Packet Access.

IEEE The Institute of Electrical and Electronics Engineers.

IMS IP Multimedia Subsystem.

IoT Internet of Things.

IP Internet Protocol.

ISM Industrial, Scientific, and Medical Band.

JSON Javascript Object Notation.

LoRa Long Range.

LoRaWAN Long Range Wide Area Network.

LPWAN Low Power Wide Area Network.

LTE Long Term evolution.

LTE-A Long Term evolution Advanced.

M2M Machine-to-Machine.

MAC Media Access Control.

MME Mobility Management Entity.

NB-IoT Narrow Band Internet of Things.

NRSRP Narrowband Reference Signal Received Power.

OFDM Orthogonal Frequency Division Multiple.

OFDMA Orthogonal Frequency Division Multiple Access.

OTAA Over The Air Activation.

OTDOA Observed Time Difference of Arrival.

PGW Packet Data Network Gateway.

PRS Positioning Reference Signal.

QPSK Quadature Phase Shift Keying.

RSSI Received Signal Strength Indicator.

SC-FDMA Single Carrier-Frequency Division Multiplexing Access.

SF Spreading Factor.

sGW Serving Gateway.

SIM Subscriber Identity Module.

SMS Short Message Service.

SNR Signal to Noise Ratio.

TDOA Time Difference of Arrival.

TOA Time of Arrival.

UART Universal Asynchronous Receiver/Transceiver.

UE User Equipment.

UICC Universal Integrated Circuit Card.

UNB Ultra Narrow Band.

UP User Plane.

UP-CIoT User Plane Cellular Internet of Things.

1 Introduction

The internet has become an increasingly essential part of the 21st century. More and more devices are connecting to the internet, and it is stated that by 2030 around 125 billion devices will be connected[1]. A substantial amount of these devices are going to be Internet of Things (IoT) devices, where some devices cannot rely on the commonly used wireless technologies such as Bluetooth or WiFi, because they are not suitable for a given use case. IoT is about connecting things around the world to the internet.
IoT devices do not consume as much as other devices. They tend to use less bandwidth, memory, processing power etc. It is often a device sending some sort of sensor data to a backend through some gateway. A simple use case could be a tracking device to track a parcel. For this use case WiFi and Bluetooth's communication range would be limited and there would be no need for such high bit rates. This use case would require a wireless technology with a much longer communication range. There are different wireless technologies out there which can satisfy IoT devices demands. Three of the emerging ones are NB-IoT, LoRaWAN and Sigfox.

This report is based on the structure from the following guidelines[2]. It will give a brief introduction to what IoT is, following an explanation of the three technologies with the main focus on Sigfox and LoRaWAN. A basic introduction to the technique behind GPS and the different techniques used by the technologies will be covered. The design and implementation phase of a transmission device with a GPS module containing Sigfox and LoRa on the same PCB will be addressed. Two tests are performed with this device, where the results will be analysed and evaluated. These results will compare the two technologies in terms of packets loss. At the end of the report, the future improvements will be discussed. This report will help future IoT developers in Lyngby (DK) to choose the best suited wireless technology for their specific requirements.

1.1 IoT Use cases

IoT is rapidly growing and having a bigger influence worldwide. It can be used for various applications and services. According to Huawei[12] they can be classified under four different categories: Appliance, Personal, Public and Industry.

1.1.1 IoT Appliance

IoT Appliance also referred to as smart home is everyday devices at homes that connect to the internet. These devices use short range technologies such as Zigbee. In order for this to be a realisation a home needs a gateway. Appliance tries to provide intelligent solutions for users to make the everyday life easier. An example could be a fridge connected to the internet. There are many possibilities and a lot of innovative ideas.

1.1.2 IoT Industry

IoT Industry offers low power wide area applications to improve industries. A few areas could be logistics, assets and smart farming. When a parcel is sent from a to b, IoT can be used to track and locate it. Tracking is very useful in the industry. With small sensors sending small payloads, alerts and actions can be triggered to notify a user. Asset tracking can be done by GPS but also by other methods which this report will address later. In the farming industry different sensors can be used to keep an eye on the crops . By using sensors optimisation can be assured.

1.1.3 IoT Personal

IoT Personal also offers low power wide area network applications. It can be seen as ”smart accessories”, which can optimise health. As an example a smart watch can monitor a pulse, keep track on the amount of calories burnt a day and in general help a user staying updated. Whether its the news, weather or health.

1.1.4 IoT Public

IoT Public focuses on improving the public services, in general offer applications that can serve the public and can be used as the term ”smart city”. As an example it can be applied as metering or alarms. Metering can save time and manpower by allowing a device to send the collected data on electricity usage etc. over a network. Alarms can be applied at homes, industries or wherever a secure system is needed that can ensure safety in case of fire, burglary etc.

2 Theoretical overview

This section introduces LPWAN and the wireless technologies LoRa, NB-IoT and Sigfox. It also gives an introduction to localisation methods in general and for IoT.

2.1 Low Power Wide Area Network

Low power wide area network (LPWAN) as the name states is a network used for wide areas and consumes low power. It is often used when the more popular wireless networks such as WiFi or Bluetooth are not sufficient enough. They are not optimal for long range communication. Another type of wireless network could be cellular M2M networks, but they are costly and consume a lot of power. They are also expensive in terms of hardware and services. Figure 1 shows the sweet spot for LPWAN in terms of bandwidth and range compared to other wireless technologies.

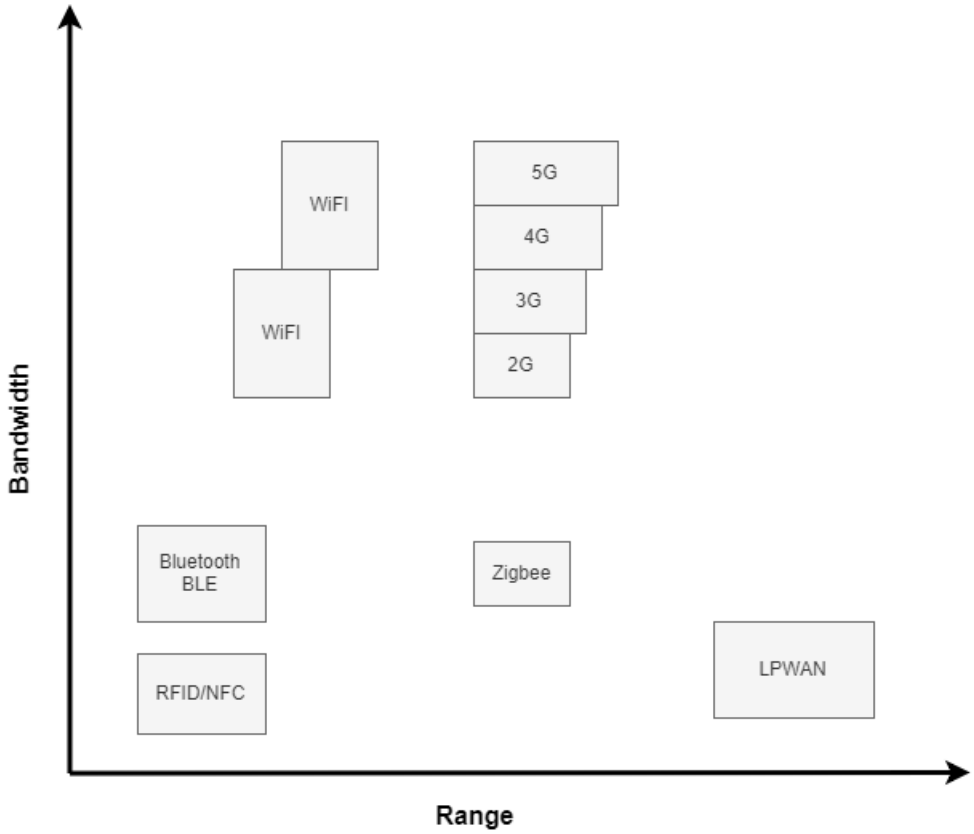


Figure 1: Chart comparing range and bandwidth for different wireless technologies.

LPWAN's are best suited for connecting devices that need to transmit a small amount of data over a long range and not consume a lot of power, so that a long lasting battery life can be maintained. IoT devices in general sends small amount of data for instance a humidity sensor in a basement, which only transmits data when the humidity is over some threshold, which can notify an air purifier to clean the air. A device such as this would consume low power that will allow it to complete its task with minimal cost and battery drain.
Here are the most important LPWAN requirements:

- **Long range:** Depending on which technology is used, the end nodes can be up to 10 km from the gateway.
- **Low data rate:** The data rate is maximum 5000 bits per second and there is approximately 20-256 bytes[3] per message which is sent several times a day.
- **Low power consumption:**
  Low power consumption with long lasting battery life, approximately 5-10 years.

2.2 LoRa

LoRa stands for long range and is a LPWAN developed by Cycleo and later acquired by Semtech. LoRa is the physical layer of the wireless technology. The communication protocols and system architecture is defined by LoRaWAN, whereas LoRa ensures the long range communication. LoRa offers low power consumption, low cost, positioning and more. This technology is useful for applications that need to send a small amount of data over long distances a couple of times an hour or day depending on the use case. The range depends on the environment in a given location, but LoRa has a link budget greater than any other standardised communication technology [4]. In this report the whole technology will be referred to as LoRa.

2.2.1 LoRa

The long range which LoRa offers comes by using chirp spread spectrum(CSS) as modulation. This technology is a Semtech proprietary technology and therefore not fully open, but the parts that are open will be covered. The CSS technique maintains the same low power as frequency shift keying, which is used by other wireless communication systems but increases the communication range. This is a trade off between bit rate and communication range. Chirp is when a signal's frequency increases or decreases with time. Spread spectrum is a technique used in wireless communication to spread a signal in the frequency domain which results in a wider bandwidth. Chirp spread spectrum uses wide-band linear frequency modulated chirp pulses to encode information. The Lora modulation makes it more robust to noise and interference.
The physical layer LoRa operates on the open license 430, 433, 868 or 915 MHz ISM bands depending on which region it is used. The bit rate can reach up to 50 Kbps[4] and the payload can range from 2-255 octets for every transmission. Figure 2 shows how the LoRa protocol stack is. LoRaWAN is the MAC and application layer, where LoRa is the physical layer. There are different types of devices, which is the reason for the three classes. Each class has a different battery lifetime and communication parameters such as latency. Depending on the application, a class can be chosen for that applications requirements.

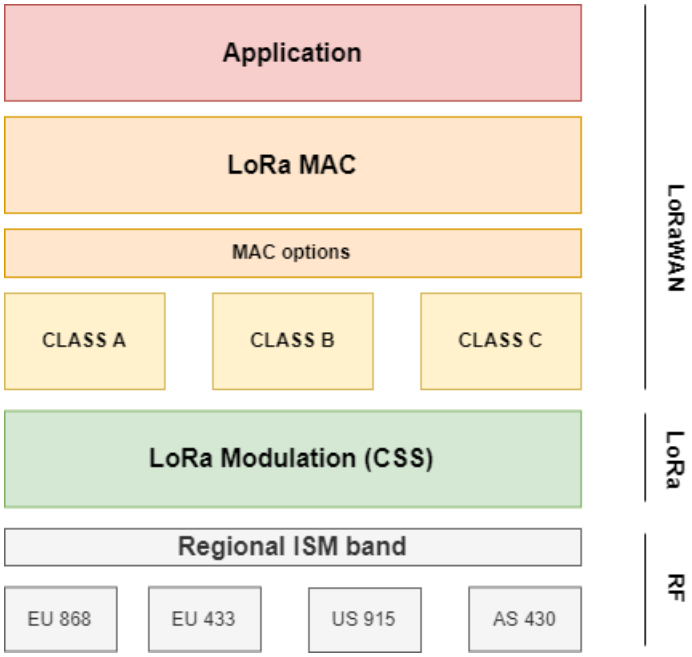


Figure 2: LoRa protocol stack.

LoRa utilises other techniques to improve upon CSS, so it can become less prone against noise and interference. Three parameters influence the bit rate of the modulation, the resistance to noise and interference and the decoding - Bandwidth(BW), Spreading Factor (SF) and Code Rate (CR).
The bandwidth plays the biggest role in the modulation. A LoRa symbol is equal to 2^{SF} [5] chirps. A symbol can encode SF bits of information. This covers the entire frequency band. LoRa can have a SF between 7-12 and the higher SF, the more resistant it becomes to noise and interference. An SF of 12 would seem ideal to use all the time, but that is not the case because SF of 12 also has the lowest bit rate.

$$T_s = \frac{2^{SF}}{BW} \tag{1}$$

Equation 1 shows the relation between SF, BW and the duration of a symbol (T_s). With a high SF a device needs to transmit for a longer period in order to transmit the same amount of data with a lower SF. Therefore there is a possibility to use adaptive data rate(ADR), which will try to use the lowest possible data rate in order to keep the transmission time as low as possible. The bit rate can be defined as:

$$R_b = SF \cdot \frac{BW}{2^{SF}} \cdot CR \tag{2}$$

, where R_b is the bit rate and CR=4/(4+n), n $\in$ 1,2,3,4.
As figure 3 shows the signal starts with upward chirps until the maximum frequency is reached, then the increase starts again from the minimum frequency.

A physical frame format is specified in Semtechs transmitters and receivers. Figure 4 shows a LoRa frame that starts with a preamble, then an optional header, payload and at last an optional cyclic redundancy check.

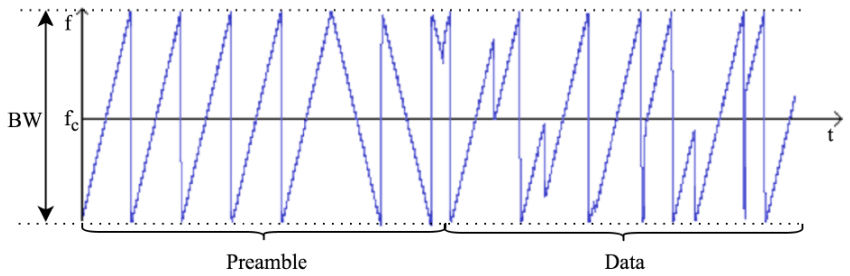


Figure 3: An example of frequency variation over time of a signal emitted by a LoRa transmitter. f_c is the central frequency of the channel and BW is the bandwidth. [5]

Figure 4: Structure of LoRa frame.

2.2.2 Network architecture & LoRaWAN

As mentioned LoRaWAN is a MAC protocol that uses the LoRa physical layer. It is designed for IoT sensors. These sensors exchange packets with the server with low data rates and long time intervals (once per hour or day).

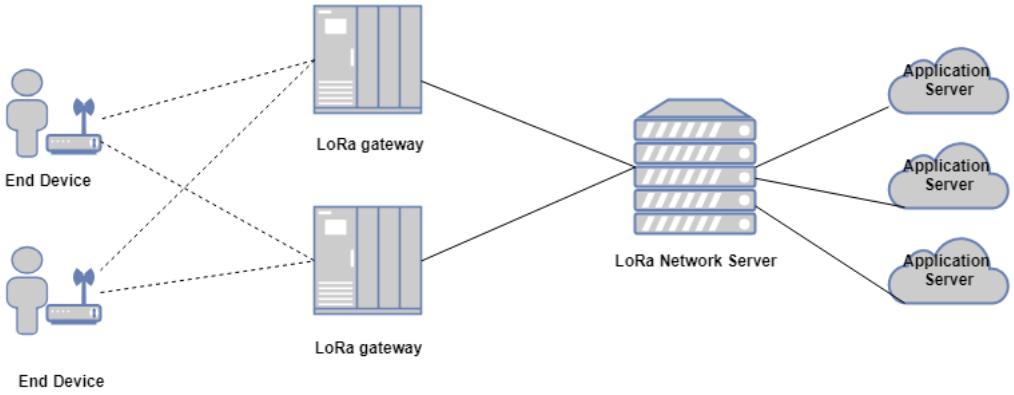


Figure 5: Network architecture of Lora.

A LoRa network uses a long range star architecture as seen on figure 5. This architecture insures longer battery lifetime preservation for long range connectivity. The network consists of end nodes, LoRa gateways and a network server. In a LoRa network end user devices are not associated with a specific gateway like a UE with a eNode-B in a cellular network. Instead end user devices transmit data to multiple gateways. The gateways act as link layers that forwards packets received from the user devices to the network server. The gateways are also responsible for adding information about the transmitted signal before forwarding the packets. In order to have a big network capacity, the gateway is built so it can handle multiple messages received from nodes. It also uses a multichannel multi-modem transceiver. This allows it receive simultaneous messages on multiple channels.

A transmission in this LoRa network would typically be an end node communicating with a LoRa gateway using LoRa. This gateway would forward LoRaWAN frames from the end node to a network server over a high speed backhaul interface which uses IP. It is the network server which demultiplexes the packets and multiplexes the packets that has to be sent back to the end node. In a LoRa network the complexity lies in the network server. Here there are different intelligent functionalities such as deleting duplicate packets, choosing the gateway for sending an acknowledgement and performing adaptive data rate. There is also no need for handovers for mobile devices as in a cellular network. This enables tracking possibilities.

In LoRa there are different device classes which was briefly mentioned when introducing the protocol stack. The end devices are created differently so they can serve different application requirements. The device classes have a trade off between network downlink communication latency and battery life. These devices are asynchronous which means they communicate with the LoRa gateway when they have data to send, whether it is event driven or scheduled. This method is referred to as ALOHA and is used in mesh networks. Comparing this with a cellular network, where the nodes frequently wake up to synchronise with the network and check for messages. This procedure consumes a lot of energy and therefore is not ideal for an IoT device to implement.
The three devices classes:

- **Class A:**
  Class A end devices have bi-directional communication capabilities. It is battery powered and the most energy efficient. Downlink communication is only available short time after a sensor has transmitted. When it comes to transmission scheduling a type of ALOHA protocol is used.

- **Class B:**
  Class B is an addition to class A. It opens extra receive windows at scheduled times. The end device receives a time synchronised beacon from the gateway which will allow the device to open its receive window at the scheduled time. This makes class B less energy efficient than class A.

- **Class C:**
  Class C devices are powered because they use the most energy. This allows the device to listen continuously except when they are transmitting. By listening almost all the time it gives no latency for downlink communication.

Figure 6 shows how the ALOHA type protocol LoRa uses works. A device opens two receive windows after an uplink transmission as seen in situation 1. The server shall only answer to one of the windows, if it does not respond to any, it will have to wait until the device transmits uplink data again. Situation 2 and 3 shows the server responding in the first and second receive window, whereas the situation on the right is the class C device that keeps the receive windows open except when it is transmitting.

2.2.3 Setting up an end device

For a device to join a LoRa network, it must be activated. This can be done in two ways. Either via Over The Air Activation (OTAA) or Activation By Personalisation (ABP). The

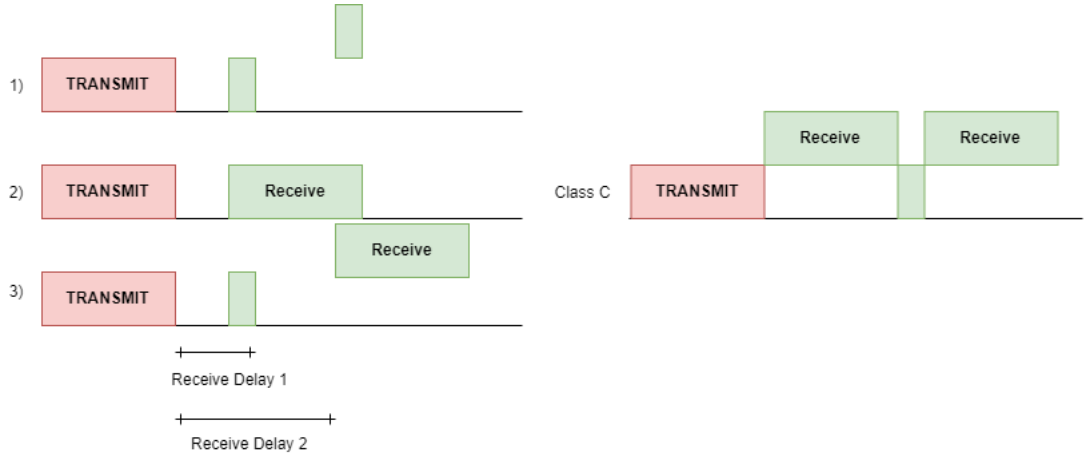


Figure 6: Illustration of the bi-directional communication.

following parameters are essential to join the network:

- *End device adress (DevAddr):* A 32-bit identifier for end devices that is not unique. Of them seven bits are for the network identifier and 25 are for the network address of the device.
- *Application identifier (AppEUI):* A unique 64-bit application identifier in the IEEE EUI64.
- *Network session key (NwkSKey):* A 128-bit key is used for the communication between end devices and the network server. It is used for data integrity and is used to verify every message.
- *Application session key (AppSKey):* A 128-bit key used to encrypt and decrypt the payload of a message. It is used by the network server and the end device.
- *Application Key (AppKey):* A 128-bit key sued to derive the AppSKey and NwKSkey during the activation phase.
- *Hardware identifier (HWEUI):* A 64-bit address which every end device has. It is registered on the network manually, when activating the device deployment.

In Denmark Teracom[6] is the go to provider in terms of LoRa gateways. They have gateways around Denmark and also has Denmark's only LoRa-server, where all the sensor data will be sent. To register a LoRa device, it has to be done on Teracom's server, which will be addressed further in the implementation section. It is also possible for a user to setup a gateway and a network server themselves.

OTAA is the preferred option security wise. An end device performs a join procedure, where the devAddr is dynamically assigned and security keys are exchanged. With ABP the devAddr is not dynamically assigned. It is hardcoded in the devices and does not have a join procedure, which gives a faster connection but it is less secure.

Security is also very important in communication. LoRaWAN uses two layers of security. For the network it insures authenticity of the nodes in the network. For the application layer it insures the network operator does not have access to the end nodes application data. Both layers are encrypted using AES. This aspect will not be covered further in this report.

2.2.4 Regional differences

LoRa specifications varies from region to region because of different spectrum allocations and regulatory requirements. This report focuses on its use in Europe, more specifically Denmark. In Europe LoRa has ten channels, where eight of them are multi data rate varying from 250 bps-5.5 kbps. There is also a single high data rate channel at 11 kbps and a single FSK channel that goes up to 50 kbps. In Europe LoRa operates at the frequency band 867-869 MHz which is an unlicensed band. The ISM band in Europe is defined by European Telecommunications Standards Institute (ETSI). They have split the unlicensed band into five subbands where LoRa operates in some of them. There is a duty cycle of 1% set by them but it does vary depending on the subband. In the range LoRa operates, the maximum output power allowed is 14 dBm.

2.3 Sigfox

Sigfox is a french network operator that builds LPWA (Low Power Wide Area) IoT networks. Like LoRa, Sigfox offers long range communication for low powered devices but unlike LoRa Sigfox operates as a network operator. It provides connectivity for IoT devices without having to deploy network infrastructures for each application. This means that customers do not need to spend a ton of money on network equipment. A device is required to be Sigfox certified in order to join the network. It has similarities to cellular networks but with less functionlities and for devices that only need to send small amounts of data. Like other cellular networks, Sigfox sets up base stations.

2.3.1 Network architecture

Figure 7 shows the network architecture of Sigfox. As seen there are end nodes, Sigfox base stations and a Sigfox cloud. Sigfox has many partners such as IBM, Microsoft etc. which offer their cloud service to be "integrated" to the Sigfox network. High hardware costs have been overcome due to the Sigfox software defined radio. A simple data flow scenario would be a device sending data wirelessly to a base station, which forwards it through to the backhaul network. The backhaul network uses DSL and 3G or 4G as backup. The core network manages the base stations, monitors the network and processes the transmitted the data. The data is stored in two locations, so the data can be used for services and keep the data available for customers to retrieve the data whenever they like. At last a customer can access their data via web interface or APIs. The actual data that is sent can have a maximum payload size of 12 bytes. This is sufficient enough for a lot of use cases in the IoT world. The communication is bi-directional but for downlink data the payload is decreased to 8 bytes, because it is not a necessity to have more data in order to trigger an event, a device could react to.

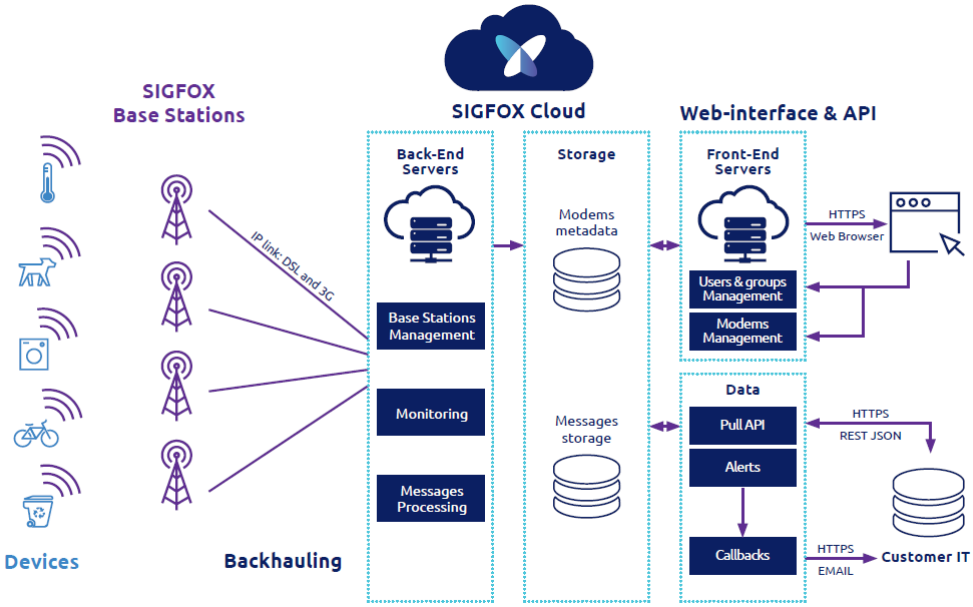


Figure 7: Sigfox network architecture [7].

2.3.2 Radio access

In Europe Sigfox uses the unlicensed ISM band between 868-868.2 MHz, where it also needs to comply with the duty cycle regulations set by ETSI. Of this band Sigfox uses 192 KHz to exchange messages with. This band utilisation is called ultra-narrow band(UNB). Depending on the region, a message is sent with a bit rate of 100-600 bits per second[7] and is 100 Hz wide.

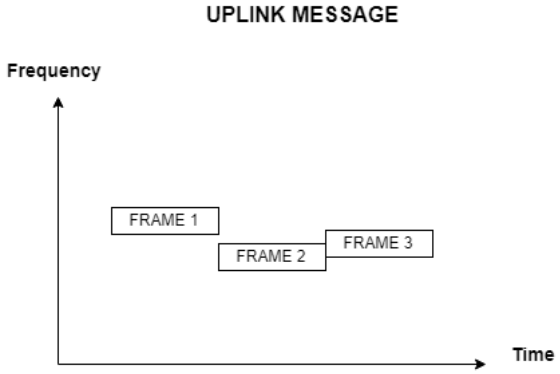


Figure 8: A sent message with two replicas at a different time and frequency.

For the random access scheme, it uses a time and frequency diversity scheme that. Every transmission is unsyncronised between the device and network, like LoRa. A device transmits a message on a random frequency, followed by 2 exact copies on a different frequency and time. An illustration of this is shown figure 8. Sigfox have calculated their 12 byte payload to take 2.08 seconds over the air with a rate of 100 bps[7]. When devices are transmitting data, the Sigfox base stations looks for UNB signals in the 192 KHz spectrum. Sigfox has a principle called "Cooperative reception", which means that a device is not attached to a specific base station as in other types of cellular networks. This means that when a message

is sent from a device, it is received by whichever base station that is closest to the device. Sigfox achieves its long range by multiple factors. The low output power, low link budget and low bit rate is the most significant reasons. The radio transmission modulation is binary phase-shift keying (BPSK) uplink and Gaussian Frequency Shift Keying (GFSK). These modulation schemes allows a bit rate of 300 bps uplink and 600 bps downlink. This is considerably lower than LoRa, but due to this Sigfox can provide a larger communication range. With the larger range Sigfox does not need to deploy that many base stations. As mentioned earlier Sigfox is bidirectional, but in a limited matter. A base station can only send a message to a device, when it receives an uplink message from the device. This message shall require a downlink message in order for the base station to reply.

2.3.3 Setting up a device

Setting up a Sigfox device is fairly simple. Once a user has their Sigfox certified product, it must be registered on Sigfox's backend server. When purchasing a subscription, a PAC number is received which is an ID and a product activation key is received. This PAC number is registered on the backend with the specific device type a user is registering when activating the subscription. In Denmark the Sigfox operator is called IoT Denmark[9]. They are responsible for the Sigfox network. Further details on the registration process will be addressed in the implementation section.

2.4 LTE

To understand how Narrow-band IoT works, it is firstly necessary to understand the fundamentals of how the present 4G mobile networks architecture is constructed.
4G also known as Long Term Evolution Advanced (LTE-A) was standardised in release 10 by 3GPP (3rd Generation Partnership Project). This release is the "true 4G" which fulfils the requirements of set by ITU-T (International Telecommunication Union). It is an enhancement on LTE (release 8) which could not fulfil the requirements for 4G, but commercially is branded as 4G.
As each new generation of network expects, this generation offers higher bit rates, lower latency and more capacity. LTE is the first fully packet switched network released by 3GPP. It builds on top of the features for High-Speed Packet Access (HSPA), where more functionality is moved to the node-b (telecommunication node which connects a mobile phone to the wider network). Compared to the previous generation it also has new features such as MIMO antenna systems and new radio access scheme based on OFDMA (Orthogonal Frequency Division Multiple Access) and SC-FDMA (Single Carrier-FDMA), but these will not be addressed further.
To understand how Narrow-band IoT is a part of the existing cellular network, an overview of the LTE network is shown on figure 9. Here it is seen that the network consists of three major components: UE, E-UTRAN and EPC. The different entities in the architecture are:

- UE (User Equipment):
  UE is the user equipment such as a mobile phone for LTE. It has a UICC (Universal Integrated Circuit Card) which is more commonly known as a SIM card(Subscriber Identity Module). The SIM card contains information about the user, unique identifiers for a specific mobile network and keys which are used for authentication, ciphering and security. The UE handles the communication with the eNode-B.

- E-UTRAN (Evolved UMTS Terrestrial Radio Access Network):
  E-UTRAN contains eNode-Bs. All radio related functionality is condensed into this one entity. The eNode-B controls UEs in one cell or more cells. It also handles radio resource management, mobility management, scheduling and retransmissions.

- EPC (Evolved Packet Core):
  EPC has three entities on the simplistic figure of the LTE network, MME, PGW and SGW. MME stands for Mobility Management Entity. It provides the functionalities of mobility management, authentication and security. SGW (Serving Gateway) operates like a router and forwards data between eNodeB and PGW. PGW (Packet Data Network Gateway) is the gateway to the packet data network and communicates with it.

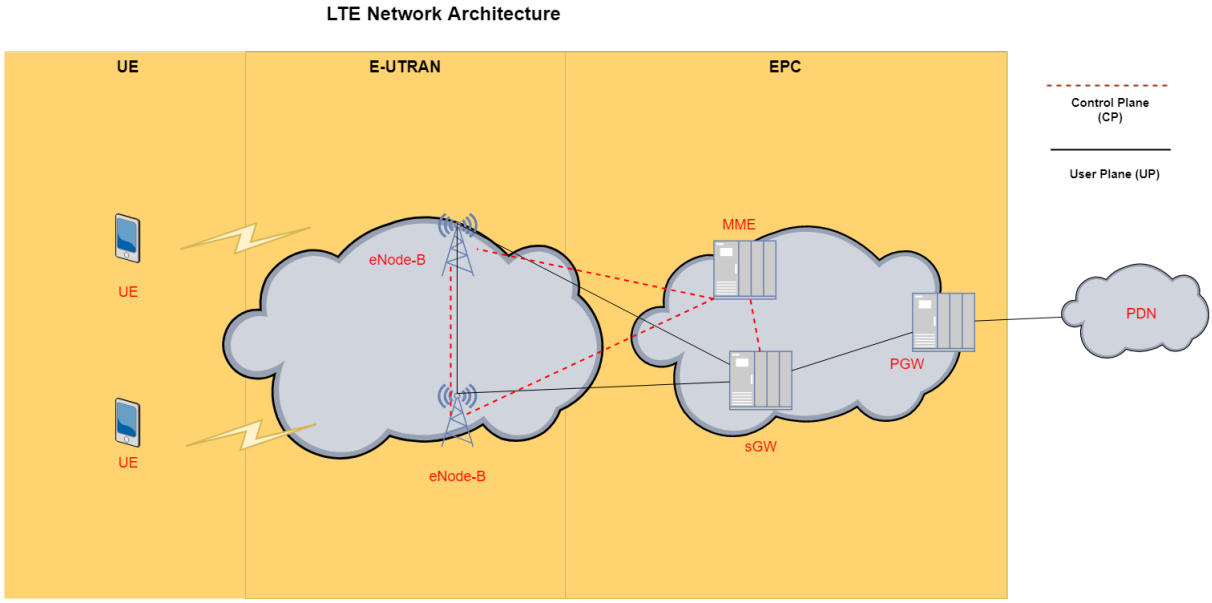


Figure 9: LTE network architecture.

Figure 9 shows the LTE architecture and the three main entities as mentioned. The striped red line indicates the Control plane which handles signalling in the network and the black line indicates the user plane which handles the user data. The separation of these are clear opposites to the previous releases. PDN (Packet Data Network) is "outside" of the architecture because it can either be an internal or external IP domain of a given operator, which will allow the UE access to the internet or IMS (IP Multimedia Subsystem).
The frequency band currently used for LTE in Denmark is 800, 1800 and 2600 MHz[8]. The radio access part and the architecture will be addressed further in the Narrow-band Internet-of-Things section.

2.5 Narrow-band Internet-of-Things

Narrow-band Internet of things, abbreviated as NB-IoT, is a LPWAN standardised by 3GPP as part of release 13. It is based on LTE and its integration with the LTE standard is regarded as a new air interface. This means it is not backward compatible with LTE.
With all these emerging IoT devices, NB-IoT focuses on minimising signal overhead, securing

the system, long lasting battery life, Support Data with and without IP and supporting SMS as a deployment option. For NB-IoT to achieve these requirements, most of LTE and LTE-A features are not supported - not even handovers which means there is no mobility.

2.5.1 Network Architecture

For NB-IoT to be a realisation, some optimisations had to be made. A cellular internet of things (CIoT) was defined in the EPC and the two planes, control and user, were optimised for IoT. Figure 10 shows the architecture of NB-IoT. As seen the user plane and the control plane are optimised for IoT so they can access the CIoT Services.

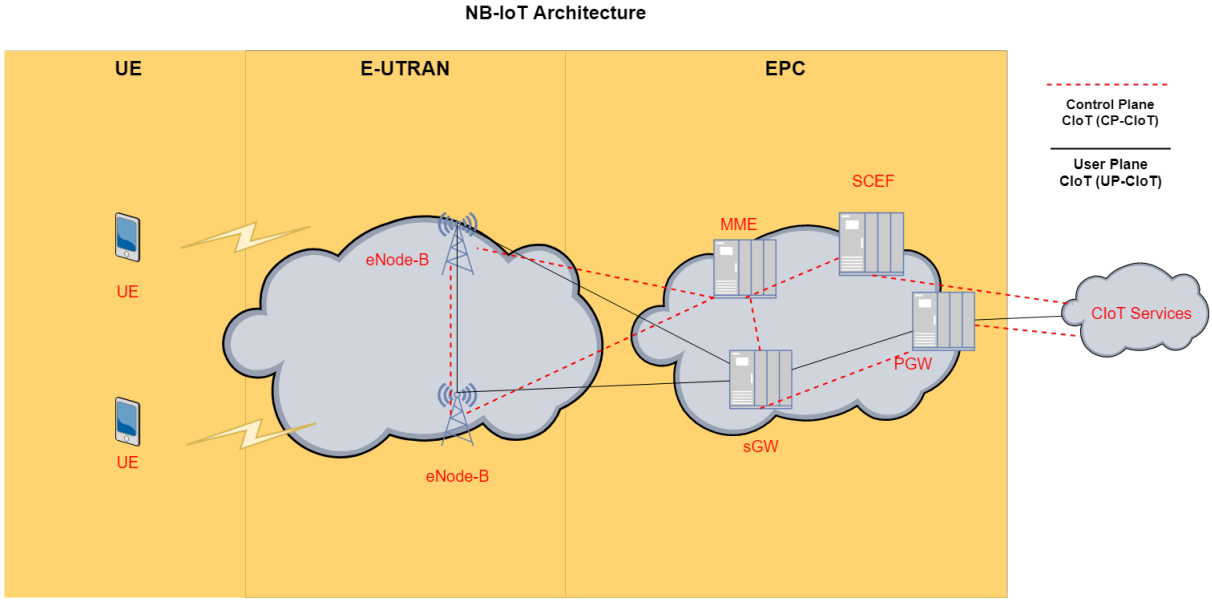


Figure 10: NB-IoT network architecture.

Looking at the transmission flow on the Control Plane CIoT (CP-CIoT), data is sent from the eNode-B in the radio access network. The radio access network is referred to as Cellular Internet of Things Radio Access Network (CIoT RAN). From the eNode-B the data is sent to the MME. As seen on the figure, the data can then either be sent to the sGW or the SCEF (Service Capability Exposure Function). Via sGW the data can be sent to PGW and finally be forwarded to the CIoT services. The other option from MME to SCEF is however only possible for data without an IP. Likewise from the SCEF the data is forwarded to the CIoT Services. The plane is for downlink and uplink data.

Looking at the transmission flow on the user plane CIoT (UP-CIoT), data is sent from an NB-IoT device to the eNode-B, which is the same way as for any other device using LTE. From here the data is sent to the PGW via sGW, from where it is forwarded to the CIoT services. This plane supports both data with and without IP. Unlike LoRa and Sigfox, NB-IoT does not have a backend server, so a users data can be sent to whichever cloud service they like.

Comparing NB-IoTs architecture with LTE's a new node has been added - the SCEF. It is designed for IoT devices and used for delivery of data without an IP via the control plane

as mentioned earlier. It also functions as an interface for the following network services: authentication, authorisation, discovery and various capabilities for the access network.
NB-IoT uses the same low frequency band numbers as LTE and these low frequencies are defined for NB-IoT. The lower the frequency the bigger reach of the signal, which suits NB-IoTs requirements. NB-IoT occupies a frequency band of 180 kHz bandwidth [10]. Comparing it with LTE it corresponds to one resource block in a transmission. NB-IoT will offer three deployment scenarios, Gaurd band, in band and stand alone. Stand alone will be using new bandwidth, guard band will be using the reserved bandwidth in the guard band of LTE and in band uses the same bandwidth as LTE. By new bandwidth it is meant that NB-IoT can utilise the GSM spectrum when available. With a bandwidth of 200 kHz there is a guard interval of 10 kHz on both sides of the spectrum.

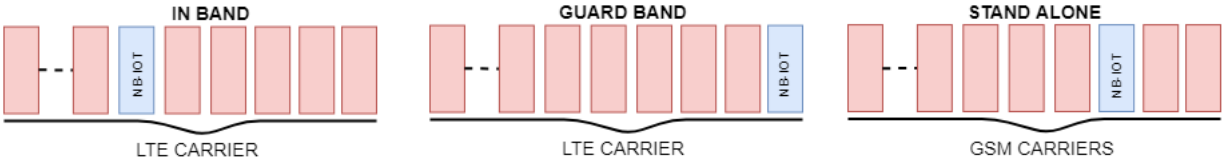


Figure 11: NB-IoT deployment possibilities.

Figure 11 shows the three different deployments options. The resource assignment is not fixed in the in-band deployment but there are some frequencies in the resource block that are allowed to be used for cell selection.

2.5.2 Radio Access

Downlink and uplink are separated in frequency because NB-IoT only supports half duplex. This allows the device to either receive or transmit but not at the same time. As for all 3GPP standards there are a lot of different channels with different functionalities. There are channels such as physical channels, transport channels and physicals signals. These will not be addressed in this report but they do have an important role in the downlink and uplink of data. In the multiple access part for downlink Orthogonal Frequency Division Multiplexing (OFDM) is used. This technique allows 12 subcarriers to be allocated for the 180 KHz bandwidth. The downlink modulation is Quadrature Phase Shift keying (QPSK) which allows around 234.7 kbps[11]. For the uplink multiple access Single Carrier Frequency Division Multiple Access (SC-FDMA) is used. This technique allows 48 subcarriers for the 180 KHz bandwidth. Depending on the amount of used subcarriers in the resource block different modulation schemes can be used for the uplink transmission. The different possibilities are BPSK and Quadrature Phase Shift Keying (QPSK). For the uplink transmission the highest possible bit rate is 204.8 kbps.

2.5.3 Improvements in newer releases

In release 14 by 3GPP[16], there are some new added features and improvements to NB-IoT. Only the most noticeable improvements and major additions will shortly be mentioned. In release 13 NB-IoT was introduced but not with sufficient enough functionalities to be useful for a broad range of use cases. Release 14 adds the possibility of location services that are based on enhanced cell-ID (E-CID) and Observed Time Difference of Arrival (OTDOA). There is also introduced multicast downlink transmission based on using single cell point to multipoint, higher data rates and Support of paging.

Release 15 by 3GPP[17] also have different improvements to enhance the NB-IoT experience. New latency and power consumption reduction techniques are being used. All these improvements can be found in 3GPPs release 15 and will not be addressed further. In the section about localisation, NB-IoTs location services will be further explained.

2.6 Specification comparison

This sections compares the different technologies in terms of specifications.

	NB-IoT	**LoRa**	**Sigfox**
Specification	3GPP	Open	Private
Spectrum	Licensed	Unlicensed	Unlicensed
Channel BW	180 KHz	7.8-500 kHz	100 Hz
Max Payload Size	1600 bytes	243 bytes	12 bytes UL 8 bytes Dl
System BW	180 KHz	125 KHz	200 KHz
Peak Data Rate	UL: 204.8 Kbps DL: 234.7 Kbps	180 bps - 37.5 kbps	UL: 100 bps DL: 600 bps
Max Number Of Messages Per Day	Unlimited	50000 (BTS)	50000 (BTS)
Device Peak Tx Power	23 dBm	14 dBm	14 dBm
Device Power Consumption	Low	Low-Medium	Low
Coverage	22 km	14 km	17 km

Table 1: Specification comparison of NB-IoT, LoRa & Sigfox.

Table 1 shows a comparison between NB-IoT, LoRa and Sigfox. The Specifications are from Telia[11] who together with TDC are the mobile network operators that offer NB-IoT in Denmark. When deciding what technology to use for a given use case it is good to look at the differences between the available technologies. Telia's NB-IoT coverage has a wider range than LoRa and Sigfox, which can be explained with the 9 dBm increase of Tx power which NB-IoT uses. NB-IoT are not restricted to 14 dBm as LoRa and Sigfox because it uses the licensed band of the frequency spectrum. NB-IoT are neither affected by ETSI's duty cycle rule, which means a user can send as many messages per day as needed, whereas LoRa and Sigfoxs base stations are limited to 50000 messages per day. NB-IoT claim to have low power consumption but there are not any concrete values for how it stacks up against the other two technologies. Cellular networks are very power consuming but with some tweaks in the newer releases the power consumption has been optimised.

2.7 Localisation

In the early days people used to use the stars to navigate while sailing across the ocean or exploring the earth. Localisation has always been an important practice in order to navigate around the world we live in. In the modern world of technology there have been developed different methods to calculate locations. The common techniques are triangulation, trilateration and multilateration. The most common navigating system among others is Global Positioning System (GPS) which uses trilateration. In this section different localisation methods will be introduced and point to which ones can be used for LoRa, Sigfox and NB-IoT.

2.7.1 Trilateration

To locate a device different location techniques can be used. Triangulation uses the geometry of a triangle to locate a device. Two end-points (base stations) form this triangle with the device. By using Angle of Arrival (AoA)[18] of a signal sent from either one of the end-points or the device, the location can be calculated based on the geometry of a triangle. This technique is not suitable for long range and will therefore not be discussed further.

Trilateration is based on measuring distances and is used in GPS among other. Figure 12 shows an example of it in 2D. For GPS, satellites are used instead of radio towers but the principle is the same. By having three satellites, GPS sends information to the neighbour satellites about their location and gives a timestamp. The distance between a satellite and the device that has to be located, is equal to the radius of a satellite's circle(Range) as seen on the figure. The distance can be determined by the time of arrival (TOA), the time of flight (TOF) or the received signal strength indicator (RSSI). Once the distance from each satellite has been determined by multiplying the time it took to transmit from a satellite to the device and the electromagnetic waves speed which is the speed of light in vacuum, all three circles will intersect and a approximation of where the device is located will be given. In order for this to be as precise as possible the satellites must be synchronised.

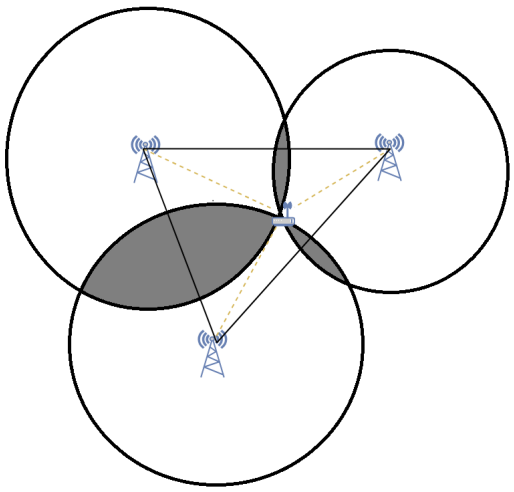


Figure 12: Trilateration example.

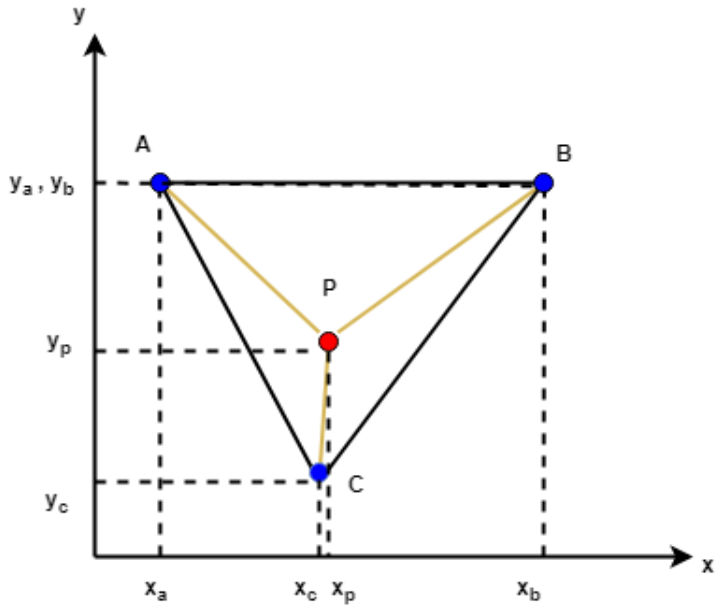


Figure 13: 2D Cartesian coordinate system example of trilateration.

To mathematically calculate the distances between the satellites and the device, figure 13 has been made to illustrate the situation in a 2D Cartesian coordinate system. The red point P, represents a device, where the blue dots A, B and C represent the blue points. In this example it is assumed that location of A, B and C are known and the lengths $|AP|$, $|BP|$ and $|CP|$. By using simple trigonometry, the distances from the point P to the satellites can be determined by the following relations:

$$PA = \sqrt{(x_a - x_p)^2 + (y_a - y_p)^2} \tag{3}$$

$$PB = \sqrt{(x_b - x_p)^2 + (y_b - y_p)^2} \tag{4}$$

$$PC = \sqrt{(x_c - x_p)^2 + (y_c - y_p)^2} \tag{5}$$

The three equations are now squared, expanded and simplified to two equations:

$$\begin{cases} PA^2 - PB^2 = x_a^2 - x_b^2 - 2 \cdot x_a \cdot x_p + 2 \cdot x_b \cdot x_p + y_a^2 - y_b^2 - 2 \cdot y_a \cdot y_b + 2 \cdot y_b \cdot y_p \\ PA^2 - PC^2 = x_a^2 - x_c^2 - 2 \cdot x_a \cdot x_p + 2 \cdot x_c \cdot x_p + y_a^2 - y_c^2 - 2 \cdot y_a \cdot y_c + 2 \cdot y_c \cdot y_p \end{cases} \tag{6}$$

From here the variables can be separated from the constants and the two equations can be solved using a non-iterative or iterative algorithm. After solving the value for x_p and y_P will be determined. This method is also used in multilateration, where the difference lies in the coordinate system which is in 3D.
To get a more precise location a fourth satellite is used in order to compensate for the lack of an atomic clock in the devices receiver.

2.7.2 Multilateration

Multilateration is similar to trilateration but in this method only the transmitters have to be synchronised, not the receiver. This is because multilateration uses the time difference of arrival (TDOA). The method does not need to know the exact distances from the receiver to the transmitter, only the differences in the distances which is determined by finding the differences in the timestamps for each transmitter to the receiver. With this approach there is calculated one time difference at a time for each pair of transmitters. Therefore TDOA can be defined as[19]:

$$\Delta t_{i,j} = t_j - t_i \tag{7}$$

,where t_i and t_j are the timestamps for two transmitters.
When the TDOAs have been calculated there are different methods to calculate the actual position of a given device. There are Cartesian and spherical solutions with iterative and non-iterative algorithms. Using all the TDOAs a hyperbola can be created to determine where about a given device could be located.

2.7.3 Localisation for IoT

GPS is as accurate as one can come to determining a position, but it might not be the best localisation solution for an IoT product because of its excessive power consumption, at least not for battery powered products. Now the different localisation methods for LoRa, Sigfox and NB-IoT will be introduced.

Starting with LoRa, it uses TDOA as the geolocation method[21]. This method allows LoRa to offer a geolocation service without acquiring additional processing power and cost. LoRa can use either RSSI with a theoretical accuracy of 1-2 km or TDOA with a theoretical accuracy of 20-200 m as their geolocation method. The basics behind RSSI is using the RSSI from the transmitting device to gateway. By looking at the RSSI, the signal power that is lost under the transmission, the distance of the signal can be calculated. For this method to work optimally it shall be used where there is clear line of sight because of interference and collisions under the transmission. RSSI is better to use for indoor situations and is used for Bluetooth low-energy beacons. Therefore TDOA is the best solution for LoRa in terms of precision. It could be used for geo-fencing or tracking slow moving objects. Looking at the downsides, then TDOA is not suited for real time tracking of mobile objects or high-precision positioning.

Sigfox do not offer TDOA as a possibility for geolocation. Like LoRa Sigfox devices do not need to upgrade the hardware or software in order to use the service. Sigfox's geolocation is called Atlas. It is based on RSSI and is suitable for use cases where precision is not that important such as tracking a parcel. When sending a message, Sigfox provides real time coordinates in latitude and longitude, which can be delivered through callback or an API. Sigfox's accuracy varies a lot depending on where the device is located. The precision can range from 1-10 km[22] depending on the network coverage and amount of base stations that receive the sent message. Sigfox suggest using Sigfox's geolocation with either WiFi or GPS if higher precision is needed. These options require extra hardware which gives more cost and uses more power, especially GPS.

As mentioned NB-IoT release 14 added positioning features. The two methods are Observed Time Difference of Arrival (OTDOA) and Cell ID (CID) which already exits for LTE. OTDOA is a multilateration method, where the NB-IoT UE measures the TOA of the signal from the eNode-B or any other transmission point, then sends the reference signal time difference (RSTD) to the location server. The distance between a UE and eNode-B is calculated by using TOA on the positioning reference signal (PRS). NB-IoT's use of narrow bandwidth causes limitations to the accuracy of positioning. If more accurate positioning is needed, there is a possibility to upgrade the hardware on the UE in order to use wideband-based RSTD measurements on the existing PRS with a PRS transmission bandwidth of 1.4, 3, 5, 10, 15 or 20 MHz[17]. The positioning is also possible in idle mode.
The other method CID, it uses the position of a UE based on the location of the serving cell which means the accuracy is dependent on the coverage of the cell. Another method E-CID positioning uses additional measurement information for further enhancement it uses Narrowband Reference Signal Received Power (NRSRP) among others, This is reported to the location server.
In LTE these two methods are used for emergency and to compensate GPS when a device is for instance indoors or underground.

3 System Design

This section will describe the design phase of the product.

3.0.1 Requirements & hardware design

Before designing a product it is important to figure out which requirements it should have. Doing the right research is vital in order to find out, if the hardware can live up to the desired specifications. The product for this project is intended to be a transmission device with a GPS module containing LoRa and Sigfox on the same PCB. By pushing a button, the device should send latitude and longitude coordinates to an online cloud via the two technologies. To achieve this, the product for this project has to include a LoRa module, Sigfox module, a microcontroller and GPS module in order to send coordinates via these technologies. The GPS or more accurate GNSS module can be used for GPS, GLONASS, Galileo and QZSS, but will be using GPS as introduced in the theory. In terms of cost and performance it is good, even tough the newer version NEO 8N was the intended module. It also has a sleep mode, which can be useful for preserving a battery powered device. The following hardware is used for the product:

- ESP32 microcontroller
- LoRa module: Microchip RN2483A (class C) [25]
- Sigfox module: Wisol SFM10R1 on a Snoc BRKs01 breakout board [26]
- Ublox NEO 7N GPS module [24]
- 1x Pushbutton
- 2x led
- 2x Antennas that support the unlicensed ISM band

The microcontroller is chosen based on the need for three hardware serials, while the other modules are basic modules for the LoRA/Sigfox connectivity and the GPS module has a very accurate precision.
Figure 14 shows a sketch of how the components are connected. This is just to get an initial idea of how the setup is going to be. This sketch is made in a program called Eagle. The ESP32 allows each of its 3 UARTs (Universal Asynchronous Receiver/Transceiver) to be freely initialised by using a digital pin between 0-32[23]. All three UARTs are used for LoRa, Sigfox and GPS. This means there is not a UART available for debugging.
In the design process there was thought about having a potentiometer that could adjust a timer to x seconds for every time a message should be sent. This would be useful for when driving a car or riding a bicycle so that one would not need to push a button every time.
The intention with the two led's are that one of them has to turn on when the device is powered. The other led should turn on when the button is pressed and when it is, the first led shall start blinking to indicate that the device is sending data.
When making the electrical circuit, the compactness of the design was not the purpose. It was made to give an idea of how the product could be wired.

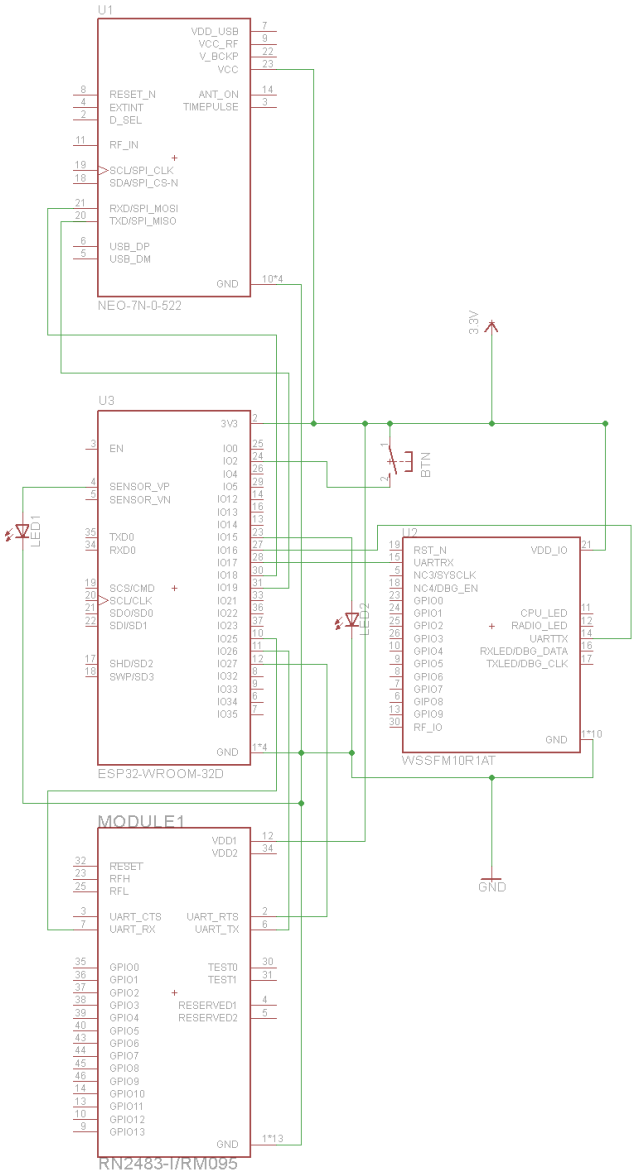


Figure 14: Electrical circuit diagram of the product design.

Figure 15 shows how the first prototype was wired. The sketch is drawn in a program called Fritzing. The antennas are not drawn but each module has an external antenna except the GPS module which has a internal and the capability to attach an external. Figure 16 shows the implementation of the design from the electrical circuit and the Fritzing sketch. It is very messy and not very convenient for testing outdoors. This is because the functionality of the product was the first priority. There was also an red led in the setup, which would turn on when the button was pressed to signal that it is ready to send and then the otherwise constant green light would start flashing.

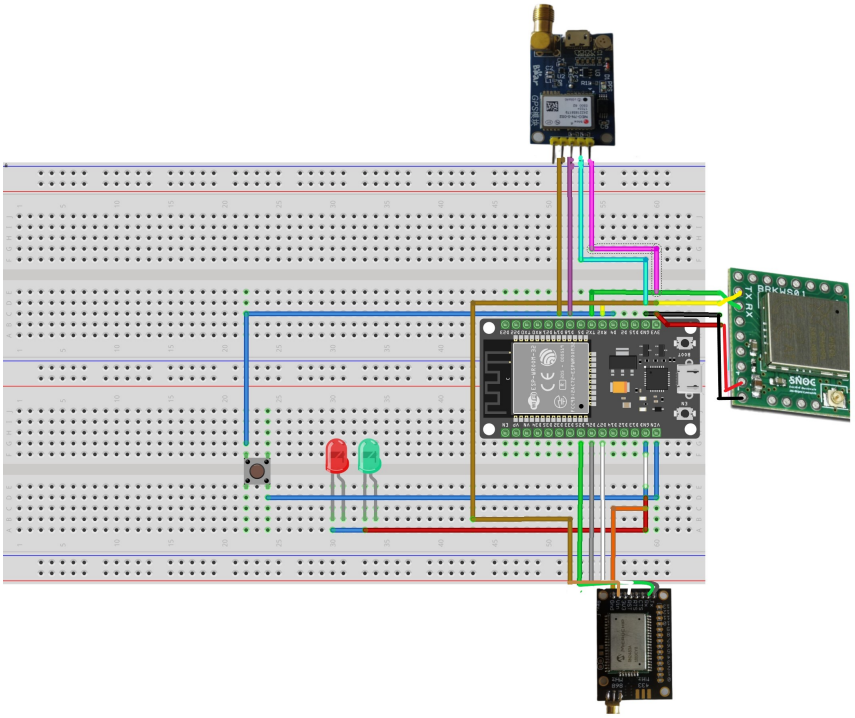


Figure 15: The product design on breadboard.

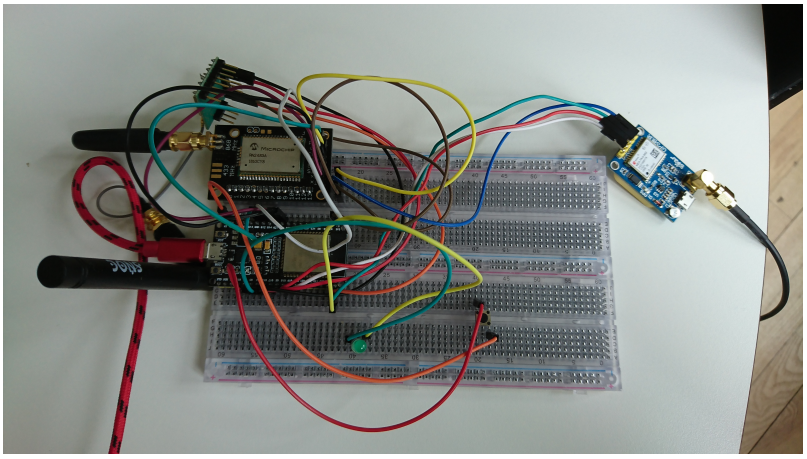


Figure 16: The product in the testing phase.

3.0.2 Solution

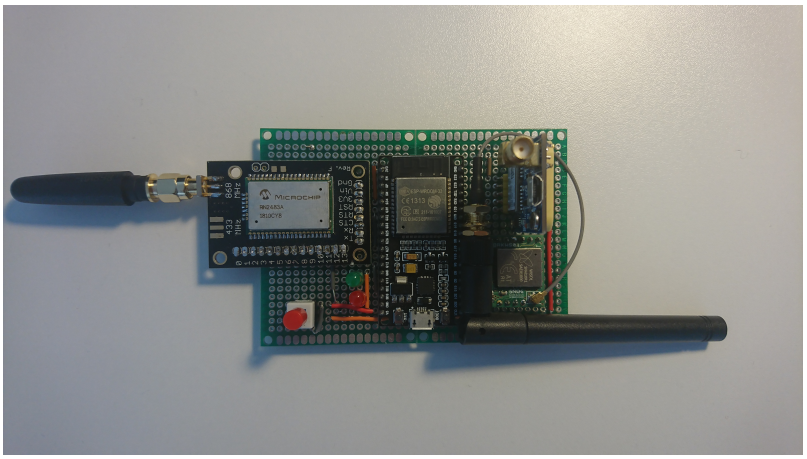


Figure 17: The finished product.

Figure 17 shows the finished product. When comparing it to the breadboard version it is more compact and neat. The external antenna for the Sigfox module could be glued or mounted in some other way, instead of being lose and dangling. The button has also been changed to one that keeps the state. This is chosen because of the lack of a potentiometer that could adjust a timer, which could make the device send data every x seconds without having to press the button every time a transmission is wished. The wiring is the same and more can be found in the "README" file. Looking besides the functionality and hardware, it would be nice with a case for the device. This would add robustness and more convenience when using the device. An optimisation for the device as it is, could be done by having one antenna that both Sigfox and LoRa shared. This would be possible because they both use the unlicensed ISM band but this would mean that they both would not be able to send data "simultaneously". By having one antenna the device could be more compact, preserve a bit of power, weigh and cost less. On the contrary Sigfox's ultra narrow band is resilient to interference with other signals, however LoRa's spread spectrum signal is affected by the ultra narrow band. As a precaution a little delay can be put between the two signals sending which could prevent the interference from happening.

3.0.3 Data flow

Now that a device is built it should send data to an online platform. As mentioned in the theory, Sigfox sends data to their backend server and LoRa sends data to Teracom's Nordic server. From the servers a callback can be made to some different online clouds. At first an online IoT cloud called Wia was used, but was quickly replaced by Microsoft Azure to get a more professional experience. There are a lot of services out there but in real life production one of the big firms solutions would be ideal. Amazon, Google IBM and Microsoft offer an online cloud service for IoT just to mention a few. They have different package solutions and offer different services in the cloud that suits different target groups from start ups to large scale productions.
The dataflow of the product can be seen on figure 18. It starts by the device receiving GPS coordinates from satellites. When received it sends the data via LoRa and Sigfox through

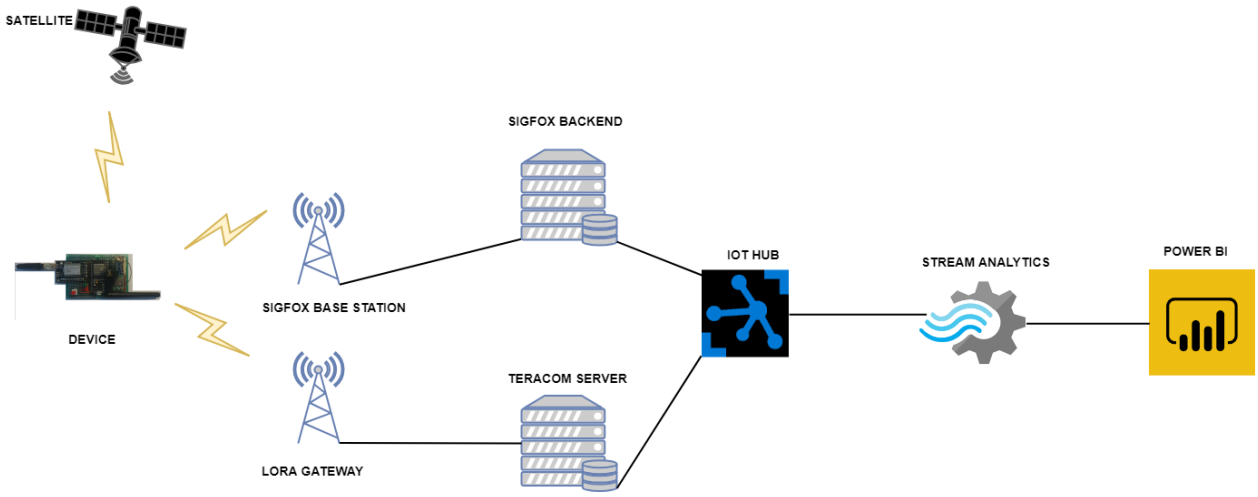


Figure 18: Dataflow of the project.

their gateway/base station to their backend server. From here a callback is made to Microsoft Azure, where there is created an IoT hub. In the hub the devices are registered, so that the data can be sent to it. The data is stored in a query. While this is all happening, a Microsoft application called Stream Analytics is running. Stream Analytics has to manually be activated in order to run it. This application fetches data from the IoT hub query in real time and outputs it to another Microsoft application called Power BI, where the data is stored in a data set. In this application the data can be visualised and given the data is coordinates, it is pinned on a Bing map. Power BI can be accessed from a browser or a mobile app which gives great flexibility to monitor or visualise one's sensor data. This flow could also have included a SQL database but is not chosen in order to reduce cost. Each application has their own billing plan and a customer can choose whichever fits their use. Microsoft Azure offers a big variety of applications, that can be connected to the IoT hub in order to fit any IoT use case a customer wishes. Most of the applications are also available as mobile apps which means a customer does not need to create their own and can rely on Microsoft's products.

4 Implementation

This section describes how to connect to LoRa & Sigfox network, the arduino code for the solution and how to connect from the backend servers to Microsoft Azure. It is assumed that a user for Teracom's server, Sigfox and Microsoft Azure has been created.

4.1 LoRa connection

To implement a device such as the one in this project, one has to start from one end. A simple setup with the LoRa module and ESP32 was made. It is the same wiring as for the end product and can be found in the "README" file. The first objective was to join the LoRa network. This was done without the GPS module. In order to join the network some commands had to be sent from the microcontroller to the module. The following commands were used:

- To get firmware version: sys get ver
- To get hardware ID: sys get hweui
- To set appeui: mac set appeui
- To set appkey: mac set appkey
- To set adaptive data rate: mac set adr
- To save settings: mac save
- To join via OTAA: mac set join otaa
- To transmit unconfirmed: mac tx uncnf
- To reset: sys factoryRESET

Each command is sent following backslash r and a backslash n. First of "sys get ver" was sent in order to check whether the connection between the two modules was correct. If the latest firmware, in this case "RN2483 1.0.3 Mar 22 2017" showed up, there was connection. Next step is to get the hweui by sending "sys get hweui". This is used to register the device on Teracom's server. With the following Teracom's server can be accessed: "https://iotnet.teracom.dk". Once a application is made on the server, a device can be enrolled[27]. It is important to choose "generate all parameters except DevEUI" when enrolling the device. The devEUI is manually entered with the hweui received from the module. To use the OTAA procedure to join the network, the appkey and appeui is retrieved from the server and sent as commands to the module. Now ADR can be activated in order to adjust the spreading factor as addressed in the theory. Then the settings are saved and finally the OTAA procedure can be initiated. If all goes well "accepted" is received and the device can start sending data. The data is sent as hexadecimal and can be seen on the Teracom server(only the latest 25 frames). By using this sending command it sends an unconfirmed message but there is a command for sending confirmed messages. It is also possible to receive data but this is not necessary for this project. The way of communicating with the module is through these commands and can be found in the datasheet[25].

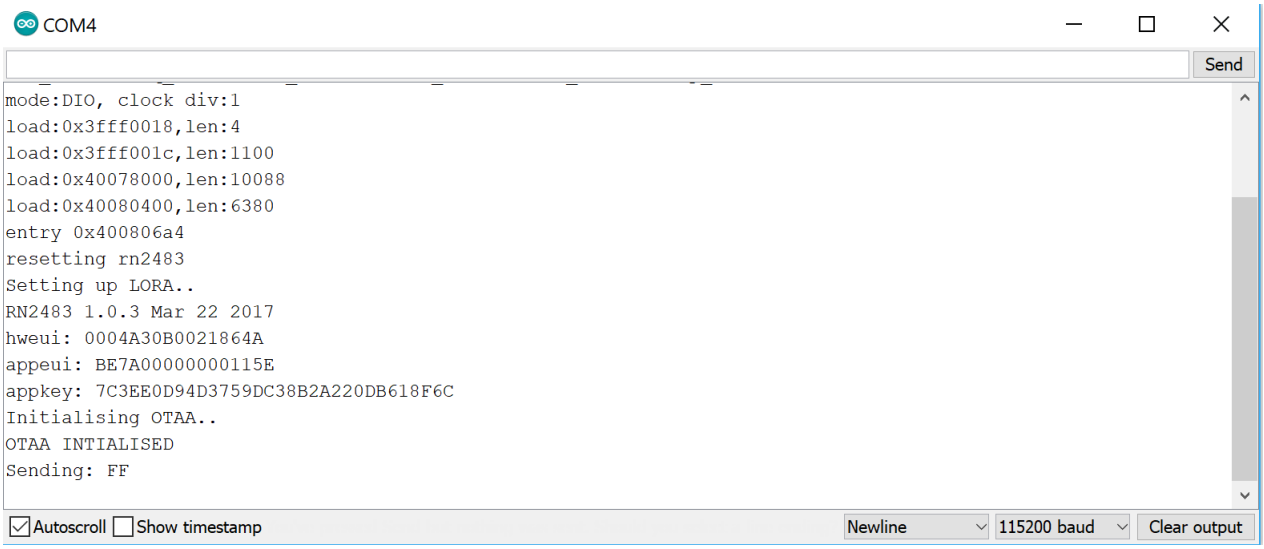


Figure 19: Arduino Serial monitor showing the different command responses.

Figure 19 shows the serial monitor from the arduino IDE with response from the LoRa module. The LoRa module actually responds for every command that is sent, but the serial monitor was only used for debugging, making sure there was connection to the module and to the network. The top part is just the ESP32 resetting. The sent data is a hardcoded hex string. This is because LoRa requires hex representation in order to sent a message.

Enrolment of the device is only done once, but before powering off the device it is a good idea to reset the device in case it becomes "bricked". It is best to reset the device before initialising the OTAA connection. Doing this every time can seem redundant. Therefore the appkey and appeui was hardcoded in the code, so every time the device was powered, the join procedure was only necessary.
A arduino library named "rn2xx3" made by JP Meijers was used instead for simplicity of the code but both ways work. This library includes a function to initiate the OTAA connection *"LORA.initOTAA(appeui, appkey)"* and returning true or false in case of connection or not. It also has other functions that can be useful but not used.

Now that the device could connect it was time to add the GPS module. By using the library "TinyGPS++" made by Mikal Hart. This library has two functions which allow the module to send latitude *"gps.location.lat()"* and longitude *"gps.location.lat()"* to the microcontroller. Further details about the GPS coordinates will be addressed in the section explaining the arduino code.

4.1.1 Sigfox connection

Instead of directly attaching the Sigfox module to the ESP32 with the LoRA and GPS modules, the Sigfox module was attached alone with the microcontroller, just like when testing the connection with LoRa. To register a device to the network, it must be done on Sigfox's website. Here the device type and PAC number is registered. Once it is done Sigfox will notify the user how many messages is allowed to be sent per day and for how long depending on the subscription. In this case 140 messages per day and for a year. This

is essentially all that is needed in order to use the Sigfox network. Like LoRA, Sigfox has some commands in order to communicate with the module. Sigfox use AT commands and the following have been used:

- To get the ID: AT$I=10
- To get the PAC number: AT$I=11
- To send a frame: AT$SF=

For testing the connection with the module the commands for ID and PAC number were sent. Like LoRa, Sigfox also has a lot of different commands[29]. If the ID and PAC is returned, there is connection as shown in figure 20. When the connection was established, the next step was connecting the GPS module. At first there was some issues with the built in function *Serial.print*, that truncated the decimals of the received coordinates from the GPS module when converting them to hex representation. For this reason two functions were made to manually convert the GPS coordinates into hex representation. This was done because the AT command that makes the Sigfox module send data requires a list of hex data. The sent data can be seen on Sigfox's backend https://backend.sigfox.com together with a timestamp and a link quality indicator.

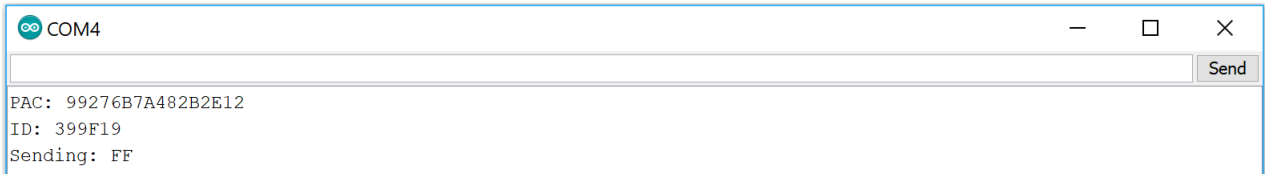


Figure 20: Arduino serial monitor showing the response from the module.

4.1.2 Cloud integration

When the verification of the two networks was done, it was time to connect all the modules to the microcontroller. Once this was done, the data had to be sent to an IoT platform in order to visualise the data. At first an online IoT platform named Wia was used: https://www.wia.io/. Wia is simplistic and a good option when being new to the IoT platforms. With Wia the basic understanding of how the Jason object was created was learnt, but to have a more secure and professional platform, Microsoft Azure was chosen. The data is sent from the Sigfox/LoRa base station/gateway via a callback made from the servers to Microsoft Azure. It is assumed that an account for Azure has been created. Here is a step by step guide for how to create a callback to Microsoft azure, starting with Sigfox.

1. Clicking the Device Type tab shows all the registered devices. Then clicking the name of the device following "Callbacks" and "new" to create it as seen on figure 21.

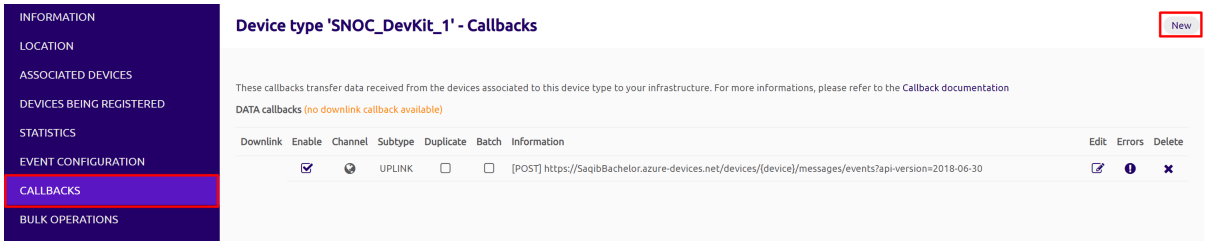


Figure 21: Step 1: Creating a callback.

2. A list of different callback types appear. Here Microsoft azure is selected. Now the callback configuration is created as seen on figure 22. Here in this case the device wants to send uplink data to Azure. This is done by retrieving the connection string from Azure. Information on how to get the connection string can be found by clicking on the question mark. As seen on the figure the JSON body includes "data", which is the data sent by the device. This JSON also includes latitude and longitude from Sigfox's Atlas, so it can be compared in Azure.

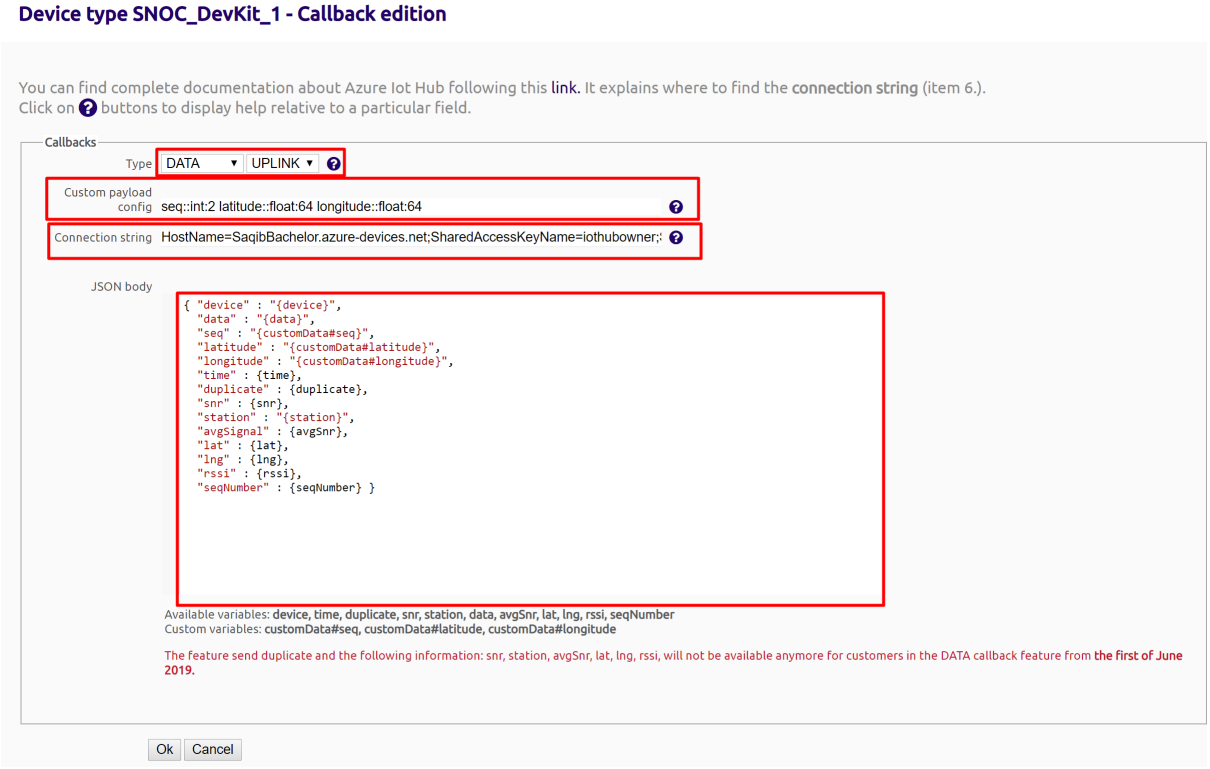


Figure 22: Step 2: Configuring the callback with the type, custom payload, connection string and the JSON body.

Now the callback should be created. As seen on figure 21 a callback is already created. It has to be ticked to be enabled. This means the next time data i sent via Sigfox, it will be sent through a callback to Azure. Before explaining what happens in Azure, a little instruction on how to create a callback on Teracoms server is described:

On Teracoms server, the tab called "output" under the application is clicked. Here it is possible to choose where and how one's data is sent. By clicking on the "add new output" button as seen on figure 23, Microsoft Azure IoT Hub can be created. The callback is created by adding the primary key from Azure's IoT Hub and naming it the same as in Azure. To find out how to retrieve the primary key, Teracom have made a little guide on how to do so on the same page.

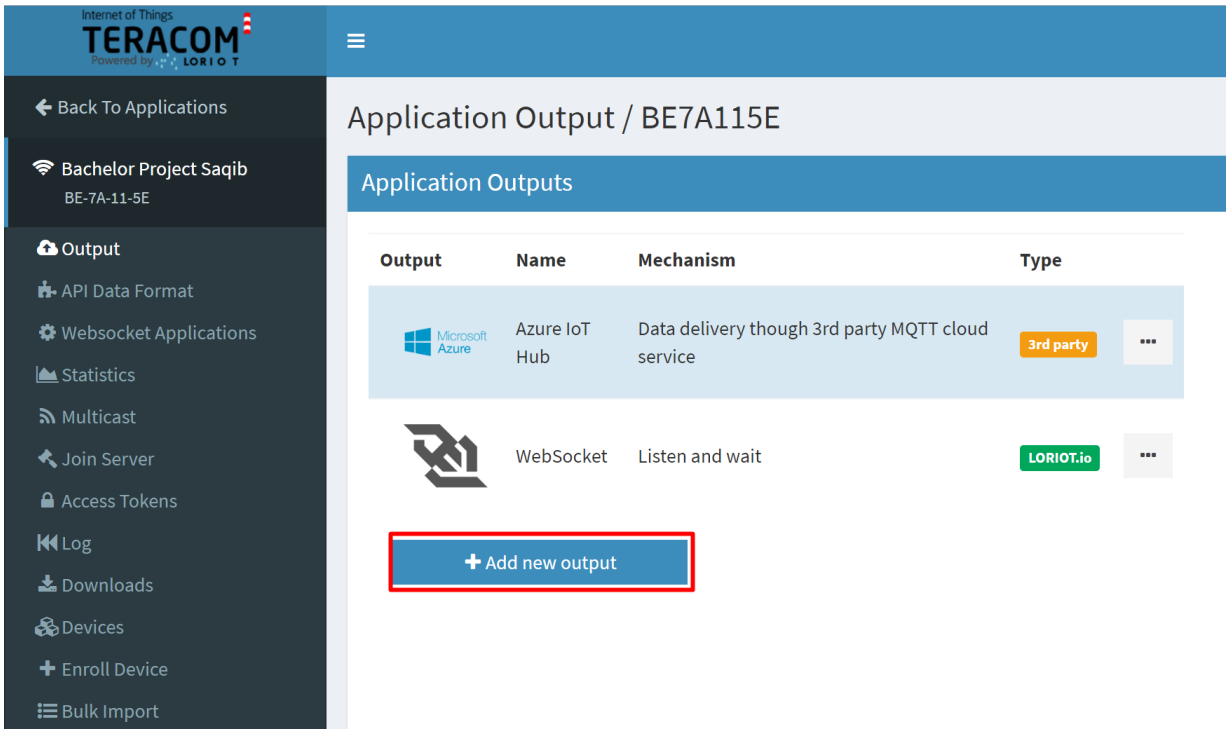


Figure 23: Creating a new output on Teracoms server.

In order to create an IoT Hub, a resource group is created first. This group contains all the different applications used for a project. Figure 24 shows the home page of the Azure portal. When creating a resource or an application a billing method is chosen. This project is running on the free trial that Microsoft offers, but when that is used up it runs on "pay as you go". After creating a resource, the main application IoT hub is created.

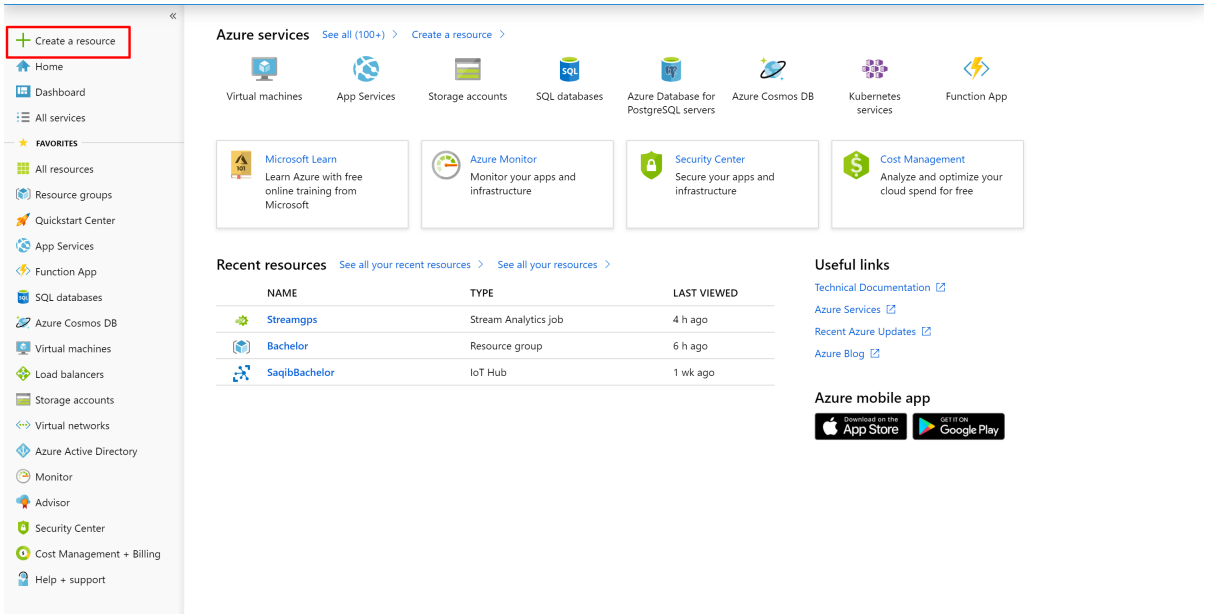


Figure 24: Azure portal homepage

When the IoT hub is created, the application is launched as seen on figure 25 and is created to the resource of a customers choice. It is possible to have multiple IoT hubs but in different resource groups. The possibilities with IoT hub is big, but only the functionalities for the implementation of this product will be addressed. The overview of the IoT shows different telemetry information, which can be useful to monitor the amount of that being received. To get access from any device to the IoT hub whether its via a callback or straight from a device, the keys can be found under "Shared access policies". The servers for Sigfox and LoRa created a callback in two slightly different ways. Sigfox used the connection string while LoRa was created with a primary key. This means that the LoRa capable device has to be added, which can be done under "IoT Devices". It is important to name the LoRa device the same as the hweui. If the Sigfox step to step guide was followed correctly, the Sigfox capable device should already be listed there.

The next step is to store and visualise the data. IoT hub only keeps the data for a certain amount of retention time set in days. The application "Stream Analytics" is created. This application takes the data as a "real time" stream from IoT Hub as a input. It can chose between different applications as input and output. Figure 26 shows the overview of the Stream Analytics job, which also shows monitoring of the data coming in to the application as input and the data it sends to the output. By clicking on input under "job topology", the IoT hub can be set as input. When this is done a output has to be chosen. For this project Microsoft's Power BI application has been chosen. Before choosing Power BI as an output, an account has to be created on https://powerbi.microsoft.com. Power BI is a great tool analyse business data and visualise it. Once the account has been created, then Power BI can be put as a output. Now the flow essentially is created as seen on figure 18 but there is still a little to do. The input Stream Analytics receives is put in a query. This allows the customer to filter the data.
Figure 27 shows the SQL statements used for this project. The selected variables are the ones from the JSON objects sent from the two servers via callbacks. UDF is a user defined function

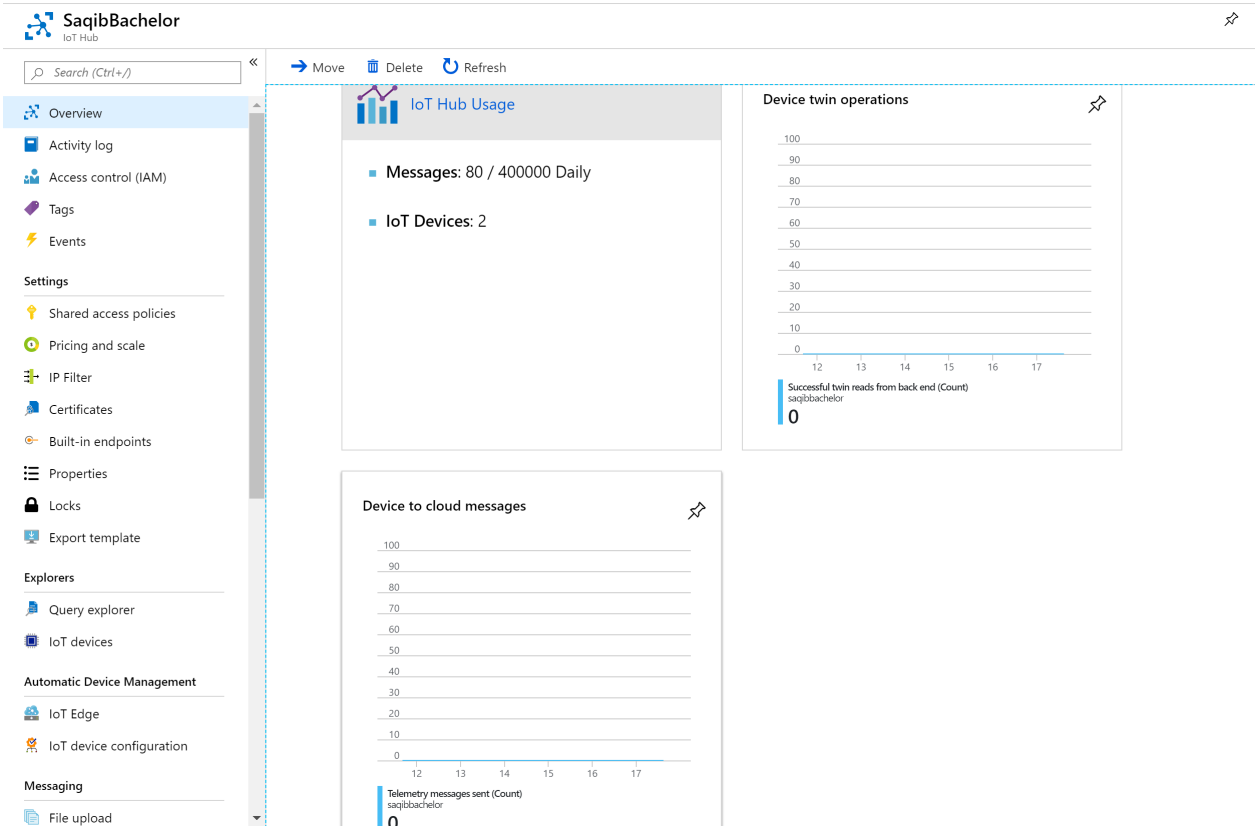


Figure 25: The IoT hub

which can be made in Stream Analytics. For this project the function "hexToAscii" is made as seen on figure 28. It takes a string as an input, in this case a substring (built in function) of the data variable, which is the one containing GPS coordinates in hex representation from the microcontroller and converts it to ASCII. The substring of data can be used because the length of data is predetermined. In this case the coordinates will have three decimal points. The UDF function in Azure can only be programmed in JavaScript.

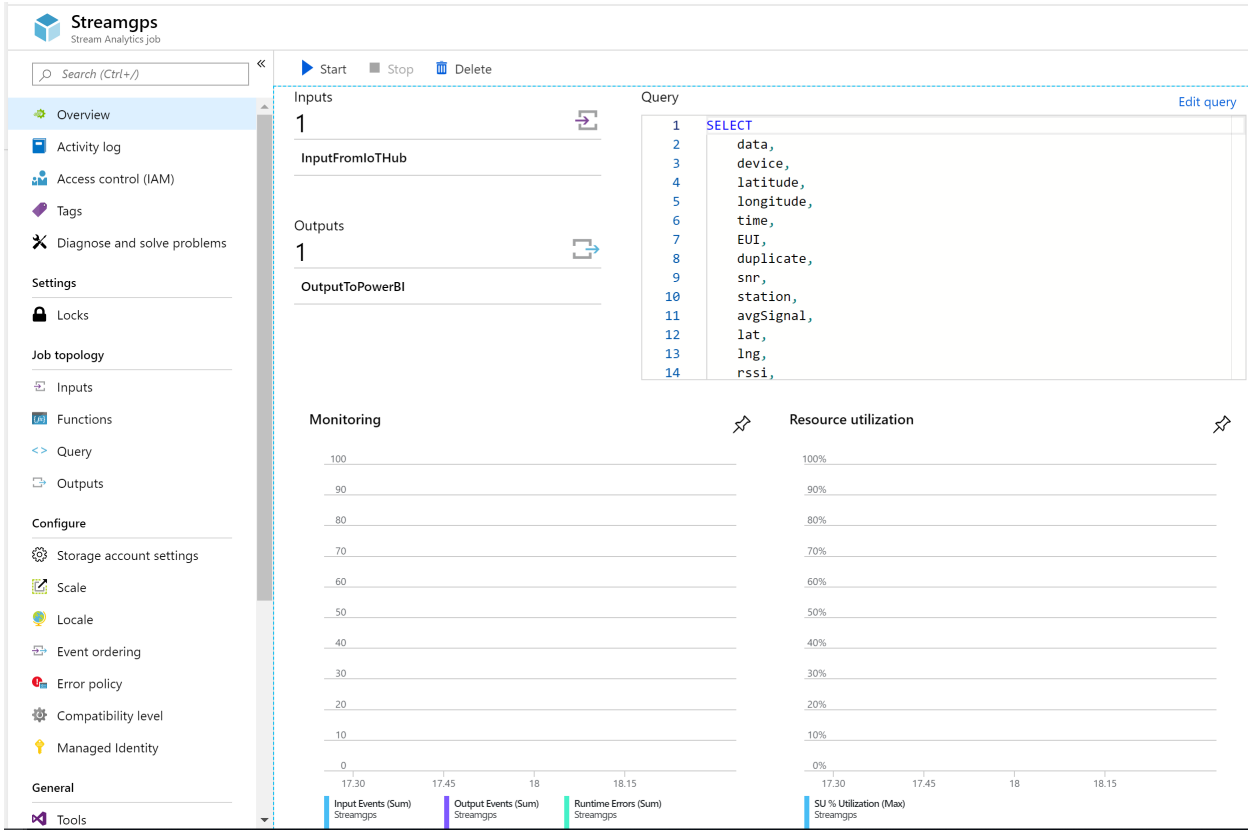


Figure 26: Stream Analytics.

The output of the query sends it to Power BI where the data is stored in a data set and table. In Power BI all sorts of visualisation can be made from tables to charts. In this case a map to pin the latitude and longitude. This will be addressed further in the "Field tests" section. In Power BI data can be visualised in real time, reports can be made and a dashboard with different visualisation windows. The accessibility Microsoft offers is also great. The data can namely be visualised in a browser but also on Power BI's mobile app. Now the that the cloud integration has been explained, it is worth looking at some differences between the callbacks made from Teracom and Sigfox's servers. Figure 29 shows the JSON body which Teracom sends to Azure. This can not be changed which sets some limitations, whereas with Sigfox as seen earlier, it is possible to have custom data in the JSON body. Therefore in terms of creating a callback to Azure, there are more possibilities with Sigfox. This says more about Teracom than LoRa as a technology. This means that data sent via LoRa and Sigfox will be set to the variable "data", which explains why there only is one "data" variable in the query. The variable could be changed in the Sigfox JSON body but there is no need it to do so. This is because in Power BI the values can be differentiated by the device name when mapping the coordinates.

4.1.3 The Arduino Code/Functionality for the End Product

There is not any advanced function for the arduino code. Turning the product on it initialises all the pins, variables, serials etc. As mentioned when setting up LoRa it is a good idea to reset the module before using it which is also done. In the setup the device tries to connect to the LoRa network via OTAA. When the device is ready a green LED turns on. The LEDs

```
SELECT
    data,
    device,
    latitude,
    longitude,
    time,
    EUI,
    duplicate,
    snr,
    station,
    avgSignal,
    lat,
    lng,
    rssi,
    seqNumber,
    UDF.hexToAscii(subString(data,1,12)) AS intLat,
    UDF.hexToAscii(subString(data,13,24)) AS intLong
INTO
    [OutputToPowerBI]
FROM
    [InputFromIoTHub]
```

Figure 27: SQL statements for the stream Analytics query.

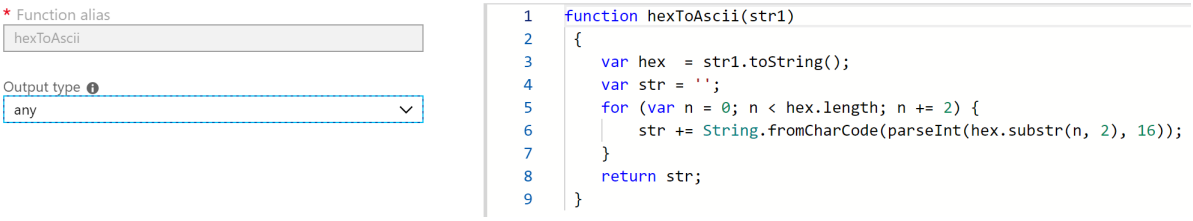


Figure 28: JavaScript function that converts hex data to ASCII.

in general are used to signal states and compensates for the lack of a Serial monitor when testing. In the main loop a while loop is implemented. The while loop checks to see if the serial to the GPS module is available, if so, it will encode the sensor data so that one char comes from the GPS module at a time and finally if the GPS location is valid (boolean), it will go into the main functionality. Here it will start fetching data(latitude/longitude) from the GPS module. As mentioned earlier the "Tinygps++" library is used to utilise these GPS functionalities. The library outputs the latitude and longitude in double. This is casted as a String in order to concatenate the coordinate points. With LoRa the concatenated string is sent as by the send command from the rn2xx3 library. Sigfox on the other hand is more tricky. The concatenated string is converted into a char array which is sent by a custom sendData function. This function essentially sends the AT command to send a frame following a while loop that sends one char that is converted manually to hex at a time. This is not a very optimal solution and is chosen because of some issues with the standard print function in the Arduino IDE. In the setup another solution with unions and structures were created, which can be seen in the attached files.

The main functionality is sending GPS coordinates when a button is pressed. When the button is pressed the red LED turns. This indicates that the device is ready to send. A delay is hardcoded to 1-2 min depending on the test it has been used for. This is not very optimal and would have been better with a timer and perhaps a potentiometer to adjust this timer. When the device is transmitting the red light stays on while the green light starts

```
{
  cmd      : 'rx';    // identifies type of message, rx = uplink message
  EUI      : string;  // device EUI, 16 hex digits (without dashes)
  ts       : number;  // server timestamp as number (seconds from Linux epoch)
  ack      : boolean; // NEW! acknowledgement flag as set by device
  fcnt     : number;  // frame counter, a 32-bit number
  port     : number;  // port as sent by the end device

  encdata? : string;  // data payload (APPSKEY encrypted hex string)
                      // only present if APPSKEY is not assigned to device

  data?    : string;  // data payload (decrypted, plaintext hex string)
                      // only present if APPSKEY is assigned to device
}
```

Figure 29: JSON body of LoRa message sent from Teracom's server to Azure.

flashing. After the device has transmitted, the red LED turns off.

The most troubling parts of the implementation was utilising Sigfox's 12 bytes payload optimally with latitude and longitude. This was very time consuming because the method used in the final code uses all 12 bytes on the coordinates which is not very efficient. It is not a problem with LoRa because of the bigger payload, but the data is kept the same to have a fair comparison. By using the whole payload on coordinates it excludes the possibility to utilise some bytes for other information such as a frame number, battery percentage etc. By using a structure or a union some bytes can be spared for other purposes as seen in the Sigfox code.
Besides this, another time consuming factor was getting to understand Azure. How the cloud functioned, how to connect, what to connect and how the different applications worked together. This was solved by trying out the different applications which can be costly despite the free trial. This is normal when developing a product, then the cost always is high when starting.
The step from the breadboard version to the compact design on the pcb was also troubling. The modules had to be arranged in a as compact way as possible, which meant some of the pins had to be changed and led to the not so ecstatically pleasing appearance.

5 Field tests

This section will address the various tests made with the product. It will analyse and discuss the results.

5.1 Verification

As described the technologies have individually been tested in order to verify that they work before the product can be assembled. To verify and debug each module, the Arduino IDE's serial monitor was used as mentioned in the implementation. In terms of LoRa and Sigfox this is not enough so every time a message was sent, the backend servers would be checked to see if the message was received successfully. To verify that the whole solution works, the different parts of the data flow has to be observed. If the GPS module blinks, it means that it is receiving data, which has been printed in the test phase to see if the data is received correctly. When the device is powered up and starting to receive coordinates, the green LED should turn on. It will keep on staying green unless the button is pressed. If the button is pressed, the red led will turn on to indicate that the device is ready to send data. When sending data, the green led starts to blink and when it is done it turns off. The button saves the state so after a hardcodded amount of time, the device will start sending again if the button is not pressed. If pressed again the red light turns off and the green stays constant. To verify whether the data is sent via LoRa and Sigfox, the backend servers are checked to see if the data is received successfully. All three UARTs on the microcontroller are in use, therefore it is not possible to use the serial monitor. If the message is received successful at the backend, the next step is to check whether the IoT Hub has received the message. Recalling the cloud integration section, the overview of the IoT Hub has an telemetry counter which counts every message that is delivered to the cloud. To check which technology is received in Azure, the Stream Analytics job has to be checked next. Here it is possible to see how many messages are received from the IoT hub and how many are sent as an output to Power BI, where it is possible to look into a table to see what data has been received and from which technology. This can seem a bit cumbersome in order to check whether a message has been lost in transmission, but this is the only way for this particular solution.

5.2 Tests

Now that the device has been explained and verified, it is time to test it in the field. The following tests have been made:

- Testing the device in a car around campus.
- Testing the device walking around campus.

These tests will check the device's and the technologies ability to transmit under different speeds. These two tests will look at the packet loss and evaluate which technology is better for mobile devices.

Before the actual tests, a test with only Sigfox was performed. This test was to ensure that the data was received in Power BI and it gave some interesting results. The test was performed by driving around in Herlev (DK) and sending data every 2. minute. Figure 30 shows the results from the test and the route.

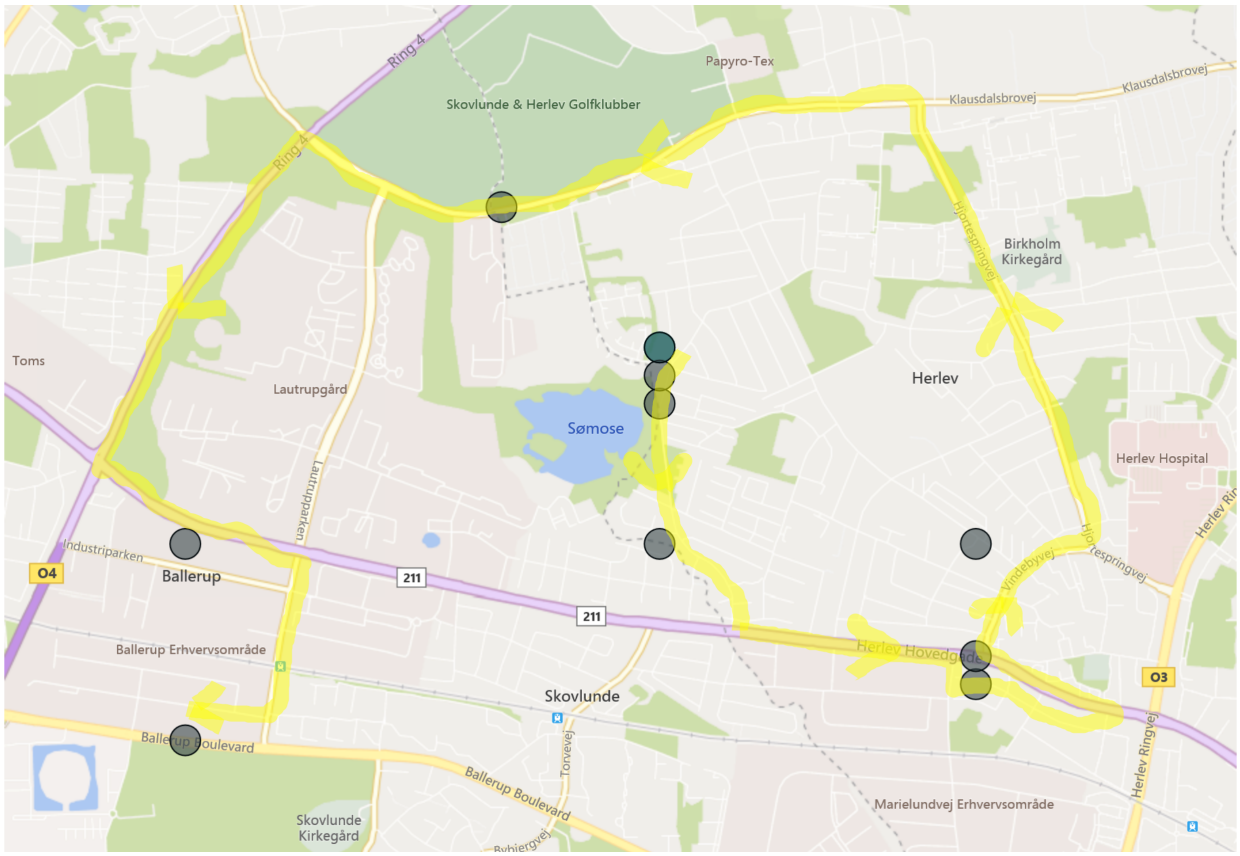


Figure 30: Various driving speeds around Herlev showing GPS coordinates sent via Sigfox.

This test was mainly done to see data visualised in Power BI but noticeably Sigfox was bad too almost not sending when driving at 60+ km/h. This route took approximately 30 minutes to drive and as seen on the map, there is only ten data points. This means five data points were lost in this process. Without further data it is difficult to argue for what the reasoning behind this result is. According to [28] LoRa allows more reliable communication over high speeds than Sigfox. This claim will be tested to see if there's any truth behind it.

5.2.1 Test 1: Testing the device in a car around campus

The device for this test was set to send data every minute, if there was data available and the button was in a HIGH state. If there was not any GPS data available, it would simply not send any data. The reason for this is to use as little data as possible. The way to check whether a message was sent via LoRa and Sigfox would be to check their backend. If the backend had received a message every minute and the sequence number on the backend incremented correctly, it meant that the message was received correctly at the backend. This data would then be compared with the one received in Azure, to see how many messages got lost in the real time stream.

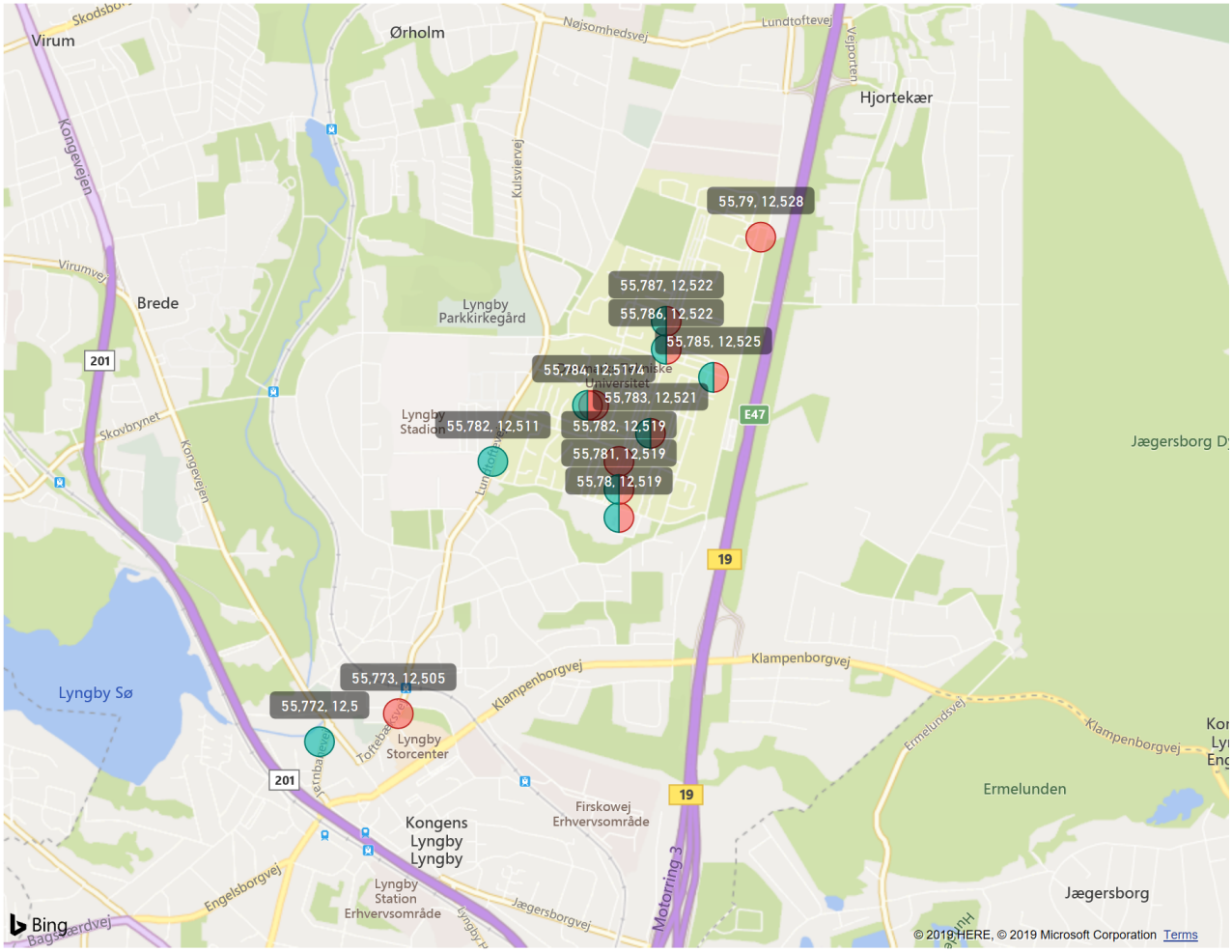


Figure 31: Test 1 showing the data points from the drive around campus and Lyngby.

Figure 31 shows the coordinates sent while driving. Three decimal points were chosen which a bit surprisingly gave a precision range of 15-20 meters. This shows the device's capability to give an fairly accurate location. The route took 45 minutes and started at building 343 at DTU. Here it was possible due to the less amount of traffic to drive between 20-30 km/h. As seen it has given some good results for the cars whereabouts. The red dots indicate a LoRa coordinate while the turquoise represents a Sigfox coordinate. It is seen that some coordinates only have one of the technologies representing. Appendix A [37] shows the received data at the two backends. Comparing the two it can be seen that Sigfox gives a bit more information. Teracom only shows the last 25 received frames while sigfox shows

all the received messages on their overview. It is possible to download a A JSON file or csv file with all the data received at the backends. Clicking on the callback button at the sigfox button, the sent JSON object can be seen with all the variables.

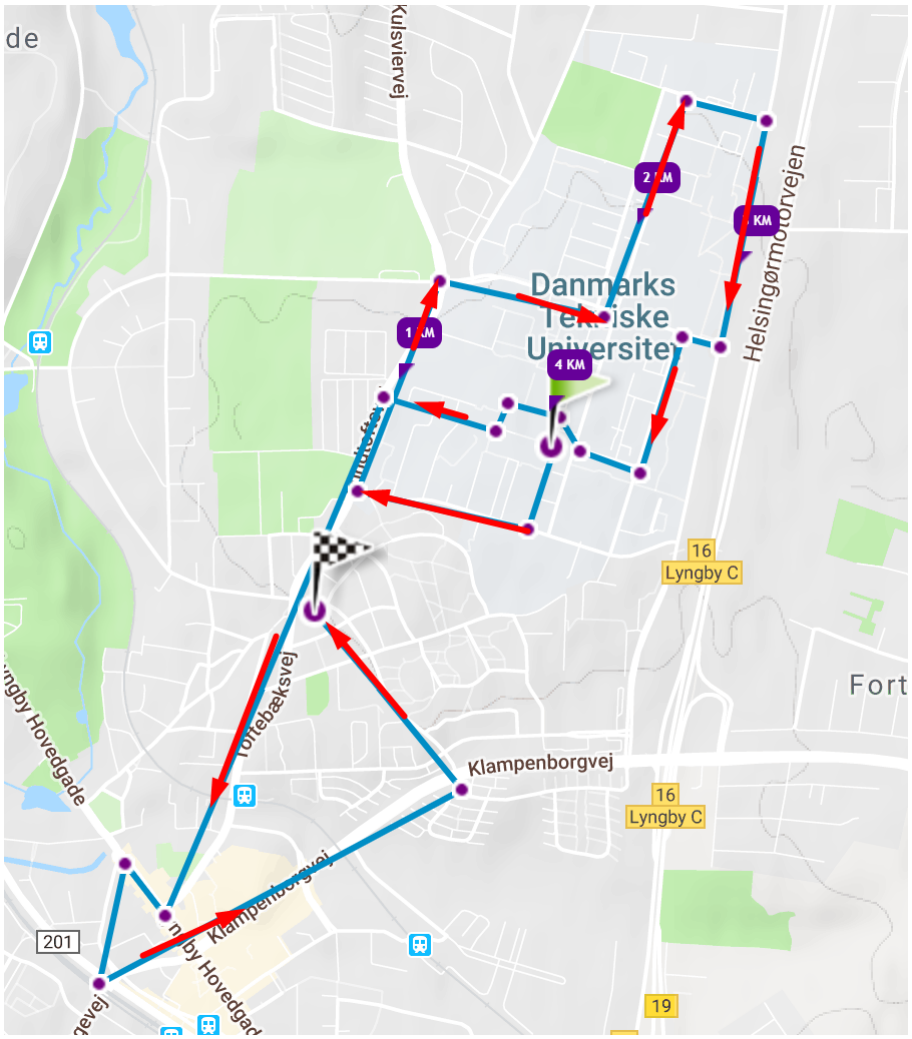


Figure 32: The actual car route.

Comparing figure 31 with figure 32 it can be seen that the coordinates are fairly accurate but it is noticeable to see that on "Klampenborvej" and in general further away from campus there were only two data points. Even tough only 12 locations can be seen on the map, looking at the data received in Power BI, it can be seen that there are multiple data with the same latitude and longitude. Figure 33 shows a table from Power BI with the received. There are a lot of more values because some of them are from other days as the time would indicate. The table produced in Power BI is cumbersome because the time is given in unix epoch which gives a time stamp in seconds starting from 1970. This is not really useful and could have been optimised by making a time converter in the Stream Analytics application. It is also notes that there is no timestamp for LoRa which is the blank device in the table. The variables intlat and intlong are latitude and longitude. These variable names should be changed to a more suitable name. Recalling from the implementation these variables are defined in the Stream Analytics job when splitting the received payload into two variables by the custom function. The device name is received from Sigfox and not from LoRa. This

is because LoRa calls the device name for EUI and since a user can not change the variables of the JSON body on Teracom's server, it could be changed in Sigfox's callback in order to display the device names under the same variable. Based on the results in Power BI there can not be given a conclusive answer. Looking back at the data in the appendix, it can be seen that only 12 out of 45 Sigfox possible messages were sent in this test, while 16 out of 45 possible LoRa messages were sent. Looking at the sequence number on Teracoms server it is seen that packet number 18 is lost during the transmission from the device to the server. The sequence number starts at 6 because the first couple of messages were used to test the device. The reason for this could be the speed of the car where the Doppler effect could have been an influence. Otherwise it can be mentioned that the dangling antenna of Sigfox could have accrued some interference. As figure 31 shows there are only 12 locations received at Power BI where 9/12 are via Sigfox and 10/12 are from LoRa. Based on these results LoRa seems the better option technology wise when it comes to mobile devices. Conclusively it is not a big enough data set to conclude on the overall technologies but based on this small test it is safe to say LoRa seems more reliable. As mentioned in the theory Sigfox can send latitude and longitude with their Atlas geolocation. This is sent every time a message is sent. Figure 39 shows the two coordinates that were received when sending data via Sigfox. With one of the coordinates being in Sweden and the other one northwest of Zealand, it is safe to say that they are far off. This was already covered in the theory, but imagining it was used as an instance for tracking an international parcel, it seems accurate enough to determine the parcels whereabouts.

intlat	intlong	device	time
55.772	12.500	399F19	1558344000
55.773	12.505		
55.780	12.519		
55.780	12.519	399F19	1558343542
55.781	12.519		
55.781	12.519	399F19	1558341672
55.781	12.519	399F19	1558341744
55.781	12.519	399F19	1558341816
55.781	12.519	399F19	1558341888
55.781	12.519	399F19	1558341960
55.781	12.519	399F19	1558342032
55.781	12.519	399F19	1558343464
55.782	12.511	399F19	1558344300
55.782	12.519		
55.783	12.521		
55.783	12.521	399F19	1558343164
55.783	12.521	399F19	1558343242
55.784	12.517		
55.784	12.517	399F19	1557492428
55.784	12.517	399F19	1557499233
55.784	12.517	399F19	1557499493
55.784	12.5174		
55.785	12.525		
55.785	12.525	399F19	1558343016
55.785	12.525	399F19	1558343088
55.786	12.522		
55.786	12.522	399F19	1558342405
55.787	12.522		
55.787	12.522	399F19	1558342486
55.790	12.528		

Figure 33: Received data in Power BI from test 1.

5.2.2 Test 2: Testing the device walking around campus

This test is based on the same terms as the first one. It sends every minute and the purpose of this test is to walk around campus with the device to see if speed has made any difference in terms of the amount of packets received between the two technologies.
Figure 34 shows the route and the results from the test. The route starts from the red dot and follows the yellow path. It is seen again that the received location is fairly accurate, some places more than other. The route took 30 minutes to complete which means 30 possible messages could have been sent. Looking at appendix C [40] 22 out of 30 messages were sent via Sigfox, while 13 out of 30 messages were sent. There is a considerable bigger amount of messages sent via Sigfox than LoRa. This could be due to the fact that Sigfox's UNB interferes with LoRa's CSS. From the results in the visualisation it is seen that only 9 locations are received. Of those 9 locations Sigfox managed to transmit all 9 times, while LoRa only manages to transmit 6 times. While performing the test the activity in the Stream Analytics application was being monitored from a mobile phone as seen on figure 35. 44 messages are received which is more than the combined total of the messages sent via LoRa

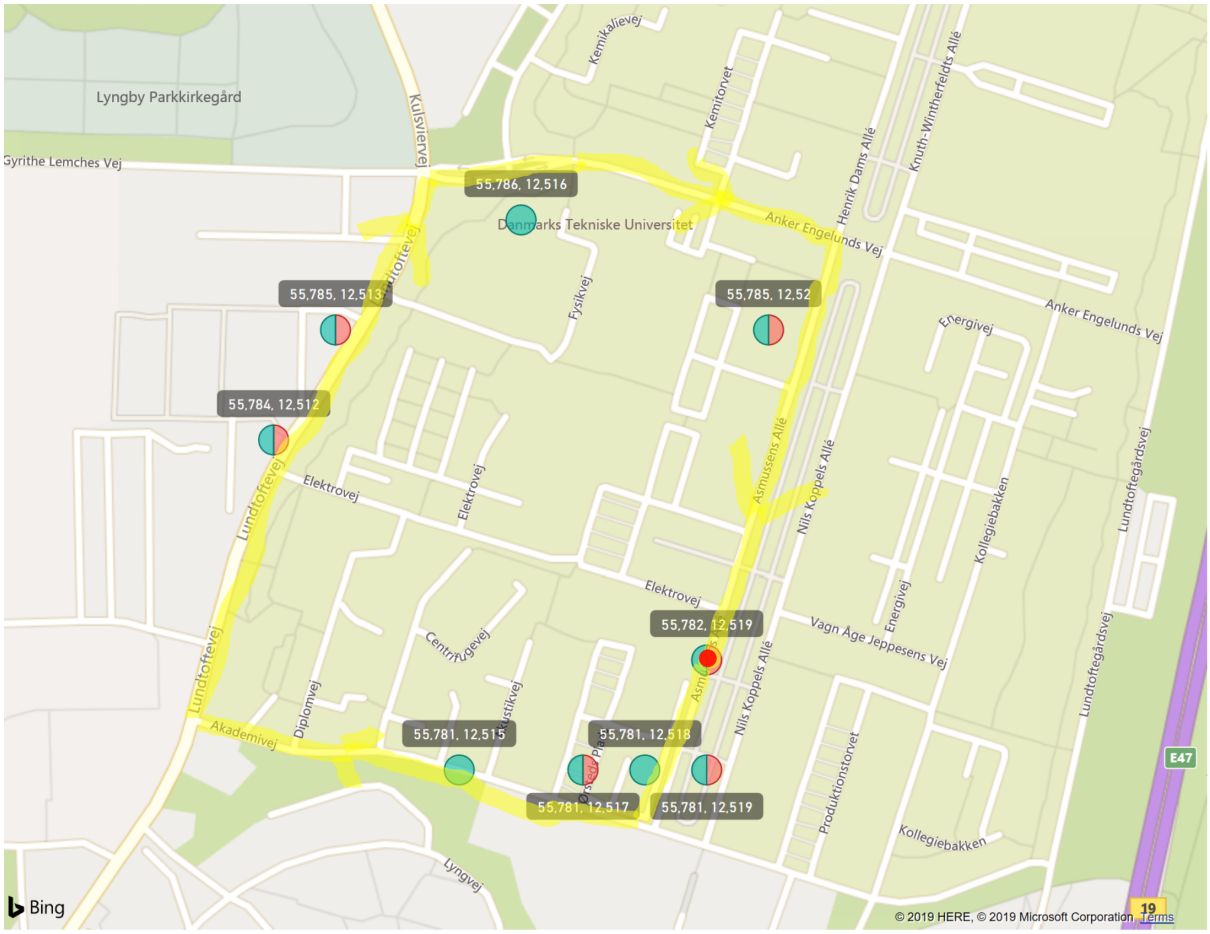


Figure 34: Test 2 showing the data points from the walk around campus.

and Sigfox. This is due to some small tests before the actual test in order to make sure the device was functioning properly before walking. 37 messages were sent as an output to Power BI, which means 7 messages wast lost in the Stream Analytics application. Out of these 37 messages there are a lot of values that were the same, this is because the device had not moved enough given the sent GPS precision.

Figure 35: Overview of Stream Analytics from a phone browser.

Based on these results it is seen that most packets via Sigfox was received at the backend and Power BI, which would make it the most reliable technology when walking. This could be due to Sigfox interfering with LoRa when sending. A larger delay between the transmission could prevent this to some degree. Table 2 summarises the results received in Power BI. It is seen that Sigfox had a 100 % success rate when walking, while LoRa only had 66.66 %. In the driving test LoRa was the more reliable technology with 83.33 % success rate, while Sigfox has a 75 %.

Walking Test	No. of packets sent	Total no. of packets sent	Successrate %
LoRa	6	9	66.66 %
Sigfox	9	9	100 %
Driving Test	**No. of packets sent**	**Total no. of packets sent**	**Successrate %**
LoRa	10	12	83.33 %
Sigfox	9	12	75 %

Table 2: This table summarises the results from the two tests.

6 Future work & Discussion

This section discuses the product and the possible future improvements. Some points have been mentioned throughout the report, but this section will summarise the most important needs for this device. Appendix C [42] shows how the time was spent for the project.

6.1 Hardware

The microcontroller for the product is ESP32. The reason for this was primarily the three UARTs it offers and the small design but more or less any microcontroller with three or more UARTs could be used. For future work NB-IoT could be included. Here it would be better to have four UARTs but it would be possible to keep the ESP32. The downside would be needing to switch the serial between two technologies. As mentioned earlier it could be useful with a potentiometer to adjust the time based on a timer for when to send a message. In this project the device is being used for research about comparing Sigfox and LoRa but it has potential for other uses. Only the imagination really sets the boundary. As seen from the test it could be a useful as a tracking device, maybe not the cheapest tracker if NB-IoT was included but the capability is there. Some cost could be saved by using either of the technologies own geolocation techniques which would be optimal. The downside of replacing the GPS module would remove the possibility to track in real time. By adding different sensors this device could become a multifunctional device with the capability to use different technologies. In someway like a smart phone, here it is just a "smart device". This would make the device a bit pricey and above the label of "cheap" which does not correspond with the requirement of IoT. In reality there is no point in having three technologies to determine a location. It just makes the device more expensive. The main purpose of this device could be for firms, that want to test which technology is best in their area and for their use case. It makes an ideal testing device when all three technologies are added. By having a device with all different kinds of sensors with the technologies would allow it to be used for a broader range of firms to use as a tool to determine which device would suit their needs. Based on the two technologies that have been tested it is hard to conclude which is best in Lyngby, more tests are needed and other factors such as RSSI, SNR etc. shall be evaluated. With this device it is possible though.
Looking at the device as it is, it can be hard to monitor what is going on. Therefore an addition of an LCD screen in order to get some information about, what the device is doing would be nice but also consume more power. Mentioning power it would also be ideal to make the device battery powered. This would make the testing process much easier instead of carrying a power bank or some other type of external power around.

6.2 Software

In terms of software on the microcontroller, it would be optimal to use timers instead of delay. The current code in general needs optimisation to make it neater and more efficient. To prevent interference, each technology could send if a response is received from given a timer, if no response, then it would start sending with the next technology. Error correction could be implemented to prevent errors. The use of Sigfox's payload has to be more efficient, so that other information such as sequence number, battery percentage and temperature can be added. Otherwise there is not much else beside security.
The SQL in the query could be improved and more variables from the sent JSON objects could be added and sent to Power BI. The Stream Analytics job can tend to crash if collision is occurred for instance when the variable "data" is received. It is the same name for the data sent via Sigfox and LoRa. A function for time conversion should also be made, so that a more common time format can be seen when visualising the data. With the visualisation in Power BI there a different possibilities to visualise and various kinds of information. This gives good possibilities when wanting to visualise different sensor data. A dashboard can be created where different visualisations can be shown in windows and adjusted to a users preference. Figure 36 shows the dashboard for this project on a mobile phone using Power BI's app. In terms of the cloud choice in general Microsoft offers a lot of different applications and capabilities with Azure. There are different billing options but other options can more suitable depending on a users usage and demands. The good thing about Microsoft is its various applications that easily can be integrated and launched based on actions and events.

6.3 Tests

In the future more tests have to be made to compare the technologies in more depth. By using Power BI there are some limitations when it comes to the filtering possibilities of the data. Therefore it might be a good idea to store the data in a SQL database and from there send the data to Power BI. This would mean that the data would not be visualised in real time, but then unnecessary data does not get transferred to Power BI. These tests that have been performed also sent unnecessary "test" data to Power BI which can cumbersome to remove. Especially when Power BI does not allow a user to edit the data set. The data set can be downloaded and manually changed. To download and change the set a user must upgrade their Power BI to a PRO version. To visualise data in Power Bi it is also important to look into what data type/format a given visualisation requires. As an example in order to map these coordinates, it has to be a String. It will not allow any other type.
New and better tests have to be performed in order to get more significant data that can be analysed and evaluated. The device will not be able to test every aspect of the technologies, but in terms of coverage, RSSI, packet loss etc. it would be good to test this over a larger area. RSSI and SNR as example were received in the performed tests but because of Powwer BI's limited filtering capabilities and lack of consistency in terms of the sent data and JSON body from the Teracom server.

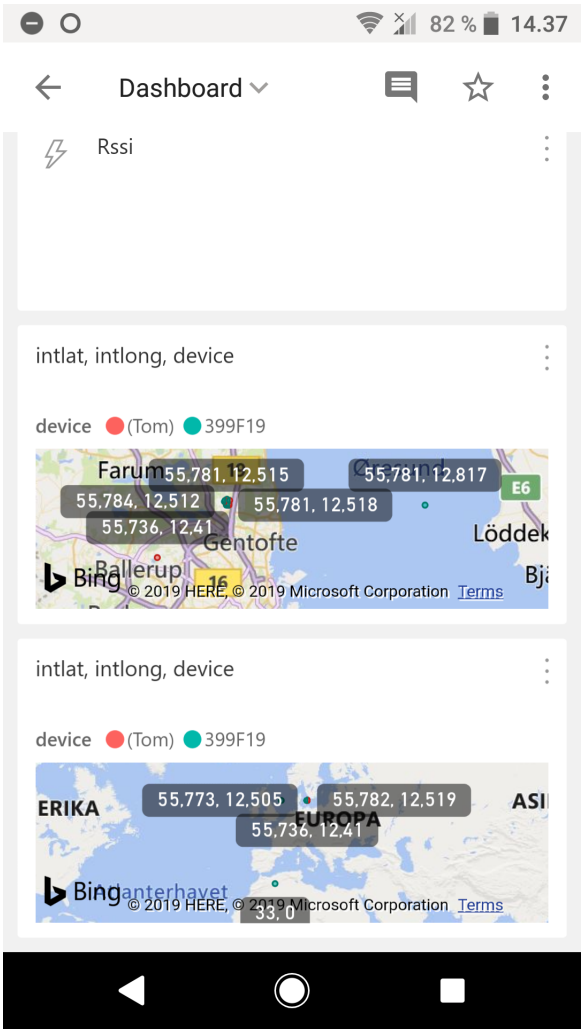


Figure 36: Power BI dashboard from mobile.

7 Conclusion

This bachelor report introduced IoT and the need for LPWANs in order to fulfil some of the requirements of IoT in terms of low power and long range. There was given a theoretical overview of LoRa, Sigfox and NB-IoT. This gave a basic understanding of the different parts of the network architectures and how the data is sent from a device and where it is received. LoRa and Sigfox send data to gateways while NB-IoT sends to base stations. From here LoRa and Sigfox send their data to a backend server, while NB-IoT sends the data to the core network. The technologies have their advantages and disadvantages in terms of their capabilities as LPWANs. A specification overview was given to show some of these capabilities. They differ a bit in what use case their optimal for. It comes down to the key elements: the payload, power consumption and range.
The basic technique behind GPS and localisation, trilateration and multilateration was explained mathematically. The location techniques for the three technologies were also addressed. LoRa can use either TDOA or RSSI to determine a location. TDOA is the preferred option in terms of accuracy and gives a precision of 20-200 m compared to RSSI's 1-2 km. Sigfox uses their own geolocation called Atlas which is based on RSSI. It gives a precision ranging between 1-10 km. While NB-IoT can use either cell-ID or OTDOA. The difference in accuracy for the technologies give them different use cases in terms of tracking when it comes to precision.
A compact transmission device with a GPS module containing LoRa and Sigfox was built on a PCB. The design and implementation phase of the created transmission device was explained and discussed. The design phase explained which requirements there was for the device and how the actual device would look like. The antenna of Sigfox was not steady, which could have caused some interference because of it dangling. The implementation phase addressed step by step the different parts of the data flow, how the software and hardware was implemented but also how the integration with Microsoft Azure happened.
Furthermore two tests with the device were performed: One walking around campus and another driving around campus while transmitting every minute. The test results showed that Sigfox had a success rate of 100 % while walking and LoRa had 66.66 % when it came to successfully received data packets in Microsoft's visualisation application called Power BI. On the contrary when driving Sigfox only had 75 % while LoRa had 83.3 %. These results were visualised on a map in Power BI. The results mean Sigfox is more reliable at locating the device at walking speed, while LoRa is better at driving speeds. The low percentage for Sigfox when driving could be due to the Doppler effect, while walking Sigfox is more reliable than LoRa which could be due to Sigfox's UNB interfering with LoRa's spread spectrum.
At last the future work possibilities were discussed and the future use of the device. In the future NB-IoT should be included as an addition. The device could be used as a tool for companies to test what technology suits their needs in terms of range, payload, data rate etc. and what technology is best suited for their specific IoT use case.

References

[1] IHS Market,https://technology.ihs.com/596542/number-of-connected-iot-devices-will-surge-to-125-billion-by-2030-ihs-markit-says, October 2017, Localised 20-02-2019.

[2] Ruepp, Sarah, Guidelines for M.Sc. theses, B.Sc. projects and Special cources @ DTU Fotonik - Networks - IoT Topics, January 2019.

[3] LinkLabs: A comprehensive look at low power, wide area networks for 'Internet of Things' Engineers and Decision makers, 2016.

[4] LoRaWAN: What is it?, LoRa Alliance November 2015.

[5] Clausen, T., Augustin, A., September 2016, A study of LoRa: Long Range & Low Power Networks for the Internet of Things.

[6] Teracom, https://www.teracom.dk/produkter/IoT/, localised 15-04-2019.

[7] Sigfox Technical overview, Sigfox May 2017.

[8] Sigfox connected objects: Radio specifications, Sigfox, February 2019.

[9] IoTDenmark, https://iotdk.dk/sigfox-iot/, localised 16-04-2019.

[10] Narrowband Internet of Things, Rohde-Schwarz August 2016.

[11] Godsk, Anders, Oktober 2018, Sammenligning mellem NB-IoT og Sigfox.

[12] Huawei: NB-IoT Enabling new business opportunities 2015.

[13] Sauter, Martin, 2014, From GSM To LTE-Advanced: An Introduction to Mobile Networks and Mobile Broadband, 2nd Edition Wiley.

[14] Christiansen, Henrik, 2017, Lecture 12: LTE I, Introduction to Mobile Communication 34330.

[15] The Things Network, https://www.thethingsnetwork.org/docs/lorawan/, localised 15-04-2019.

[16] 3GPP, ETSI TR 121 914, 3GPP TR 21.914 version 14.0.0 Release 14.

[17] Ratasuk, R. et al., Enhancements of narrowband IoT in 3GPP Rel-14 and Rel-15 5, September 2017.

[18] Polson, J., Fette, B.A., https://www.sciencedirect.com/topics/engineering/angle-of-arrival, Localised 21-04-2019

[19] Petersen, Martin N., Fargas, Bernat C., GPS-free Geolocation using LoRa in Low-Power WANs, 2017.

[20] Gu, C., et al.,LoRa-Based Localization: Opportunities and Challenges, January 2019.

[21] LoRaWAN, GEOLOCATION WHITEPAPER, January 2018.

[22] Sigfox, Geolocation: The simplest and cheapest IoT Service.

[23] MEAM ESP32 PINOUT, https://alliance.seas.upenn.edu/ medesign/wiki/index.php /Guides/ESP32-pins?fbclid=IwAR3ekxL6vfyS9dbYTs1LsXzjE-PZTMuIXnwqOgqgX2ZZhc5qIYq76wjSdSA, Localised 02-03-2019.

[24] ublox, NEO-7: ublox 7 GNNS modules datasheet, January 2014.

[25] Microchip, RN2483 LoRa technology module command reference user's guide, 2015.

[26] Wisol, WISOL / WSSFM10R1AT DATA SHEET Rev.12.

[27] Petersen, Martin N., RN2483 hook up guide.

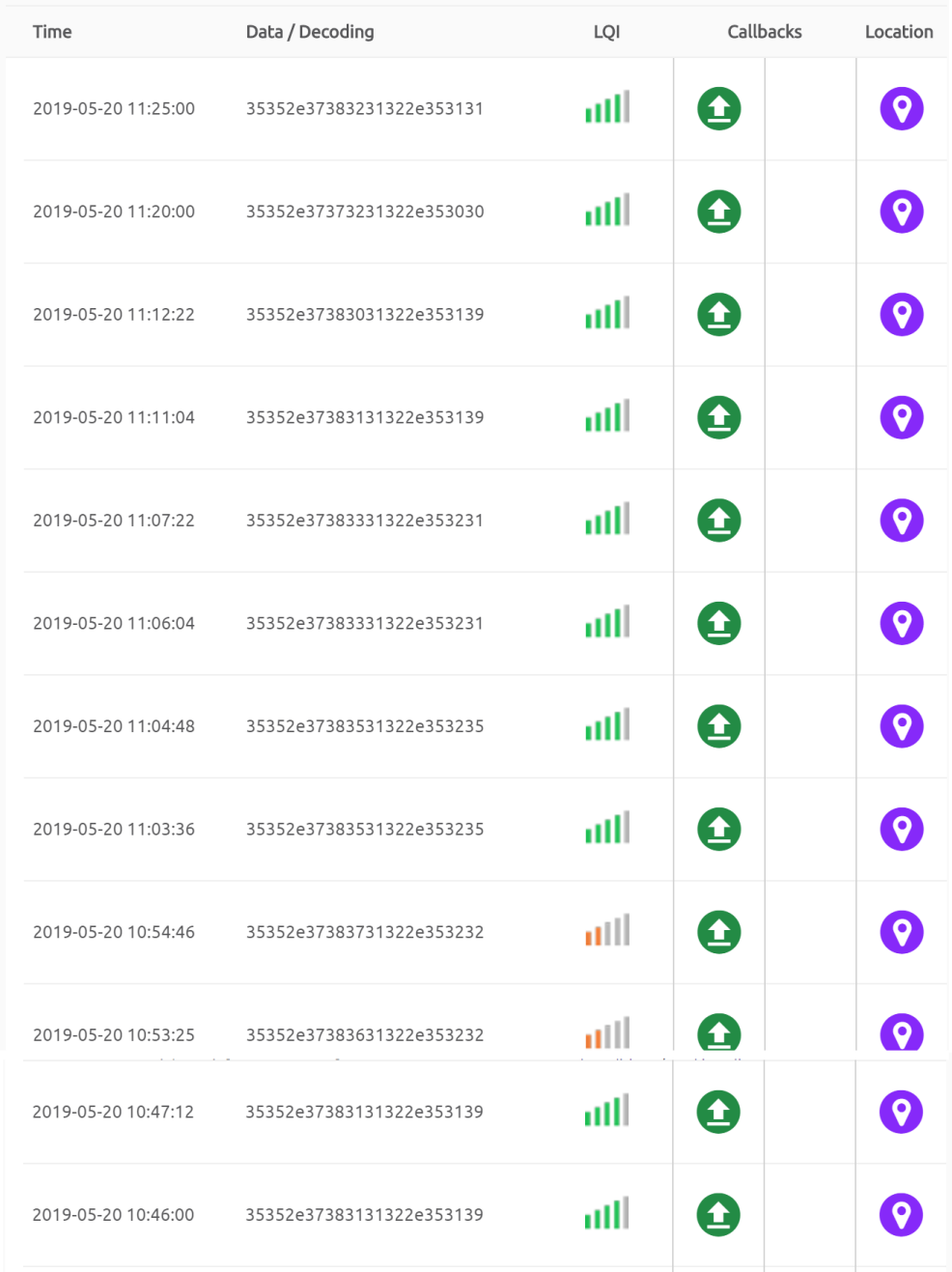
[28] Mekki, K., et al.,A comparative study of LPWAN technologies for large-scale IoT deployment, September 2017.

[29] On Semicondoctor, Ultra-Low Power, AT Command Controlled, Sigfox Compliant Transceiver IC for Up-Link and Down-Link, March 2016.

8 Appendix

8.1 Appendix A

This appendix shows screenshots of the backend data from test 1 and results from Sigfox's geolocation service Atlas.



Time	Data / Decoding	LQI	Callbacks	Location
2019-05-20 11:25:00	35352e37383231322e353131			
2019-05-20 11:20:00	35352e37373231322e353030			
2019-05-20 11:12:22	35352e37383031322e353139			
2019-05-20 11:11:04	35352e37383131322e353139			
2019-05-20 11:07:22	35352e37383331322e353231			
2019-05-20 11:06:04	35352e37383331322e353231			
2019-05-20 11:04:48	35352e37383531322e353235			
2019-05-20 11:03:36	35352e37383531322e353235			
2019-05-20 10:54:46	35352e37383731322e353232			
2019-05-20 10:53:25	35352e37383631322e353232			
2019-05-20 10:47:12	35352e37383131322e353139			
2019-05-20 10:46:00	35352e37383131322e353139			

Figure 37: Received data at the Sigfox backend from test 1.

Last 25 frames received

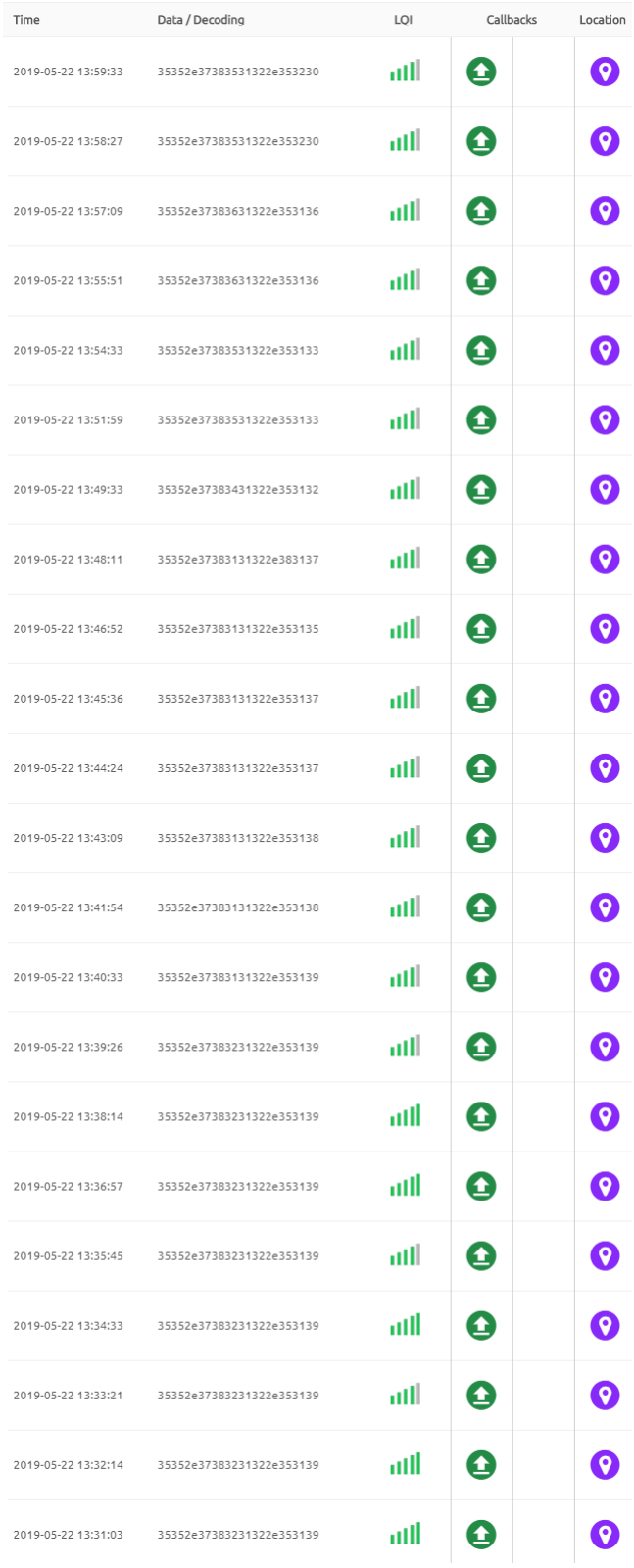
Device EUI	SeqNo	Time	Port	Data
00-04-A3-0B-00-21-86-4A	22	8 minutes ago	1	35 35 2e 37 37 33 31 32 2e 35 30 35
00-04-A3-0B-00-21-86-4A	21	9 minutes ago	1	35 35 2e 37 37 33 31 32 2e 35 30 35
00-04-A3-0B-00-21-86-4A	20	14 minutes ago	1	35 35 2e 37 38 30 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	19	17 minutes ago	1	35 35 2e 37 38 30 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	17	19 minutes ago	1	35 35 2e 37 38 31 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	16	22 minutes ago	1	35 35 2e 37 38 31 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	15	24 minutes ago	1	35 35 2e 37 38 33 31 32 2e 35 32 31
00-04-A3-0B-00-21-86-4A	14	25 minutes ago	1	35 35 2e 37 38 35 31 32 2e 35 32 35
00-04-A3-0B-00-21-86-4A	13	27 minutes ago	1	35 35 2e 37 38 35 31 32 2e 35 32 35
00-04-A3-0B-00-21-86-4A	12	29 minutes ago	1	35 35 2e 37 39 30 31 32 2e 35 32 38
00-04-A3-0B-00-21-86-4A	11	32 minutes ago	1	35 35 2e 37 38 37 31 32 2e 35 32 32
00-04-A3-0B-00-21-86-4A	10	37 minutes ago	1	35 35 2e 37 38 36 31 32 2e 35 32 32
00-04-A3-0B-00-21-86-4A	9	38 minutes ago	1	35 35 2e 37 38 32 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	8	42 minutes ago	1	35 35 2e 37 38 31 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	7	43 minutes ago	1	35 35 2e 37 38 31 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	6	44 minutes ago	1	35 35 2e 37 38 31 31 32 2e 35 31 39

Figure 38: Received data at the Teracom server from test 1.

Figure 39: Sigfox Atlas results from test 1.

8.2 Appendix B

This appendix shows screenshots of the backend data from test 2.



Time	Data / Decoding	LQI	Callbacks	Location
2019-05-22 13:59:33	35352e37383531322e353230			
2019-05-22 13:58:27	35352e37383531322e353230			
2019-05-22 13:57:09	35352e37383631322e353136			
2019-05-22 13:55:51	35352e37383631322e353136			
2019-05-22 13:54:33	35352e37383531322e353133			
2019-05-22 13:51:59	35352e37383531322e353133			
2019-05-22 13:49:33	35352e37383431322e353132			
2019-05-22 13:48:11	35352e37383131322e383137			
2019-05-22 13:46:52	35352e37383131322e353135			
2019-05-22 13:45:36	35352e37383131322e353137			
2019-05-22 13:44:24	35352e37383131322e353137			
2019-05-22 13:43:09	35352e37383131322e353138			
2019-05-22 13:41:54	35352e37383131322e353138			
2019-05-22 13:40:33	35352e37383131322e353139			
2019-05-22 13:39:26	35352e37383231322e353139			
2019-05-22 13:38:14	35352e37383231322e353139			
2019-05-22 13:36:57	35352e37383231322e353139			
2019-05-22 13:35:45	35352e37383231322e353139			
2019-05-22 13:34:33	35352e37383231322e353139			
2019-05-22 13:33:21	35352e37383231322e353139			
2019-05-22 13:32:14	35352e37383231322e353139			
2019-05-22 13:31:03	35352e37383231322e353139			

Figure 40: Received data at the Sigfox backend from test 2.

Device EUI	SeqNo	Time	Port	Data
00-04-A3-0B-00-21-86-4A	12	4 hours ago	1	35 35 2e 37 38 35 31 32 2e 35 32 30
00-04-A3-0B-00-21-86-4A	11	4 hours ago	1	35 35 2e 37 38 35 31 32 2e 35 32 30
00-04-A3-0B-00-21-86-4A	10	4 hours ago	1	35 35 2e 37 38 35 31 32 2e 35 31 33
00-04-A3-0B-00-21-86-4A	9	4 hours ago	1	35 35 2e 37 38 34 31 32 2e 35 31 32
00-04-A3-0B-00-21-86-4A	8	4 hours ago	1	35 35 2e 37 38 31 31 32 2e 35 31 37
00-04-A3-0B-00-21-86-4A	7	4 hours ago	1	35 35 2e 37 38 31 31 32 2e 35 31 37
00-04-A3-0B-00-21-86-4A	6	4 hours ago	1	35 35 2e 37 38 31 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	5	4 hours ago	1	35 35 2e 37 38 32 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	4	4 hours ago	1	35 35 2e 37 38 32 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	3	4 hours ago	1	35 35 2e 37 38 32 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	2	4 hours ago	1	35 35 2e 37 38 32 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	1	4 hours ago	1	35 35 2e 37 38 32 31 32 2e 35 31 39
00-04-A3-0B-00-21-86-4A	0	4 hours ago	1	35 35 2e 37 38 32 31 32 2e 35 31 39

Figure 41: Received data at the Teracom server from test 2.

8.3 Appendix C

This appendix shows a gant chart over the project.

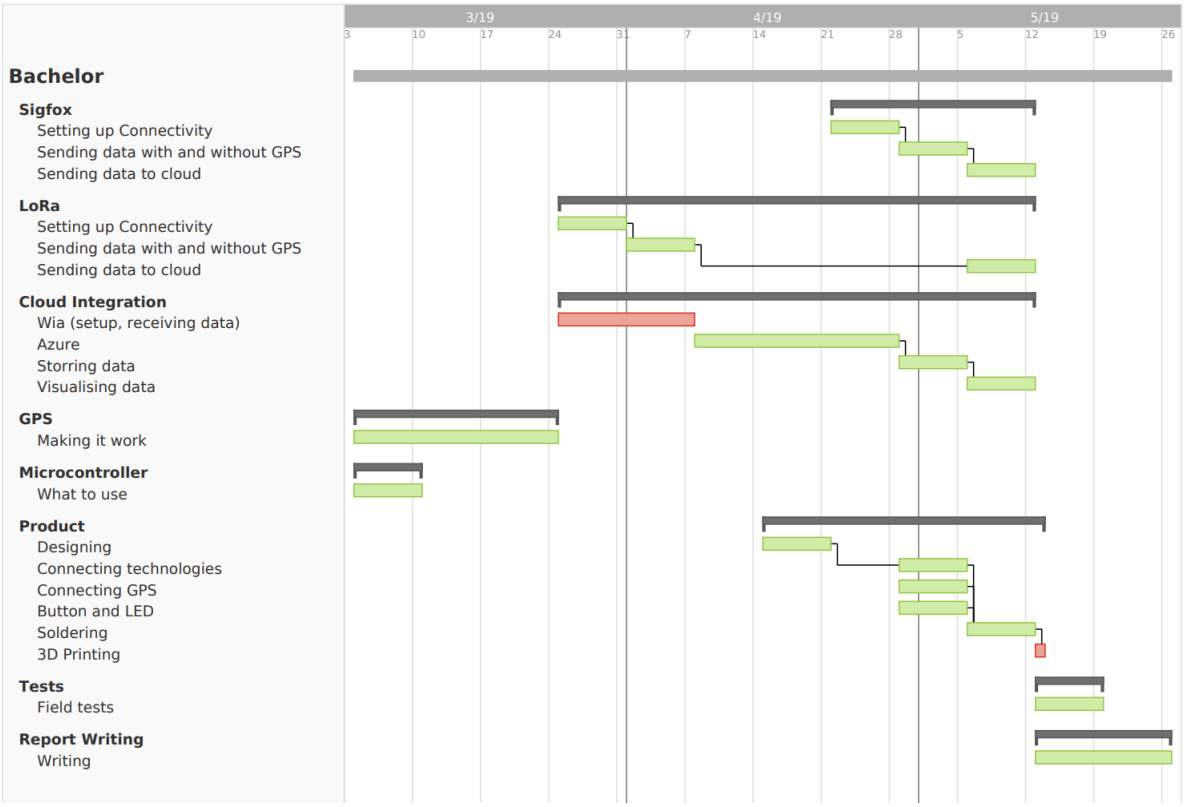


Figure 42: Gant chart over the project. Red indicates dropped and green indicates done.