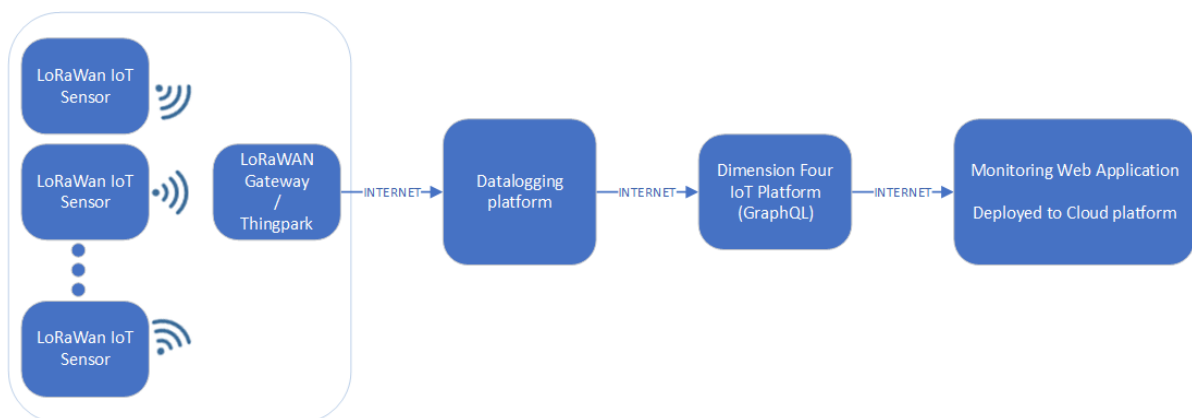


FM4017 Project 2022

# Development of Internet of Things (IoT) Solution using LoRaWAN Infrastructure for Storing and Monitoring Sensor Data



MP-15-22

**Course:** FM4017 Project, 2022

**Title:** Development of Internet of Things (IoT) Solution using LoRaWAN Infrastructure for Storing and Monitoring Sensor Data

This report forms a part of the basis for assessing students' performance in the course.

**Project group:** MP-15-22

**Group participants:** Amila Ruwan Guruge  
Eivind Knudsen  
Noorain Syed Kazmi  
Vitaly Dekhtyarev

**Supervisor:** Hans-Petter Halvorsen

**Project partners:** Dimension Four AS, Altibox AS

**Summary:**

The purpose of this project is to create an open-source end-to-end solution for easy monitoring and analysis of data from LoRaWAN sensors. LoRaWAN devices and technology are explored. A datalogging platform is developed to log sensor data to the Dimension Four IoT platform. A web application that displays current sensor signals, as well as historical data, has been developed.

*The University of South-Eastern Norway takes no responsibility for the results and conclusions in this student report.*

# Preface

This report is prepared as part of the course FM4017 Project 2022 by Master of Science, Industrial IT and Automation students at the University of South-Eastern Norway at Porsgrunn. The project task was “Development of Internet of Things (IoT) solution using LoRaWAN Infrastructure for storing and monitoring sensor data”.

During the project, we as a project team worked meticulously and excitedly to learn about LoRaWAN technology, devices, GraphQL, Dimension Four IoT platform, Altibox ThingPark portal, datalogging using Python, and data monitoring using C# among other aspects of creating an open-source end-to-end solution for University of South-Eastern Norway (USN). Mr. Daniel Wathne Warholm from Altibox AS was actively involved in our project and guided us at every corner of the implementation work. We thank him sincerely for all the knowledge, guidance, and sustained support in our project. We thank our project supervisor, Mr. Hans-Petter Halvorsen for providing us with a great project task, decision-making power, dedicated support, and guidance in every way possible to the project team. We also thank the Dimension Four team for their interest in our project and the solution.

Porsgrunn, 18.11.2022

Amila Ruwan Guruge

Eivind Knudsen

Noorain Syed Kazmi

Vitaly Dekhtyarev

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>8</b>
1.1	Background .....	8
1.2	Dimension Four AS.....	8
1.3	Altibox AS.....	8
1.4	Objective.....	9
1.5	Report Structure .....	9
<b>2</b>	<b>System Description.....</b>	<b>10</b>
<b>3</b>	<b>LoRaWAN Protocol .....</b>	<b>11</b>
3.1	Infrastructure Description.....	12
3.2	Device Classes.....	13
<b>4</b>	<b>Dimension Four IoT .....</b>	<b>15</b>
4.1	Dimension Four IoT Platform .....	15
4.2	Structure of data .....	15
4.2.1	Tenant.....	16
4.2.2	Space.....	16
4.2.3	Point .....	16
4.2.4	Signal.....	17
4.3	GraphQL .....	17
4.4	Dimension Four Playground.....	19
<b>5</b>	<b>Sensors and Associated Hardware .....</b>	<b>20</b>
5.1	Adeunis Field Test Device .....	20
5.2	Adeunis TEMP.....	21
5.3	Adeunis COMFORT.....	22
5.4	Adeunis Dry Contacts .....	23
5.5	Altibox's ThingPark Portal .....	24
5.6	Network Coverage at USN Porsgrunn .....	27
5.7	Network Coverage at Porsgrunn City .....	28
5.8	Quality of Network .....	29
5.9	Installation of Gateway.....	31
<b>6</b>	<b>Datalogging Application.....</b>	<b>33</b>
6.1	ThingPark Setup .....	33
6.2	Webhook.....	41
6.3	Python Application .....	43
6.4	Alternative Application Server .....	47
6.5	Message Transformation .....	48
<b>7</b>	<b>Monitoring Application .....</b>	<b>50</b>
7.1	Main Technical Stack .....	50
7.1.1	ASP.NET Core Blazor Server .....	50
7.1.2	SignalR .....	51
7.2	Architecture of the Monitoring Application .....	51
7.3	Access to Data in Dimension Four IoT Platform .....	52
7.3.1	HTTP .....	52
7.3.2	GraphQL.....	52
7.3.3	Serialize and Deserialize JSON Data in .NET .....	53
7.4	Deployment .....	53
7.5	User Interface .....	54

7.5.1 Publicly Available Page .....	54
7.5.2 Administration (Admin) Page.....	56
<b>8 Discussion.....</b>	<b>58</b>
<b>9 Conclusion .....</b>	<b>59</b>
<b>References .....</b>	<b>60</b>
<b>Appendix.....</b>	<b>63</b>
Appendix A: Project description .....	63

# Nomenclature

ABP: Activation-By-Personalization

API: Application Programming Interface

AS: A private limited liability company in Norway/ Application Server

ASP.NET: Active Server Pages Network Enabled Technologies

AWS: Amazon Web Services

CRUD: Create, Read, Update, Delete

CSS: Chirp Spread Spectrum modulation/ Cascading Style Sheets

dB: Decibel

dBm: Decibel milliwatts

DevEUI: Device Extended Unique Identifier

DTO: Data transfer object

ED: End Device

FTD: Field Test Device

FY: Fiscal year

GPS: Global Positioning System

GraphQL: Graph Query Language

HTTP: Hypertext Transfer Protocol

IP68: Ingress Protection rating

IP: Internet Protocol

IoT: Internet of Things

JMESPath: JSON Matching Expression paths

JSON: JavaScript Object Notation

JoinEUI: Join Extended Unique Identifier

Kbits/sec: Kilobits per second

LoRa: Long Range

LoRaWAN: Long Range Wide Area Network

MQTT: Message Queuing Telemetry Transport

MS: Microsoft

MVC: Model-View-Controller

NOK: Norwegian Krone

NS: Network Server

OTA: Over-The-Air  
PC: Personal Computer  
PER: Packet Error Rate  
RG: Radio Gateway  
RPC: Remote Procedure Calls  
RSSI: Received Signal Strength Indicator  
SaaS: Software as a service  
SDL: Schema Definition Language  
SNR: Signal to Noise Ratio  
TV: Television  
URL: Uniform Resource Locator  
USB: Universal Serial Bus  
USN: University of South-Eastern Norway  
UUID: Universally Unique Identifier  
VPN: Virtual Private Network  
VoIP: Voice over Internet Protocol  
Wi-Fi: Wireless Fidelity  
XML: Extensible Markup Language

# 1 Introduction

In this project, LoRaWAN network infrastructure provided by Altibox AS is employed to acquire signals and data from various LoRaWAN sensors. The acquired data is stored in a structured form using the Dimension Four IoT API platform. The stored data is displayed on a web application for easy monitoring and analysis.

## 1.1 Background

In May 2022, Dimension Four AS and Altibox AS announced their partnership to support innovative projects in academics and research by providing low-cost use of Altibox's LoRaWAN network infrastructure in association with the Dimension Four IoT platform where data can be structured and stored. [1]

Since the autumn of 2018, the Altibox and Altibox partners have expanded the LoRaWAN Sensor Network in Norway and currently, the network has coverage for more than 1 million households in 100 municipalities. The LoRaWAN Sensor Network is a natural extension of the Fiber network with major synergies of established infrastructure and the development continues. Altibox and its partners now offer IoT Network Access as a commercial service, so more people can use the LoRaWAN Sensor Network for their own sensors.

## 1.2 Dimension Four AS

Dimension Four AS is a software development company located in Skien, Norway. It provides end-to-end IoT cloud services for storing data from sensors through APIs. They focus on removing the complexity of the IoT backend by providing easy to use IoT platform.

Dimension Four is a part of the New Normal Company which is an entrepreneurial company supporting and investing in new and independent startup companies [2]. The company has invested in various startups such as Crystallize, Froya Life, Okay, Cemit, Dimension Four, etc. [3]

## 1.3 Altibox AS

Altibox AS is a fully owned subsidiary of Lyse AS providing digital entertainment, fiber broadband, TV, IP VPN, Internet, gaming and VoIP services to business and households in Norway. Altibox is a member of LoRa Alliance® that has created and manages LoRaWAN network. It currently covers around 1 million households in Norway. At the end of FY21, Altibox had a total of 784, 918 customers and its revenue was NOK 957 million. [4]



## 1.4 Objective

This project aims to explore LoRaWAN devices and create applications for datalogging and monitoring. A list of major tasks for this project is provided below.

- Get an overview about the LoRaWAN technology and LoRaWAN devices provided by Altibox
- Document LoRaWAN signal coverage in the USN, Porsgrunn campus and install LoRaWAN devices at USN, Porsgrunn
- Create a datalogging application to send sensor signals from the ThingPark portal to Dimension Four IoT platform for storage
- Build a publicly available monitoring web application deployed on cloud that retrieves and displays sensor data which is stored in Dimension Four IoT platform

## 1.5 Report Structure

The report starts by explaining the system sketch, giving an overview of the system to be developed in order to fulfil the objectives of the project. Then, a general introduction to LoRaWAN protocol and Dimension Four IoT platform along with the data structure is given. An introduction to the GraphQL query language is also presented. Further to this, information about LoRaWAN devices used in this project is provided along with their installation and usage. Afterwards, the methods for the development of datalogging platform are documented and discussed in detail with results. Lastly, the methods and results related to creation and implementation of monitoring web application are presented.

## 2 System Description

The overview of the system as a system sketch is displayed in Figure 2.1.

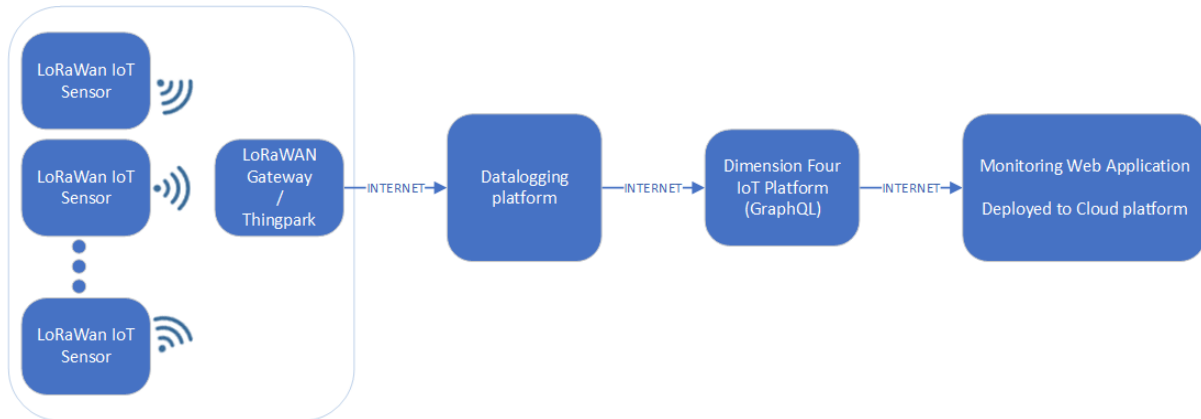


Figure 2.1: System sketch

As per the system sketch, the system starts with installing and testing LoRaWAN sensors and LoRaWAN network coverage at USN, Porsgrunn. The data from the sensors is pre-configured to be forwarded to the Altibox ThingPark portal through Altibox LoRaWAN network infrastructure. ThingPark portal is configured to forward the LoRaWAN sensor data to a webhook site to be used for further processing in the datalogging platform. After the datalogging platform retrieves the data from the webhook, it is decoded and published to Dimension Four IoT platform in a structured form using GraphQL.

When the data is published on Dimension Four IoT platform, a monitoring application retrieves it using GraphQL and displays processed data on a publicly available cloud-based web page.

An administration (admin) section is also included in the monitoring application to perform CRUD operations of create, read, update and delete spaces and points in Dimension Four IoT platform for a particular tenant of Dimension Four.

### 3 LoRaWAN Protocol

Frequency of modulation and its method defines the coverage of radio networks and the capacity of transmitted data. LoRa is a patented and proprietary peer-to-peer radio communication technique based on spread spectrum modulation through chirp spread spectrum (CSS). LoRaWAN, a part of low power wide-area-network (LPWAN), defines a long-range communication protocol and architecture employing LoRa modulation [5]. The protocol is managed by the LoRa Alliance [6]. Battery operated devices such as Internet of things (IoT) sensors can wirelessly and securely connect to the internet through LoRaWAN protocol to enable low-cost applications. The advantages of using LoRaWAN for IoT are shown in Figure 3.1. [7] [8]

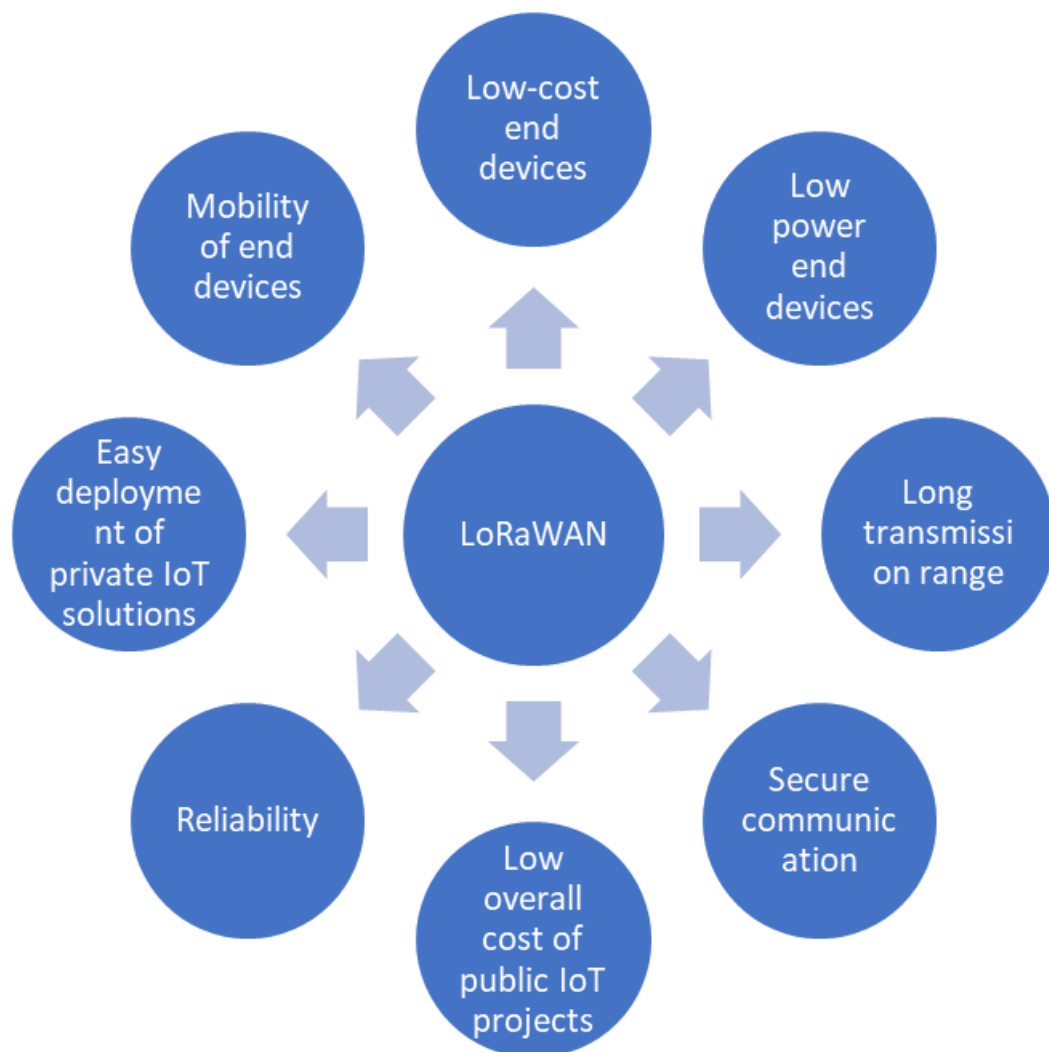


Figure 3.1: Advantages of LoRaWAN for IoT applications

Communication protocols such as Bluetooth, Wi-Fi and 5G are already in operation worldwide and documented. Comparison of LoRaWAN with these currently available communication protocols is shown in Table 3.1.

Table 3.1: Comparison of LoRaWAN with other radio communication protocols [6] [9] [10] [11]

	<b>Bluetooth 5</b>	<b>Wi-Fi 802.11n</b>	<b>5G</b>	<b>LoRaWAN</b>
Frequency of operation	2.45 GHz	2.4 GHz, 5 GHz	28 GHz, 39GHz	169Mhz, 433MHz (Asia), 868MHz (Europe), 915 MHz (North America)
Range	Short (10m)	Short (100m)	Long (300m)	Very long (20km)
End Device battery drain	Slow (1 day)	Slow (1 day)	Fast (1/2 day)	Very slow (1 – 10 years)
Cost for IoT deployment	Low	Low	High	Low
Data rate	2 Mbps	600 Mbps	20 Gbps	27 kbps
Security	Low	Low	High	High

Looking at the comparison and the advantages, it can be seen that LoRaWAN is suitable solution for developing IoT and monitoring applications because it allows long battery life in devices (a battery charge for LoRaWAN device may last several years), an exceptionally long range of operation so that IoT applications can be built for remote and economically unfavorable areas of the world and the cost of the entire IoT solution is very low.

### 3.1 Infrastructure Description

LoRaWAN implements a star-of-stars topology [12] and provides diverse ways of communication for mobile and stationary devices [13]. Schematically the infrastructure of LoRaWAN is shown in Figure 3.2.

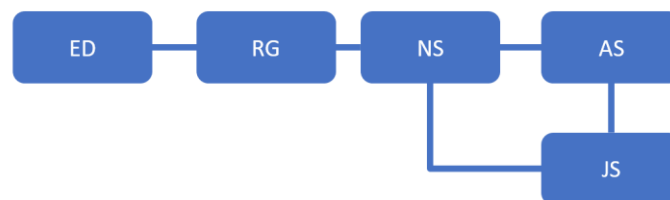


Figure 3.2: LoRaWAN Infrastructure

The End-device (ED) represents a device that is remotely connected to Radio Gateway (RG), with a radio signal, RG may also be called a concentrator or base station. The function of RG is to decode the packets received from ED and send them further to Network Server (NS) using a secured Internet Protocol (IP).

RG also encodes packets from NS for transmission using radio signal to ED.

NS is the center of the topology. It responds to requests from ED and provides data exchange between ED and corresponding Application server (AS) and between ED and Join server (JS). NS is also responsible for holding description parameters of ED such as Device Profile, Service Profile, Routing Profile and Device Extended Unique identifier (DevEUI), which are then used for communication.

Device Profile is needed to establish a wireless communication with ED, Service Profile is required to setup interface with AS and Routing Profile is needed by NS to set up forwarding of data from ED to a correct AS.

JS has a Join Extended Unique identifier (JoinEUI), the server performs activation of EDs. Each ED sends a Join-request downlink to JS using JoinEUI of that JS. The JS responds with a Join-accept uplink to ED. Several NS may be connected to several JS. This procedure is used for Over-The-Air (OTA) activation. But ED may also be activated using Activation-By-Personalization (ABP), in this case all necessary information is provided manually and ED can start communication as soon as it is switched on.

AS processes application-level data EDs and provides it to the end-user, at the same time, AS generates payloads from application-level requests towards the ED. NS provides data transmission between ED and AS using DevEUI. As with JS, multiple AS may be connected to multiple NS.

## 3.2 Device Classes

There are three classes of ED that are specified in LoRaWAN documentation: Class A, Class B and Class C. All three classes are defined as EDs with support for bidirectional communication. However, the difference between the EDs is the way they receive downlink from gateway.

Class A is the most power-saving class as EDs only listen to downlink messages during two brief time windows right after they had sent an uplink message. The ED has preconfigured parameters that determine the time delay after receiving a downlink message and the duration of time windows. The duration, however, must last long enough so that downlink can be properly recognized. Any message sent to ED outside the open receive time windows will be postponed until the window is open. This functionality must be available on all ED that are used in LoRa network.

Class B, in addition to listening windows as of Class A, provides periodic windows when it can receive downlinks. For that purpose, RG sends a periodic beacon to synchronize the devices and using that beacon ED opens additional receive windows, called “ping slots” on a deterministic basis. The data rate, data channel and frequency of sending periodic packets must be provided to NS out-of-band. Later these parameters can be modified when ED is already

connected to the network. If the device cannot receive synchronization beacon from RG, it is automatically reconfigured to Class A, then Class B set up procedure must be reinitiated.

Class C has continuously open receiving window with settings RX2 that is only closed when device is transmitting data on RX1. However, ED switches receive window from RX2 back to RX1 shortly after transmission is over on RX1 and the ED is already on RX2.

Schematically, transmission schedule for the existing classes is shown in Figure 3.3. Length of RECEIVE\_DELAY1, RECEIVE\_DELAY2 and data rates for windows RX1 and RX2 are defined for a particular region. The default values for RECEIVE\_DELAY1 is one second and RECEIVE\_DELAY2 is RECEIVE\_DELAY1 + 1 second. [14]

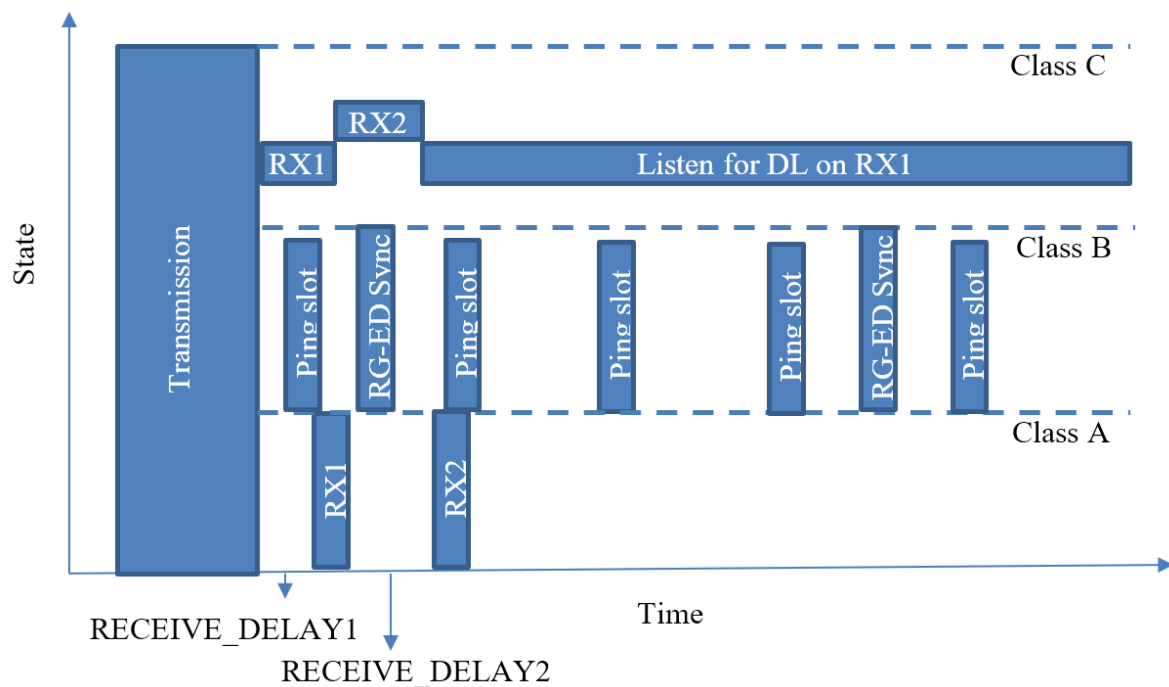


Figure 3.3: Transmission schedule for various classes of LoRaWAN devices

# 4 Dimension Four IoT

## 4.1 Dimension Four IoT Platform

In 2021, Dimension Four launched their IoT platform [2] that aimed to simplify complexity for developers, mainly by providing an easy to setup vendor friendly backend data storage system. Additionally, the platform can support large IoT projects and includes a user friendly GraphQL API. The stored data can be fetched by posting a GraphQL query to the Dimension Four IoT platform server, and server-side data can be modified by posting a mutation query. The data storage structure in Dimension Four IoT platform has the following four main types: Tenant, Space, Point and Signal.

## 4.2 Structure of data

The hierarchy of Dimension Four IoT platform types created for this project is depicted in Figure 4.1. It consists of four main parts. At the top of the hierarchy “Tenant” is located.

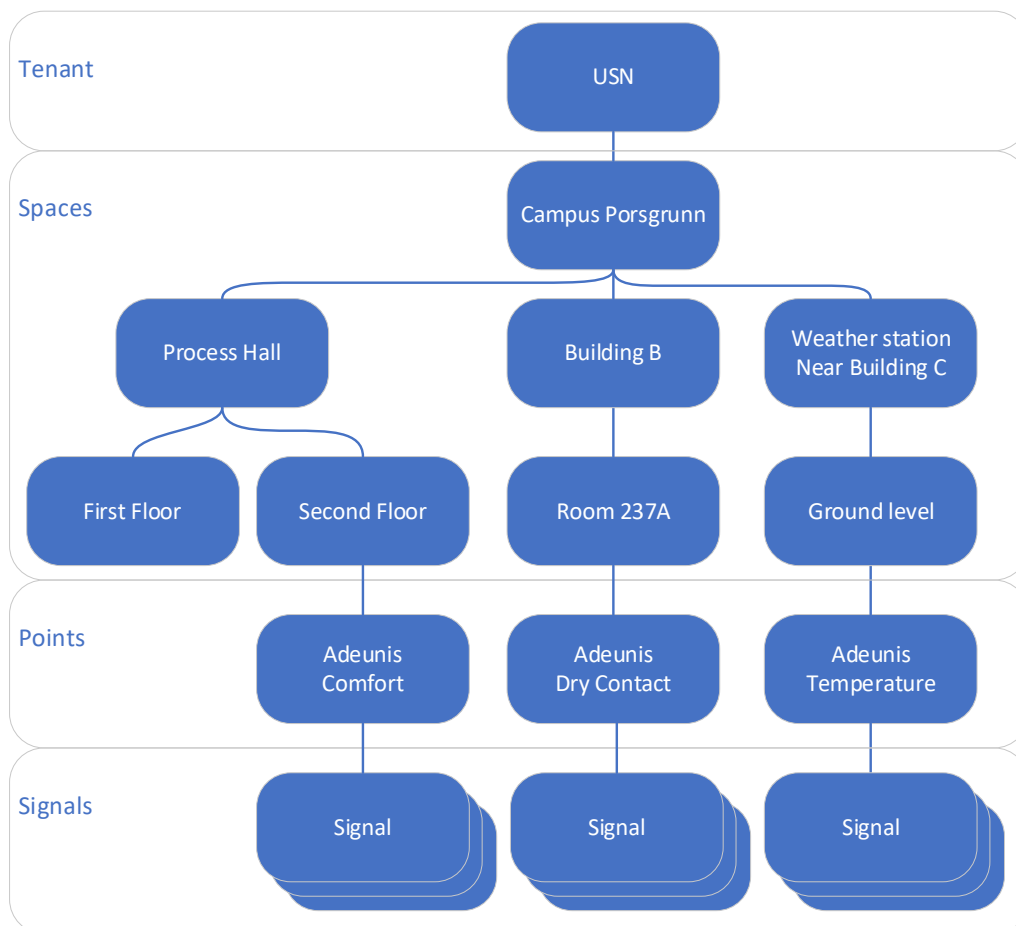


Figure 4.1: Structure of data in Dimension Four IoT platform, showing the three sensors used as Points and the location in the hierarchy

### 4.2.1 Tenant

Dimension Four IoT is a multi-tenant SaaS service. Usually, a business has one tenant. A tenant can be an organization which is a client of Dimension Four, which is purchasing its services. The tenant can include one or more Spaces. Dimension Four IoT platform can have a lot of Tenants as its clients. For this project, “USN1” is used as tenant name. A tenant has its own set of spaces, points, signals, etc. [15]

### 4.2.2 Space

Spaces can have subspaces and they can be considered as a folder. They can hold zero or more points. But there can only be one level of points and signals.

A Space is shown in Figure 4.2.

```
{
  "id": "63245fbc988748848d514544",
  "name": "Weather Station Near Building C",
  "parent": {
    "id": "63230e1c6ebecd0323d13b10"
    "name": "Campus Porsgrunn"
  },
  "metadata": {
    "longitude": 59.13818553442762,
    "latitude": 9.67317761699199,
    "imageUrl": ""
  }
}
```

Figure 4.2: A Space in Dimension Four IoT platform

### 4.2.3 Point

A point is a signal sender. It represents the source of one or more signals (e.g., a temperature sensor). Figure 4.3 code shows a structure of a Point created for a temperature sensor. It has a space id which indicates the space this point belongs to.

```
{
  "id": "6362c7defc18de3434dd2ab3",
  "externalId": null,
  "name": "Adeunis Temperature",
  "metadata": {
    "longitude": 9.673333,
    "latitude": 59.138056,
    "imageUrl": "https://www.adeunis.com/wp-content/uploads/2020/07/temperature-capteur-iot-lorawan-sigfox-lpwan-adeunis-temp.jpg",
    "DevEUI": "0018B21000004540"
  },
  "spaceId": "63246054988748b37851455c"
}
```

Figure 4.3: A point in Dimension Four IoT platform



#### 4.2.4 Signal

A signal represents a single measurement or data at a certain time. It consists of the signal data packaged with a timestamp.

Figure 4.4 code represents a Signal containing the air temperature measurement of 5.9 taken at a particular time (timestamp) with unit in Celsius. All the values are strings and Signal has a Point id from which is the source of the signal.

```
{
  "id": "637080dc97b09a57374ab6ac",
  "pointId": "6362c7defc18de3434dd2ab3",
  "type": "Air temperature",
  "unit": "CELSIUS_DEGREES",
  "data": {
    "rawValue": "5.9"
  },
  "timestamp": "2022-11-13T05:27:01.246Z"
}
```

Figure 4.4: A signal in Dimension Four IoT platform

### 4.3 GraphQL

GraphQL is an open-source and advanced query and manipulation language specifically developed to extract data from API (Application Programming Interface) in an efficient way. It can extract extremely specific data from an API and returns accurately predictable results. This results in the development of stable applications.

GraphQL has four core concepts. They are

- Schema
  - The Schema can be considered as the center of the GraphQL. It describes the functionalities available for the users.
- SDL
  - SDL is known as the Schema Definition Language. It is not a part client are engaged with. Dimension Four IoT platform uses it to define the Schema available for the client.
- Query
  - The Query is the API request made by client to read or fetch the data.
- Mutation
  - Mutations are used to change the dataset in GraphQL. It helps to create, delete, or update Spaces, Points. Also, mutations are used to create signal values in Dimension four GraphQL in the datalogging application.

One of the main benefits of GraphQL is that clients can define a structure of data for queries, and they will receive exactly what they need. So, unnecessary data will not be received.

An example of POST request is shown in Figure 4.5.

```
query SpaceNames{
  points{
    nodes{
      name
      id
      space{
        name
        parent{
          name
          parent{
            name
            parent{
              name
            }
          }
        }
      }
    }
  }
}
```

Figure 4.5: A POST request

Figure 4.6 shows the response from API.

```
{
  "data": {
    "points": {
      "nodes": [
        {
          "name": "Adeunis Temperature",
          "id": "6362c7defc18de3434dd2ab3",
          "space": {
            "name": "Ground Level",
            "parent": {
              "name": "Weather Station Near Building C",
              "parent": {
                "name": "Campus Porsgrunn",
                "parent": null
              }
            }
          }
        }
      ]
    }
  }
}
```

Figure 4.6: The response to the POST request

## 4.4 Dimension Four Playground

Dimension Four's GraphQL service can be easily accessed by using the URL <https://iot.dimensionfour.io/graph>. It gives a web application that can run GraphQL queries. When running the queries, Dimension Four IoT platform needs to know which client is trying to connect and fetch or change data. So, the authentication part is important. This is achieved by using HTTP headers.

There are two headers required to connect to the Dimension Four IoT platform. They are “x-tenant-id” and “x-tenant-key”. The first one specifies the name of the tenant; client is trying to connect. The second one specifies the key required to authenticate.

There are two types of keys available, Read Only and Read/Write. Read/Write key type can perform mutations. Read Only key type is only capable of fetching data.

A simple query fetched using Dimension Four's playground is shown in Figure 4.7.

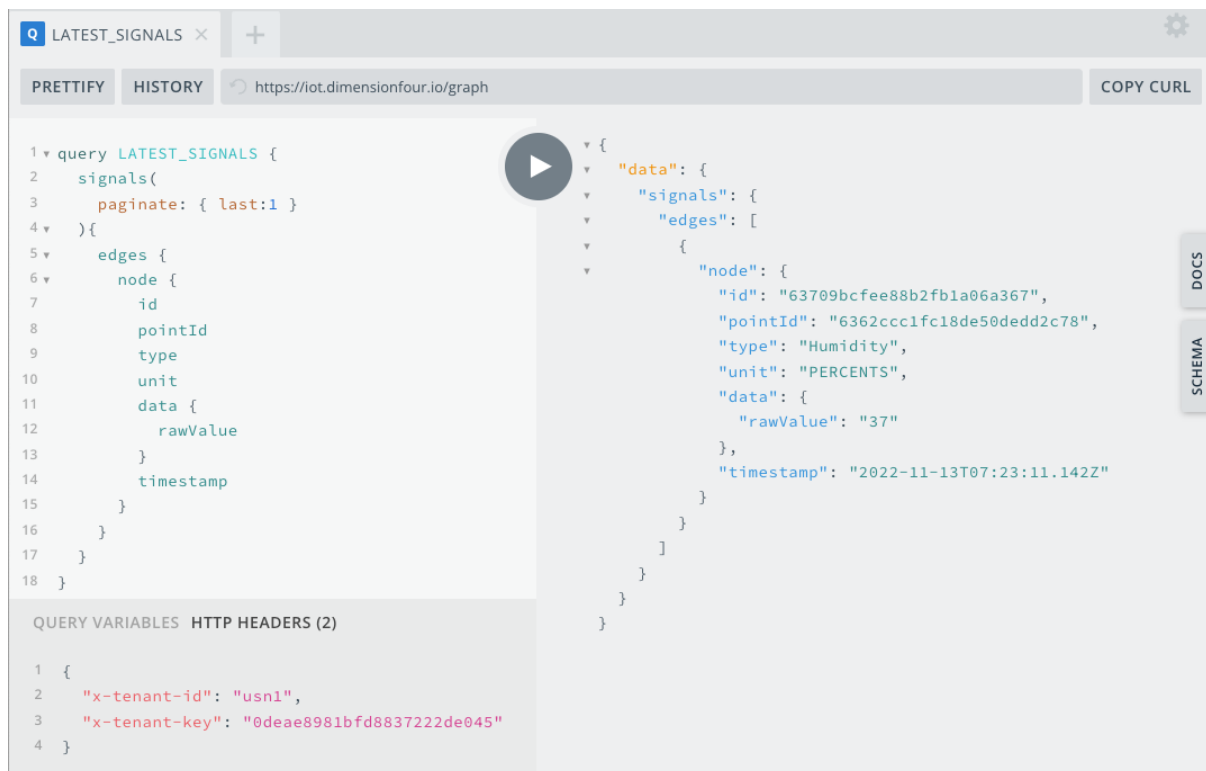


Figure 4.7: A query fetched using Dimension Four's playground

## 5 Sensors and Associated Hardware

Altibox was the main point-of-contact for this project. For the project, four LoRaWAN devices manufactured by Adeunis, France, were received by the project team from the client Altibox for deployment and testing on the USN, Porsgrunn campus, namely, FTD: network tester, TEMP: temperature sensor, COMFORT: temperature / humidity sensor and Dry Contacts: 0-1 status sensor. An indoor gateway (UFISPACE PICO CELL) was also provided by Altibox for testing the improvement of the strength of LoRaWAN network at USN, Porsgrunn.

### 5.1 Adeunis Field Test Device

Adeunis FTD field test device network tester is shown in Figure 5.1. It displayed the LoRaWAN network coverage at its location and was used to detect the quality of LoRaWAN network at USN, Porsgrunn for effective installation of LoRaWAN sensors. Additionally, it had a GPS geolocation sensor and an ambient temperature sensor [16].



Figure 5.1: Adeunis FTD: network tester

## 5.2 Adeunis TEMP

Adeunis TEMP temperature sensor is shown in Figure 5.2. TEMP provided ambient temperature signal in °C. The temperature sensor had two probes, one on the device and another at the end of the 2 m cable attached to the device. It was rated IP68 so that it could be installed outdoors in any kind of weather [17].



Figure 5.2: Adeunis TEMP: temperature sensor

TEMP was installed on the small weather station near Building C. Two screws and a screwdriver were used to attach it to the wall. It was observed that the sensor provided satisfactory performance during day, night, rain, below 0 °C temperatures. The installed sensor is shown in Figure 5.3.

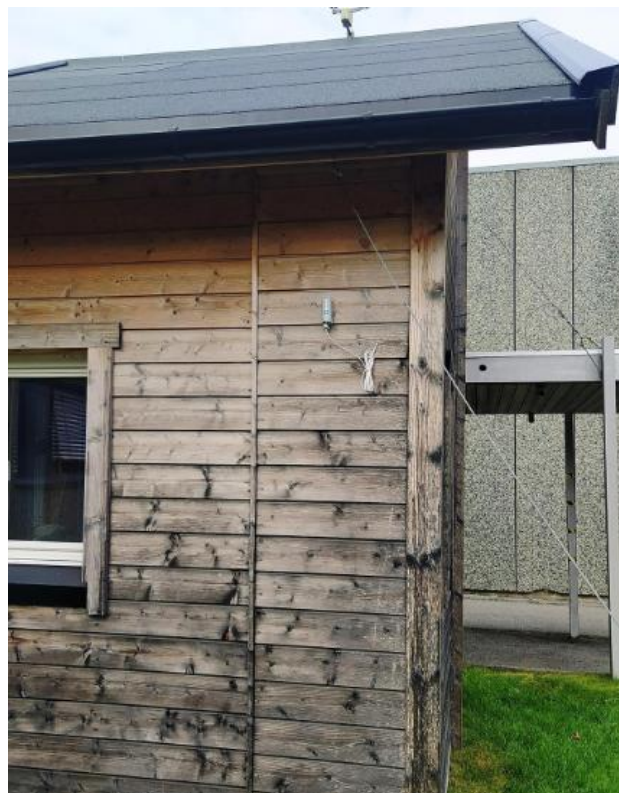


Figure 5.3: TEMP sensor installed on a small cottage near Building C

### 5.3 Adeunis COMFORT

Adeunis COMFORT temperature and relative humidity sensor is shown in Figure 5.4. COMFORT provided ambient temperature in °C as well as ambient relative humidity in percentage (%) [18].



Figure 5.4: Adeunis COMFORT: temperature & relative humidity sensor

This sensor could only be installed in an indoor environment since it was without an IP68 rating. It was installed on a wall on the second floor in the Process Hall. One screw was used to install it. The installed sensor is shown in Figure 5.5.



Figure 5.5: COMFORT sensor installed on a wall in Process Hall shown located in the yellow circle

## 5.4 Adeunis Dry Contacts

Adeunis Dry Contacts 0-1 status sensor is shown in Figure 5.6. It had 4 configurable inputs/outputs. The sensor could be configured remotely. It was rated Class C [19]. It could be configured to send 0-1 status at specific periods or at the onset of an opening/closing event. It was installed on the door of Room B237a with two screws and a tape as shown in Figure 5.7.



Figure 5.6: Adeunis Dry Contact sensor



Figure 5.7: Dry Contacts sensor installed on the door of Room B237a



## 5.5 Altibox's ThingPark Portal

The devices were already on-boarded by the Altibox team on the Altibox's ThingPark portal (<https://altibox.thingpark.com>). The access to the ThingPark portal was shared with the project team. The front page of ThingPark portal after logging in is shown in Figure 5.8.

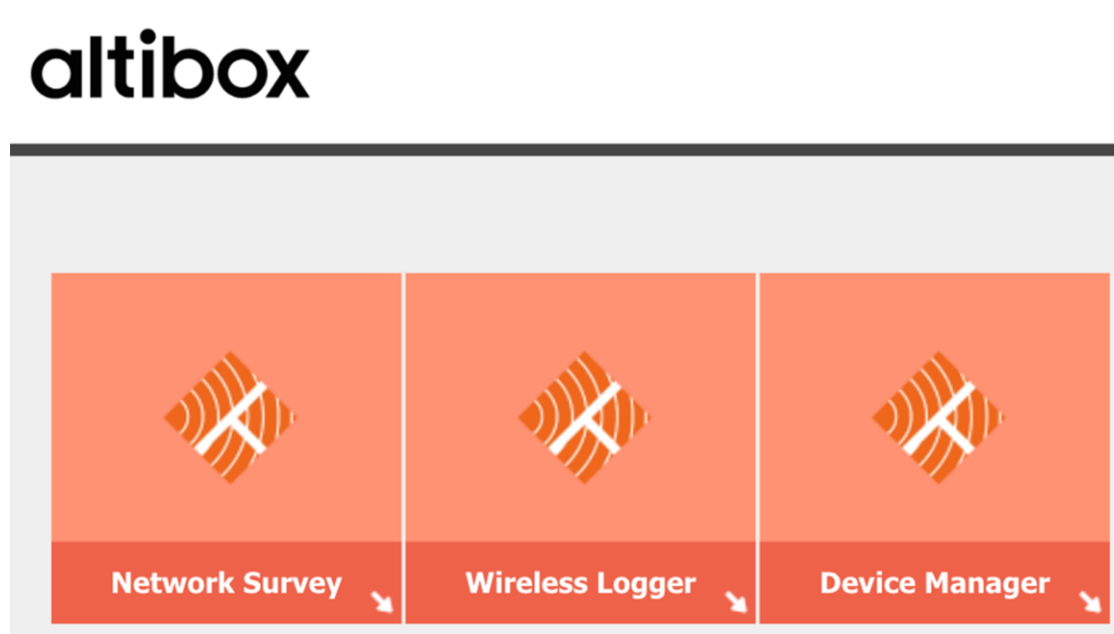


Figure 5.8: ThingPark portal subscriber front page

The project team was given access to the Network Survey, Wireless Logger and Device Manager options in the ThingPark portal. The project team could only access the four devices through this portal and the three options. In the Device Manager option, the devices were displayed as shown in Figure 5.9. This page also displayed important network parameters such as Mean PER (Packet Error Rate), Average SNR (Signal to Noise Ratio) and Battery level among other parameters for each device.

Name / Type	Identifiers	Connectivity	Average packets	Mean PER	Average SNR	Battery	Alarm	Locate	
Adeunis - Comfort Comfort Sensor	0018B210000038BF FC004EA5	ALTIBOX Standard XL TPX	161.0/day	0.0%	8.9 dB		5		
Adeunis - Dry Contact Dry Contact Sensor	0018B22000001FD2 FC004E1F	ALTIBOX Standard XL TPX	26.0/day	3.8%	-0.7 dB		2		
Adeunis - Field Test Device Field Test Device	0018B2000000263A8 FC004F8A	ALTIBOX Standard XL D4	0.0/day	0.0%	3.4 dB		6		
Adeunis - Temperature (Wire) Temperature Sensor	0018B21000004540 FC004E0E	ALTIBOX Standard XL TPX	150.0/day	2.0%	4.7 dB		5		

Figure 5.9: Device list in ThingPark portal

In the Connectivity plans of Device Manager option as shown in Figure 5.11, 4 devices were installed. The name of the plan was ALTIBOX Connectivity Supplier / ALTIBOX Standard XL (6) with LoRaWAN connectivity. The complete service of this plan is shown in Figure 5.10.



It can be seen that maximum and minimum Spreading Factors are not set, Class B support is disabled, and routing is enabled for third party Application Servers (webhooks) and ThingPark X (MQTT, AWS, Azure, etc.).

Connectivity plan details	
Connectivity Plan Name:	ALTIBOX Connectivity Supplier / ALTIBOX Standard XL (6)
Connectivity Plan ID:	altibox-cs/altibox-standard-xl
Connectivity:	LoRaWAN
Communication Type:	Unicast
Connectivity Plan Description:	Altibox Standard XL

Uplink frame parameters	
Acknowledged uplink frame:	Enabled
Rate regulator:	144 frame(s)/day, 5 frame(s) burst
Uplink regulator policy:	Mark
Base Station buffering policy:	Not set.
Force Adaptive Data Rate:	Disabled
Asynchronous UL processing:	Not set.

Downlink frame parameters	
Downlink transmission:	Enabled
Acknowledged downlink frame:	Enabled
Rate regulator:	24 frame(s)/day, 2 frame(s) burst
Downlink regulator policy:	Mark
Device status request rate (request/day):	24.0/day
Report Device battery level:	Enabled
Report Device signal margin:	Enabled
Minimal RX1 delay:	Not set.

Network parameters	
Mobility:	Disabled
Network geolocation:	Disabled
Add Base Station metadata information:	Enabled
Class B support:	Disabled
Network Partner access control:	No access control

Routing parameters	
ThingPark X routing:	Enabled
Third Party Application Servers routing:	Enabled
Maximum allowed Application Servers:	99
Third Party Application Server PER information:	Enabled
Third Party Application Server downlink sent indication:	Enabled
ThingPark Kafka routing:	Enabled

Roaming parameters	
Roaming Activation Allowed:	Disabled
Passive Roaming Allowed:	Disabled
Handover Roaming Allowed:	Disabled

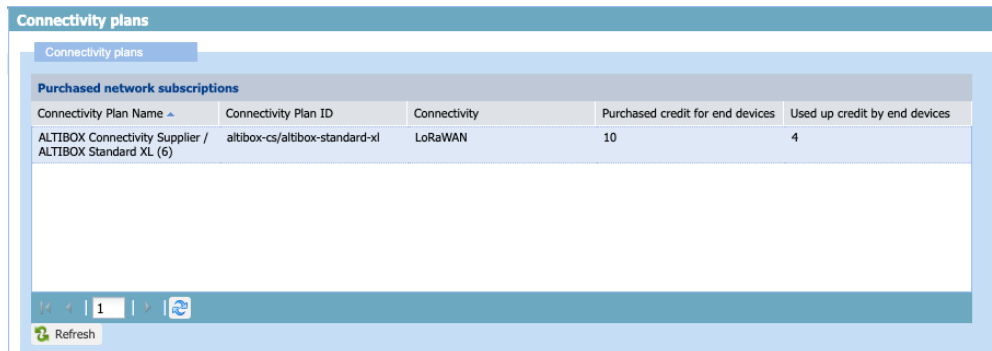
Security	
Hardware Security Module (HSM) protection:	Disabled

Grade-of-Service based ADR parameters	
Minimum Spreading Factor:	Not set.
Maximum Spreading Factor:	Not set.
Force channel mask:	Not set.
Minimum antenna (macro) diversity:	1
Force RX2 Data Rate:	Not set.
Adaptive Data Rate algorithm:	ADR v3: Packet Error Rate (PER) based ADR optimization
Device PER target:	10%
Macro diversity reliability target:	80%
Minimum number of Device uplink transmissions:	1
Maximum number of Device uplink transmissions:	3

Figure 5.10: Complete subscription plan details

As a service provider for consumers, Altibox provides IoT as a Service for which it charges 20 NOK/sensor/month (for minimum 20 sensors). This product also includes Niotix IoT Platform from Digimondo. Altibox provided the four devices, the gateway and access to the ThingPark portal free of cost for this project.



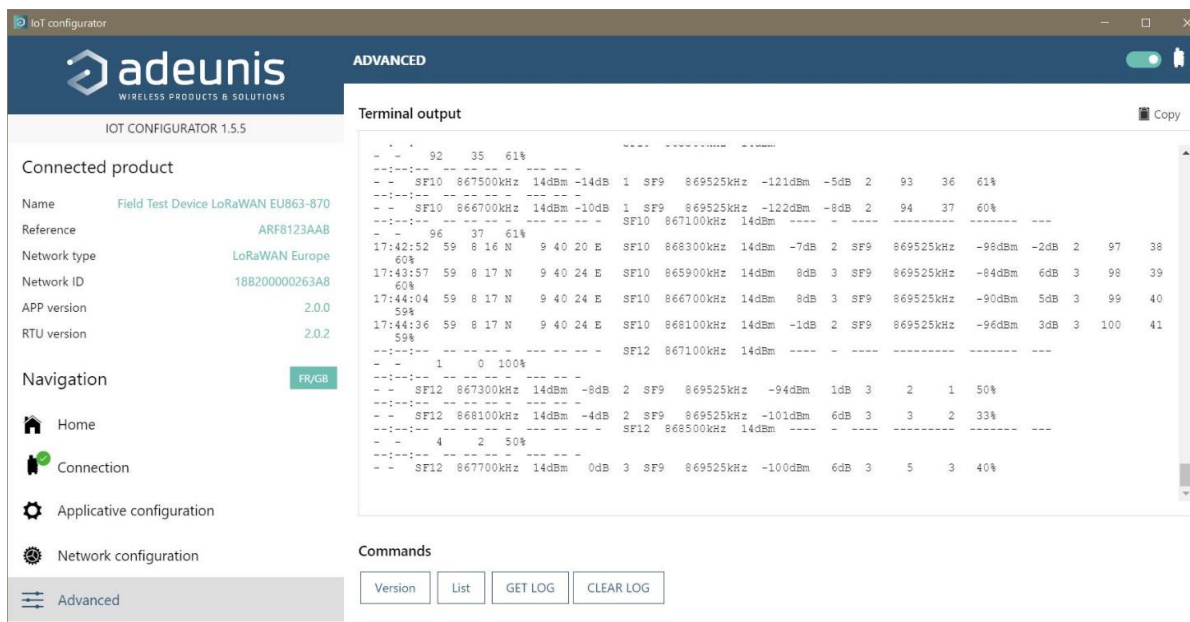
The screenshot shows a web interface titled 'Connectivity plans'. Below the title is a section 'Purchased network subscriptions' containing a table with the following data:

Connectivity Plan Name	Connectivity Plan ID	Connectivity	Purchased credit for end devices	Used up credit by end devices
ALTIBOX Connectivity Supplier / ALTIBOX Standard XL (6)	altibox-cs/altibox-standard-xl	LoRaWAN	10	4

At the bottom of the interface, there is a pagination control showing '1' and a 'Refresh' button.

Figure 5.11: Connectivity plans and subscription details

The LoRaWAN devices can be easily connected to a Windows PC or Android phone through a micro-USB cable. An application called IoT Configurator App could be downloaded from <https://www.adeunis.com/en/device-management-2/sensor-setup/> and installed on PC. The devices could be configured using the application. Additionally, historical parameters from the devices could be captured and saved in PC using the application. Figure 5.12 displays an example of FTD: network tester device connected to the IoT Configurator on Windows 10. The historical data displayed in the Terminal output could be copied and stored locally on PC. On the left of the application, it can be seen that there are options to configure the device.



The screenshot shows the 'IoT configurator' application window. The interface is divided into several sections:

- Header:** 'adeunis WIRELESS PRODUCTS & SOLUTIONS' and 'IOT CONFIGURATOR 1.5.5'.
- Left Panel (Navigation):** Includes 'Connected product' details (Name: Field Test Device LoRaWAN EU863-870, Reference: ARF8123AAB, Network type: LoRaWAN Europe, Network ID: 18B200000263A8, APP version: 2.0.0, RTU version: 2.0.2) and a 'Navigation' menu with options: Home, Connection, Applicative configuration, Network configuration, and Advanced (selected).
- Right Panel (Terminal output):** Displays a log of network activity with columns for time, coordinates, frequency, power, and signal strength. The log shows multiple entries for SF10 and SF12 frequencies.
- Bottom Panel (Commands):** Includes buttons for 'Version', 'List', 'GET LOG', and 'CLEAR LOG'.

Figure 5.12: FTD connected to IoT Configurator 1.5.5

## 5.6 Network Coverage at USN Porsgrunn

The LoRaWAN network coverage survey at USN, Porsgrunn was carried out on 29 September 2022 using the Adeunis FTD network tester device. The results can be seen in Table 5.1 and Figure 5.13. The quality of LoRaWAN radio network was found to be very good when outside but indoors the signal to noise ratio (SNR) was negative most of the time, depicting bad or poor quality of network. The areas near Building C, Process Hall, and Room B237a were also covered in the survey to verify LoRaWAN network coverage for the three installed sensors. The sensors were once stored in a locker outside the gym at USN, Porsgrunn. When they were in the locker, the ThingPark portal couldn't receive any data from the sensors showing that Altibox LoRaWAN infrastructure was not able to receive signals from the sensors while in deep indoor storage. An indoor gateway is generally required for such exceptional cases.

Table 5.1: LoRaWAN coverage at USN Porsgrunn

		Radio Link Quality		UL1 (~867.3 MHz)			DL1 (~869.5 MHz)			
Point of interest	Floor	Uplink	Downlink	SF	Power (dBm)	SNR (dB)	SF	RSSI (dBm)	SNR (dB)	GPS available
Kjolnes Housing room	Ground	Good	Very good	11	14	-7	9	-105	4	Yes
Kjolnes Housing Open area	Ground	Very good	Good	11	14	0	9	-98	-4	Yes
Near SINTEF	Ground	Very good	Very good	11	14	6	9	-92	6	Yes
USN Entrance	Ground	Very good	Very good	11	14	7	9	-99	2	Yes
USN Lobby	Ground	Good	Good	11	14	-6	9	-115	-4	Yes
Library	Ground	Good	Good	11	14	-8	9	-116	-6	Yes
Café Origo	Ground	Bad	Good	11	14	-13	9	-122	-7	Yes
Washroom Café Origo	Ground	Very bad	Very bad	11	14	-13	9	-113	-10	Yes
Outside B-237a	First	Very good	Very good	11	14	1	9	-105	2	Yes
Outside Gym	Basement	Very bad	Very bad	11	14	-4	9	-123	-7	No
Group room A-092a	Basement	Bad	Good	11	14	-10	9	-122	-8	No
Near C building	Ground	Very good	Very good	11	14	8	9	-90	5	Yes
Process Hall	First	Bad	Bad	12	14	2	9	-98	5	Yes

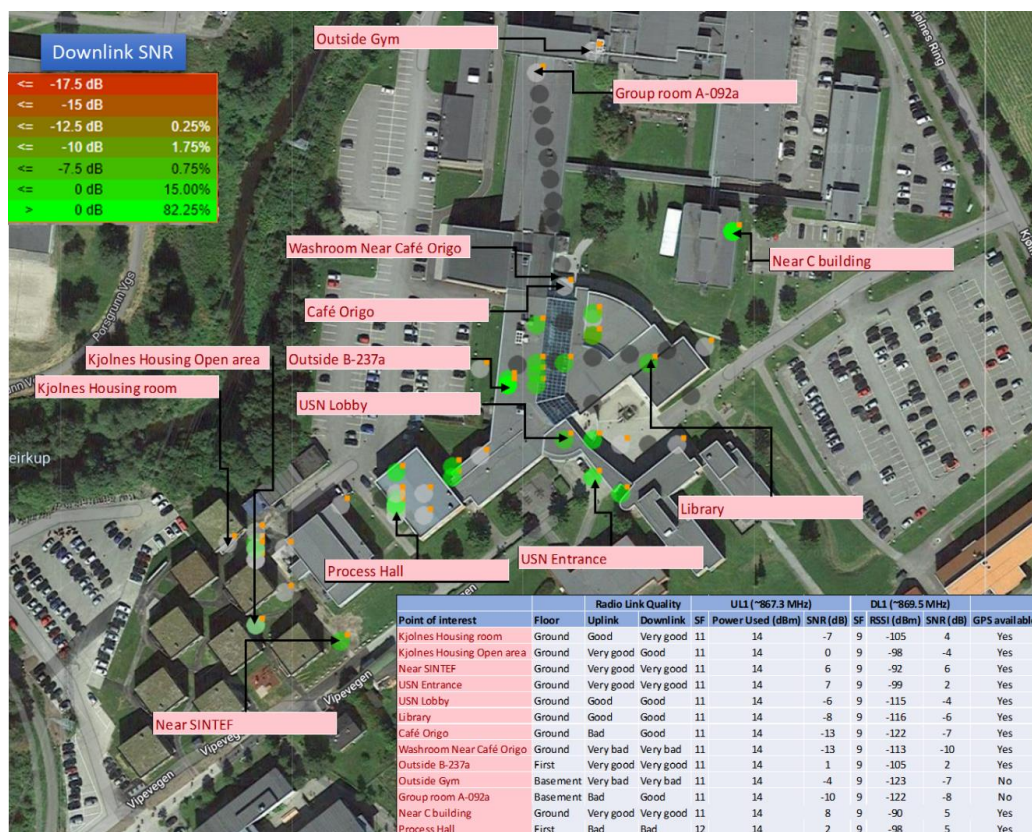


Figure 5.13: Map of LoRaWAN coverage at USN Porsgrunn

## 5.7 Network Coverage at Porsgrunn City

The LoRaWAN network coverage survey for a part of Porsgrunn city was carried out on 30 October 2022 using the Adeunis FTD network tester device. The results can be seen in Figure 5.14. The quality of LoRaWAN radio network was found to be incredibly good in open areas of the city with a positive signal to noise ratio (SNR) for 98% of the survey points.

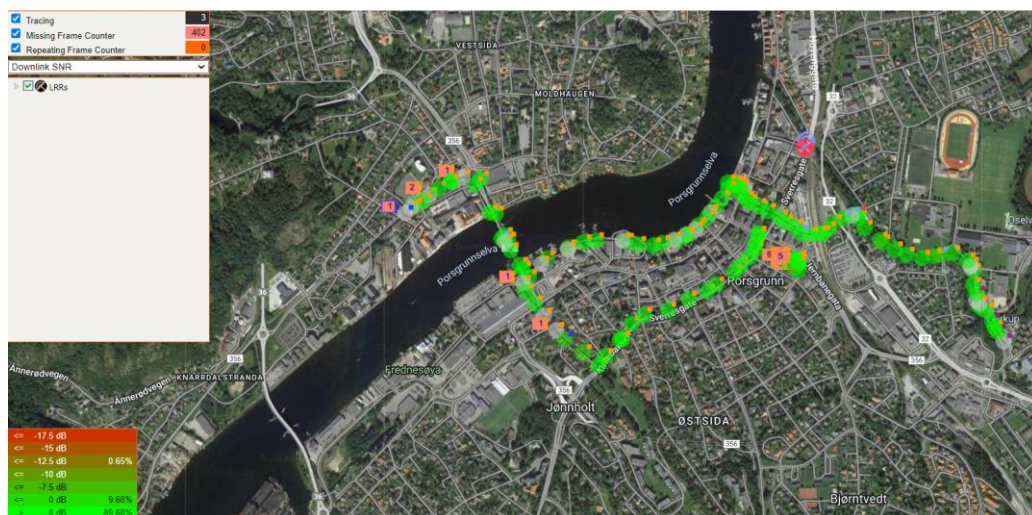


Figure 5.14: LoRaWAN coverage in Porsgrunn city

## 5.8 Quality of Network

The quality of LoRaWAN network is determined by observing the RSSI (Received Signal Strength Indicator) and SNR (Signal to Noise Ratio) at a specific sensor of interest [20]. The sensors provide RSSI and SNR values with the payload signal and these values can be analyzed to find the quality of LoRaWAN network around the sensors. In this section, the RSSI, SNR and Spreading Factor (SF) data from the installed sensors is analyzed.

Comparison of RSSI and SNR for COMFORT sensor is shown in Figure 5.15. The minimum RSSI and SNR were noted to be  $-120$  dBm and  $-13$  dB, respectively. The maximum RSSI and SNR were found to be  $-92$  dBm and  $10.5$  dB respectively. The standard deviation for RSSI was  $4.49$  dBm and for SNR was  $4.44$  dB. The number of data points for this analysis was  $1609$ . All the points were at SF7 depicting fastest data rate ( $\sim 5.5$  Kbits/sec).

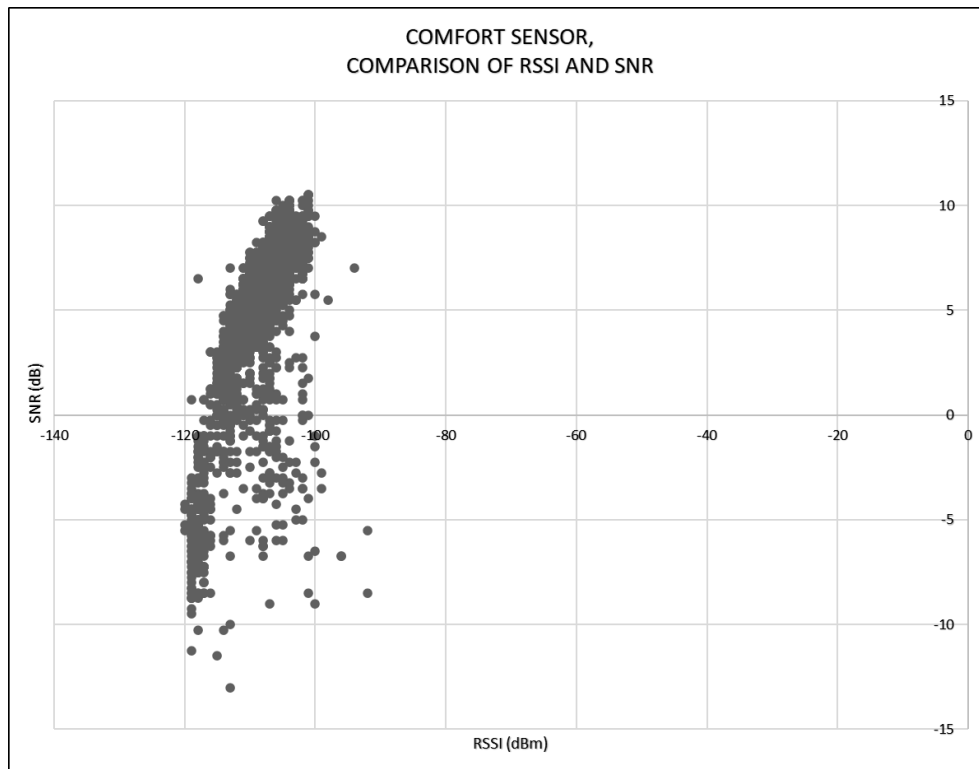


Figure 5.15: COMFORT: RSSI vs SNR

Comparison of RSSI and SNR for TEMP sensor is shown in Figure 5.16. The minimum RSSI and SNR were noted to be  $-121$  dBm and  $-9.5$  dB, respectively. The maximum RSSI and SNR were found to be  $-92$  dBm and  $10$  dB respectively. The standard deviation for RSSI was  $3.91$  dBm and for SNR was  $3.97$  dB. All the points were at SF7 depicting fastest data rate ( $\sim 5.5$  Kbits/sec).

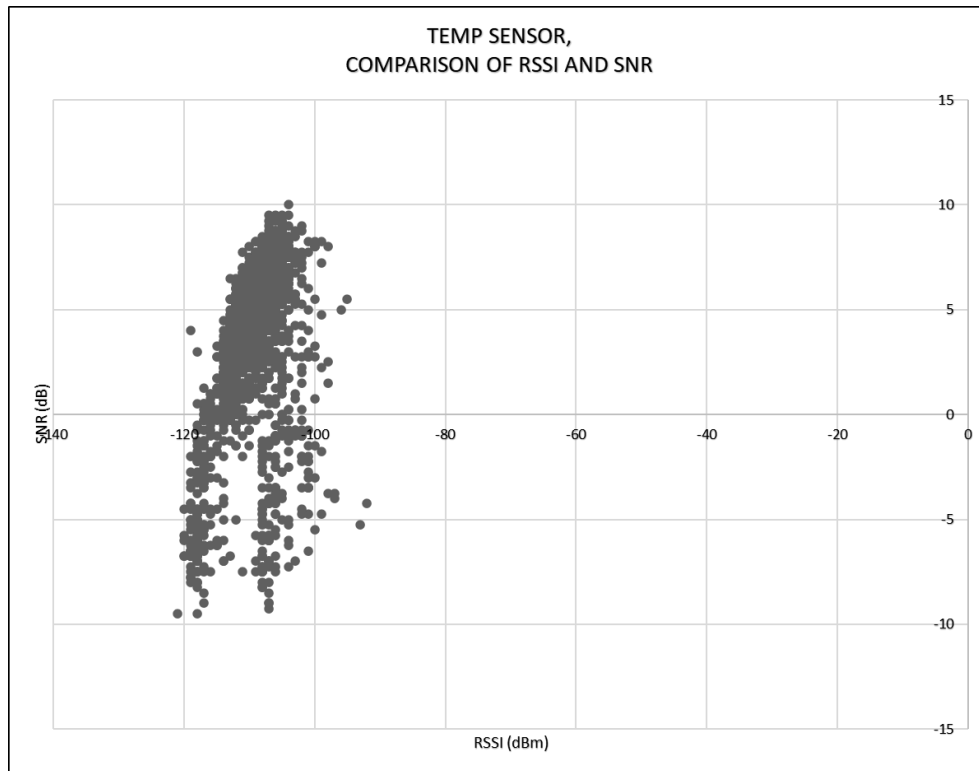


Figure 5.16: TEMP: RSSI vs SNR

Comparison of RSSI and SNR for dry contacts sensor is shown in Figure 5.17. The number of data points was 74. The minimum RSSI and SNR were noted to be  $-120$  dBm and  $-19.25$  dB, respectively. The maximum RSSI and SNR were found to be  $-93$  dBm and  $7.75$  dB respectively. The standard deviation for RSSI was  $8.26$  dBm and for SNR was  $7.77$  dB. The transmission data rate varied from SF7 to SF12 ( $\sim 300$  bits/sec) with the breakup shown in Figure 5.18. A gateway can be considered to be installed near this sensor to improve the transmission data rate and to avoid loss of signal packets.



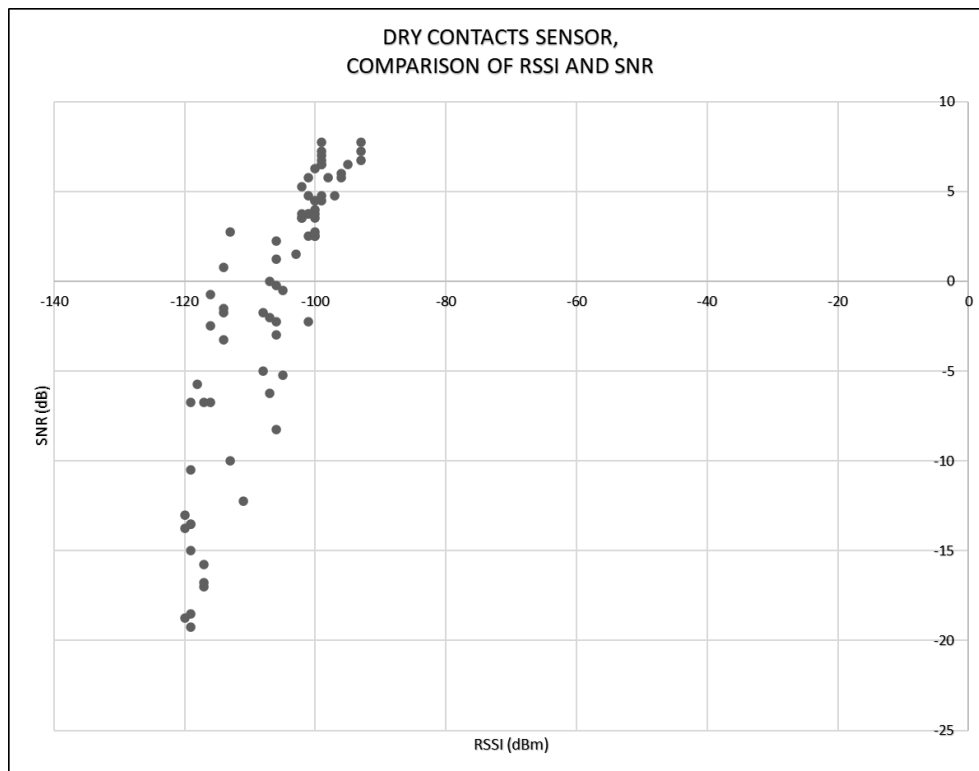


Figure 5.17: Dry contacts: RSSI vs SNR

#### SPREADING FACTOR BREAKUP OF DRY CONTACTS

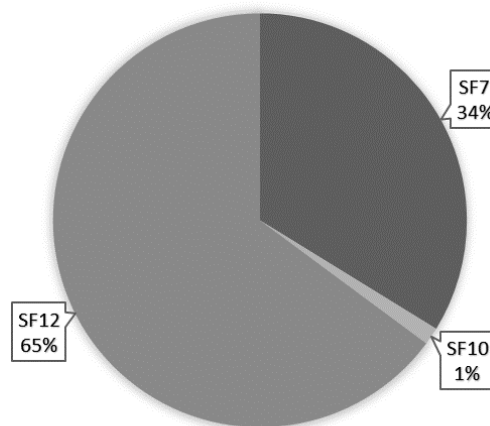


Figure 5.18: Variation of Spreading Factor for Dry Contacts sensor installed in a room B237a

## 5.9 Installation of Gateway

Since the LoRaWAN network coverage in Process Hall was found to be bad according to the network survey, it was decided to install the gateway in the Process Hall to observe the effect of a gateway on the coverage of LoRaWAN network and to improve the indoor LoRaWAN network strength in the Process Hall. The indoor gateway (UFISPACE PICO CELL

"ENTERPRISE") as shown in Figure 5.19 was installed near the COMFORT sensor in the Process Hall. The gateway was based on Semtech Ref Design v1.5 with 8 LoRa channels. The gateway could support class A and C devices. It included a 1.4 dBi internal antenna working on EU868 band.



Figure 5.19: Indoor gateway UFISPACE PICO CELL "ENTERPRISE" installed in Process Hall

Figure 5.20 shows the improvement in the RSSI and SNR of COMFORT sensor after the installation of gateway in the Process Hall. It was found that there was no negative SNR and the quality of the signal improved from the COMFORT sensor and the likelihood of loss of packets was completely removed. To improve the SNR for other sensors, this gateway can be installed at a strategic location such as between Room B237a and Process Hall so that the benefits of the gateway can be shared by two sensors in a cost-effective way.

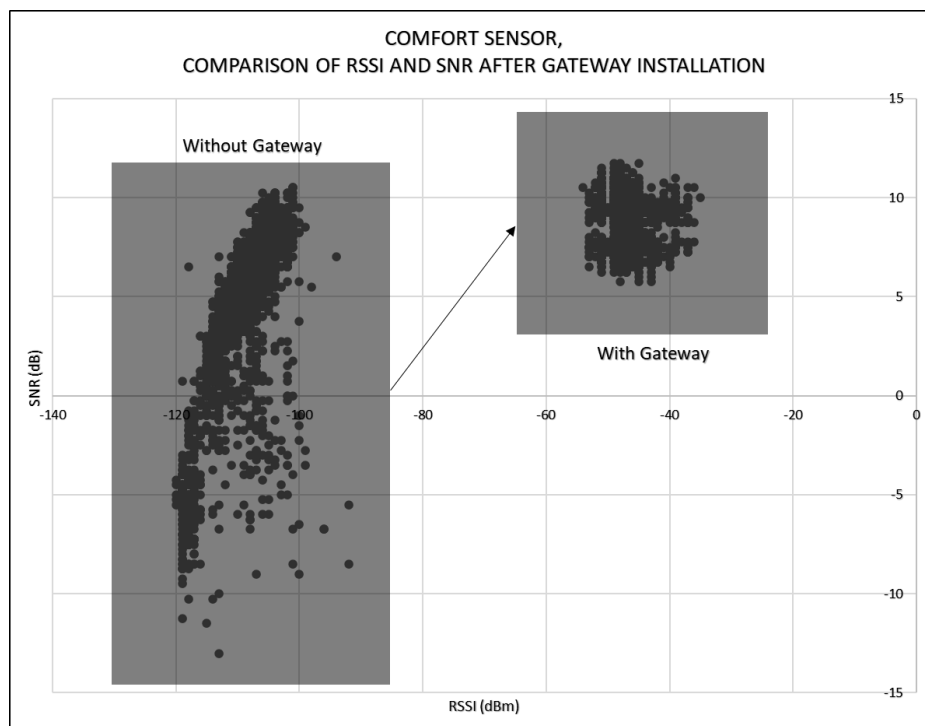


Figure 5.20: Improvement in LoRaWAN downlink coverage for COMFORT after gateway installation



## 6 Datalogging Application

In this section, the development of a datalogging system is discussed. The developed datalogging application involved three stages. (1) A LoRaWAN device is onboarded in the ThingPark portal by Altibox for management and storage of device signals. (2) Device data is transmitted from the device to an open webserver for temporary storage. (3) The data is retrieved from the temporary server, processed and sent to a structured data storage on Dimension Four IoT platform.

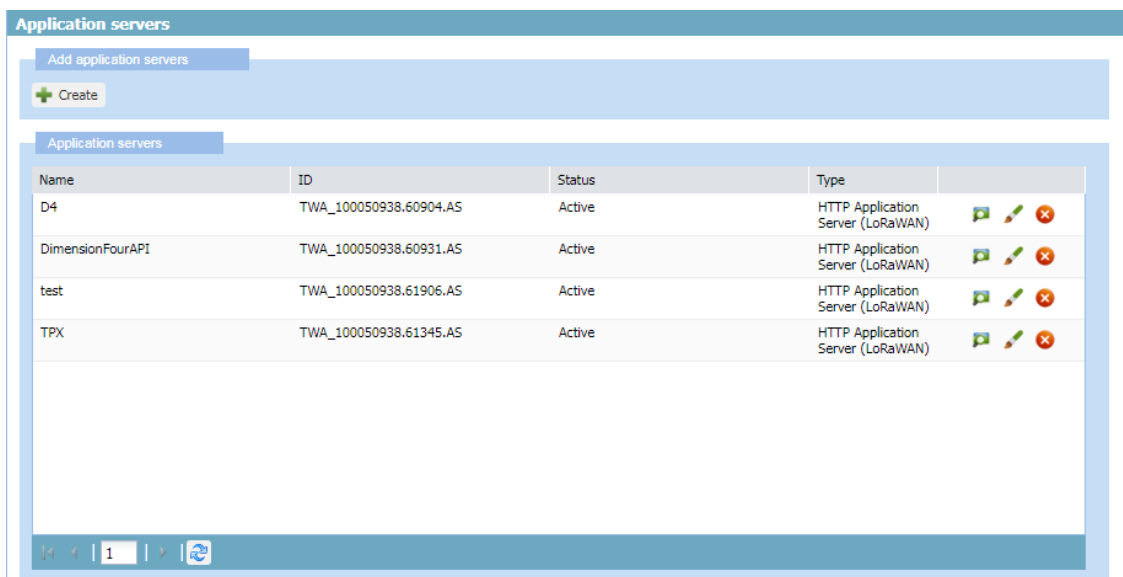
### 6.1 ThingPark Setup

To allow ThingPark to transmit messages from device an Application Server (AS) must be configured. There are three available options:

- HTTP Application Server (LoRaWAN)
- HTTP Application Server (Cellular)
- Kafka Cluster

For this project only HTTP AS (LoRaWAN) is relevant and employed.

A list of already created AS and option to create a new one as shown in Figure 6.1 is available in the tab Application servers in Device Manager in ThingPark.















Name	ID	Status	Type	
D4	TWA_100050938.60904.AS	Active	HTTP Application Server (LoRaWAN)	  
DimensionFourAPI	TWA_100050938.60931.AS	Active	HTTP Application Server (LoRaWAN)	  
test	TWA_100050938.61906.AS	Active	HTTP Application Server (LoRaWAN)	  
TPX	TWA_100050938.61345.AS	Active	HTTP Application Server (LoRaWAN)	  

Figure 6.1: ThingPark: Application Server list.

When creating an AS, the required fields are Name and Content Type. Available types of content are XML, JSON and JSON untyped. A route should be added, and URLs are required to be specified in the Destination field. Additional headers required to be included in the datagrams may be added in HTTP custom headers field as shown in Figure 6.2.

Delete
Save
Cancel
Close

Application server

Application server

Name: \*test\_AS
ID: TWA\_100050938.62206.AS
Content Type: \*XML
Type: HTTP Application Server (LoRaWAN)
Status: \*Active

HTTP custom headers

Name	Value
x-security-token	123

Add
Delete

Uplink/downlink security

Status: Inactive
Max timestamp deviation: -

Activate

Route

Source ports: \*
Routing strategy: \*

Destinations

Destination
https://url.com/

Edit
Add
Delete
Up
Down

Add a route

Add an additional route to this application server.

Add

Figure 6.2: ThingPark: Application Server creation.

As an alternative to the direct specification of AS in Device Manger, ThingPark provides a ThingPark X portal that gives access to multiple types of destination, called Connection in ThingPark X as displayed in Figure 6.3.

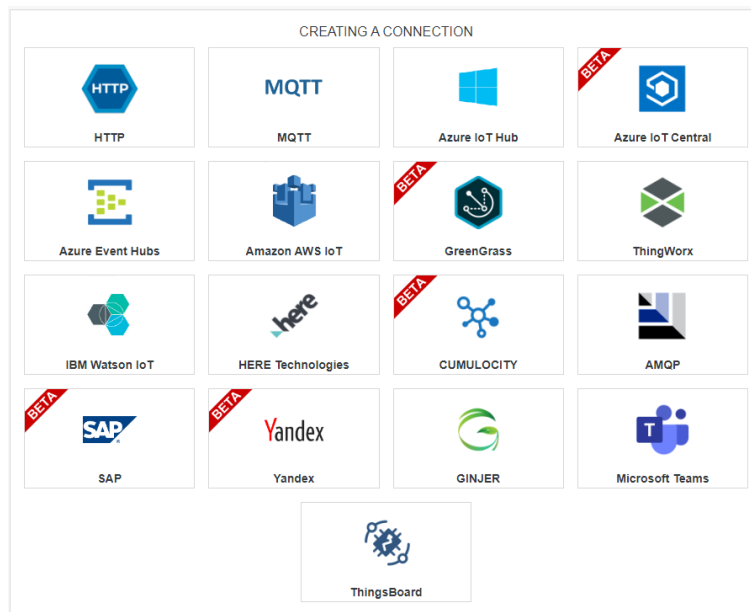


Figure 6.3: ThingPark X: available connections

One may choose a desired destination type and set it up by providing necessary information. Figure 6.4 shows an example of setting up an HTTP connection as seen in the Device Manager of ThingPark.

Figure 6.4: ThingPark X: HTTP connection setup

After creation of an AS, all the created connections appear in a list of connections (Figure 6.5), where status and general statistics for the connection can be observed. It is also possible to remove or edit the connections. The overview shows if a connection is active and the number of devices using it.










+ ADD CONNECTION							
	Connection Name	Last Restart	Active Devices (last 1h/24h)	Uplinks (last 1h/24h)	Downlinks (last 1h/24h)	State	
	Network Tester	20 days ago	2 / 3	12 / 298	0 / 0	OPENED	
	MQTT0	4 days ago	2 / 3	24 / 573	0 / 0	OPENED	

Figure 6.5: ThingPark X: HTTP connection list.

After connections are created, they need to be added to a flow to be active. Flexibility of ThingPark X follows from the fact that several connections may be added to one flow and used for data transmission to several types of destination simultaneously. During creation of a flow (Figure 6.6) a desired connection may be selected from the previously created connections and used as a combination. All created flows appear in a list which also displays types of connections.

+ ADD FLOW			
Flow Name	Connections	Matcher Type	Driver
Flow0		 MATCHED BY KEYS	<> DRIVER AUTOMATIC
Flow1	 	 MATCHED BY TAGS	<> DRIVER AUTOMATIC

CREATE A NEW FLOW

1

2

3

4

5

Name and description

Rules


Driver

Uplink Transformations


Connections

Select the connections you want to associate with your flow

+ ADD CONNECTION

 Network Tester

Selected

 MQTT0

Selected

Cancel

Back

Finish

Figure 6.6: ThingPark X: flow creation and flow list

A particular flow may also be configured to be used with a particular registered device by specifying its DevEUI (Figure 6.7).

The screenshot shows the 'FLOW0' configuration page in the ThingPark X dashboard. The page is titled 'FLOW0' and 'MQTT'. It features a breadcrumb 'FLOWS > FLOW0' and a trash icon. Below the title, there are two tabs: 'MATCHED BY KEYS' (selected) and 'DRIVER AUTOMATIC'. A note states: 'THIS FLOW WILL SELECT DEVICES THAT MATCH ONE OF THE KEYS LISTED'. The interface is divided into three main sections: 'MATCHED KEYS', 'CONNECTIONS', and 'UPLINK TRANSFORMATIONS'. The 'MATCHED KEYS' section contains two input fields with the values 'lora:0018b21000004540' and 'lora:0018b21000003bbf', each with a 'MANAGE' button. The 'CONNECTIONS' section shows a single connection named 'MQTT0' with a 'View details' link and a 'MANAGE' button. The 'UPLINK TRANSFORMATIONS' section displays a message 'There are no applied operations.' and a 'MANAGE' button.

FLows > FLOW0

FLOW0

MQTT

MATCHED BY KEYS <> DRIVER AUTOMATIC

THIS FLOW WILL SELECT DEVICES THAT MATCH ONE OF THE KEYS LISTED

MATCHED KEYS	CONNECTIONS
<div>lora:0018b21000004540</div> <div>lora:0018b21000003bbf</div> <div>MANAGE</div>	<div>MQTT0 <a href="#">View details</a></div> <div>MANAGE</div>

UPLINK TRANSFORMATIONS

There are no applied operations.

MANAGE

Figure 6.7: ThingPark X: Flow configuration.

Dashboard in ThingPark X gives access to the tools for creation of flows and connections and displays number and status of the existing ones (Figure 6.8).

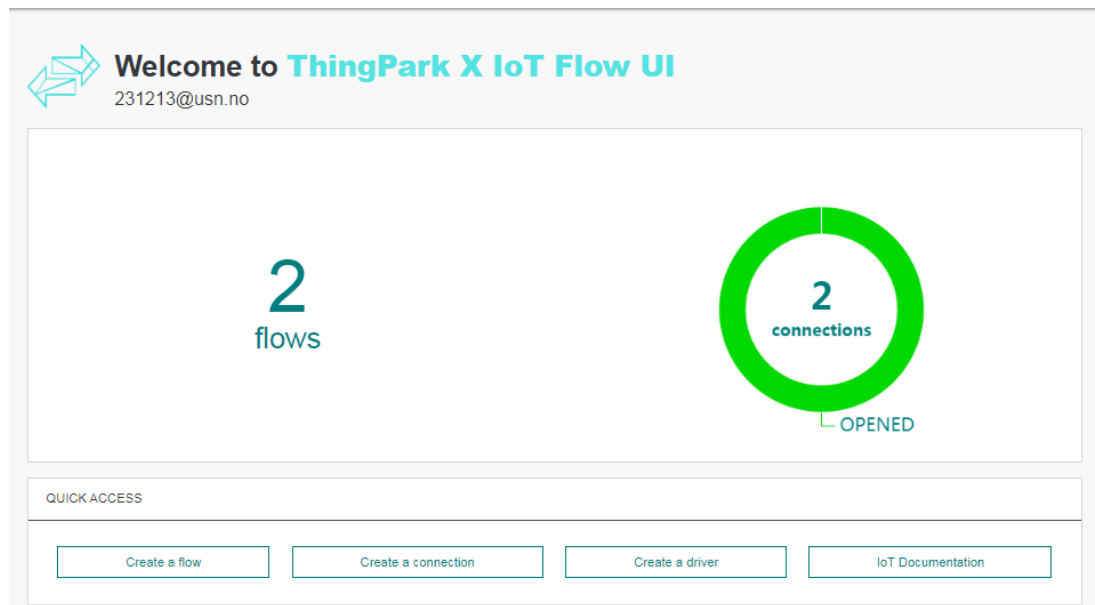


Figure 6.8: ThingPark X: Dashboard.

If any of connections are closed that status would be shown on the Dashboard.

To establish communication between devices and AS, a Routing profile must be created on the ThingPark portal. All created and available routes are listed in AS routing profiles as given in Figure 6.9 and can be modified. A new routing profile can be of type LoRaWAN or Cellular as per the connectivity plan of the user.










AS routing profiles				
Add AS routing profiles				
+ Create				
AS routing profiles				
Name	ID	Type	Is default	
D4	TWA_100050938.51841	LoRaWAN	False	  
DimensionFourAPI	TWA_100050938.51865	LoRaWAN	False	  
TPX	TWA_100050938.52039	LoRaWAN	True	  

Figure 6.9: ThingPark: Routing profile list.

After choosing a name, destinations must be specified chosen from the previously created destinations (Figure 6.10). They can be:

- Supplier application server
- Local application server

- ThingPark X

The first option is provided by a supplier; the second option gives access to the AS created on this Device Manager page. And the last one will automatically connect to the flows and will use the configured devices.

For one profile it is possible to use several AS (Figure 6.10) that will deliver the same message to different destinations.

**AS routing profile [Read only]** Close

**AS routing profile**

Name: \*

ID: TWA\_100050938.52039

Type: LoRaWAN

Is default: ☒ (This application routing profile will be used by default)

**Destinations**

Type	Destination	Status
ThingPark X	ThingPark X IOT-FLOW	Active
Local application server	D4	Active

**Status**

Last modification: 30.9.2022, 11:38:22

Updated by: Vitaly Dekhtyarev

Figure 6.10: ThingPark: Routing profile creation.

An example of a message transmitted by ThingPark is shown in Figure 6.11. As can be seen, the message contains multiple fields related to the condition of the instrument at the time of transmission. However, the content varies depending on the type of instrument and time. In this case there is additional information with battery charge level, this information is sent periodically or after a special request from gateway. Values for coordinates, LrrLAT and LrrLON, will depend on settings of the instrument and if it has GPS installed. The values that are of special interest are DevEUI, which allows to identify the device, mId, which identifies the type of the device, and payload\_hex. The latter contains specification of the data frame, is it a periodic data frame with measurements or data frame for maintenance, and measurement. The payload is to be decoded by other components of datalogging application. It may also be transmitted already decoded if the instruments have an integrated ThingPark decoder.

```

{
  "DevEUI_uplink": {
    "Time": "2022-11-10T11:59:55.694+00:00",
    "DevEUI": "0018B22000001FD2",
    "FPort": 1,
    "FCntUp": 271,
    "ADRbit": 1,
    "MType": 2,
    "FCntDn": 151,
    "payload_hex": "40e0000100000000000000e",
    "mic_hex": "69ea49a9",
    "Lrcid": "00000201",
    "LrrRSSI": -103,
    "LrrSNR": 4.25,
    "LrrESP": -104.385674,
    "SpFact": 7,
    "SubBand": "G1",
    "Channel": "LC1",
    "Lrrid": "C0001DFD",
    "Late": 0,
    "LrrLAT": 0,
    "LrrLON": 0,
    "Lrrs": {
      "Lrr": [
        {
          "Lrrid": "C0001DFD",
          "Chain": 0,
          "LrrRSSI": -103,
          "LrrSNR": 4.25,
          "LrrESP": -104.385674
        }
      ]
    },
    "DevLrrCnt": 1,
    "CustomerID": "100050938",
    "CustomerData": {
      "alr": {
        "pro": "ADRF/SENSOR",
        "ver": "1"
      }
    },
    "ModelCfg": "1:New",
    "DriverCfg": {
      "mod": {
        "pId": "adeunis",
        "mId": "dry-contacts",
        "ver": "2"
      },
      "app": {
        "pId": "adeunis",
        "mId": "dry-contacts",
        "ver": "2"
      }
    },
    "BatteryLevel": 254,
    "BatteryTime": "2022-11-10T11:59:55.694+00:00",
    "Margin": 6,
    "InstantPER": 0,
    "MeanPER": 0,
    "DevAddr": "FC004E1F",
    "TxPower": 10,
    "NbTrans": 1,
    "Frequency": 868.1,
    "DynamicClass": "A"
  }
}

```

Figure 6.11: ThingPark: message example, with battery level



Transmitted messages may also be examined in Wireless-Logger tool in ThingPark (Figure 6.12). The tool provides functionality to analyze the status of an instrument and to see a decoded payload. While the appearance of the message is different from the one in JSON format, such a tool is extremely important for debugging of a custom decoder.

The screenshot displays the ThingPark Wireless-Logger interface. At the top, there are filters for DevAddr, DevEUI, LRR Id, and LRC Id, along with fields for From, To, Packet Type, Decoder (set to Automatic), and Auto Reload. Below these are buttons for Refresh, Export size (set to 100), Export, and Map.

The 'Last packets' section shows a table with columns: UTC Timestamp, Local Timestamp, DevAddr, DevEUI, FPort, FCnt, NFCnt, AFCnt, RSSI, and SNR. The first packet is selected, showing details for a 'mac data' packet received at 2022-11-16 14:23:40.595.

The decoded data for the selected packet is as follows:

```

Mtype: UnconfirmedDataUp
Flags: AD: 1, ADACKReq: 0, ACK: 0
Mac (hex): 06fe07
MAC.Command.DevStatusAns
MAC.DevStatusAns.Battery: 254
MAC.DevStatusAns.Margin: 7
Data (hex): 4c2000d01e [not encrypted]
Driver metadata: model: adeunis:comfort:1, application: adeunis:comfort:1
Decoded data using driverId: actility:adeunis-comfort:1
{
  "frameCounter": 1,
  "configurationInconsistency": false,
  "hardwareError": false,
  "lowBattery": false,
  "configurationSuccessful": false,
  "type": "TEMPERATURE_AND_HUMIDITY_READINGS",
  "register": {
    "temperatureAndHumidityReadings": [
      {
        "offset": 0,
        "temperature": 20.8,
        "humidity": 30
      }
    ]
  }
}
Data size (bytes): 5
AirTime (s): 0.056576
  
```

Below the decoded data, there is a table for LRR, RSSI, SNR, ESP, CHAINS timestamp (GPS RADIO), ISM Band, RF Region, GWID, GWToken, DLAllowed, ForeignOperatorNetID, and ForeignOperatorNSID. The table shows data for two channels: C0001DPD and 6908PDP5.

Further down, there is a section for Device [Lat (solv): - Lat: - Long (solv): - Long: - Loc radius: - Loc time: - Alt: - Alt radius: - Acc: - North Velocity: - East Velocity: -]. The Reporting Status is 'On time'. The ISM Band is 'EU 863-870MHz', the RF Region is 'EU868\_8channels', the AS ID is 'IOT\_FLOW,TWA\_100050938.60904.AS', the Frequency (MHz) is '867.1', and the Current class is 'A'.

At the bottom, there is a table for AS ID, Status, and Transmission errors. The table shows data for IOT\_FLOW and TWA\_100050938.60904.AS.

Figure 6.12: ThingPark: Wireless-Logger.

## 6.2 Webhook

A simple HTTP AS may be configured with a webhook. This type of webpage provides a trigger for callbacks initiated by a user. In this project a public web server (webhook.site) is used which accepts defined types of HTTP methods. Any request to the webhook will be processed by the server and content of the request will be displayed (Figure 6.13).

The screenshot shows a web application interface for managing webhooks. On the left, there's a sidebar with 'REQUESTS (1/500)' and 'Oldest First' sorting. Below it is a search bar. The main area displays a list of requests, with one highlighted: a POST request from IP 52.16.83.187 on 09/11/2022 at 23:42:51. To the right, the 'Request Details' panel is expanded, showing the full URL, host, date, size, and ID. Below this, the 'Query strings' section lists parameters: LmDevEui (0018B21000003BBF), LmFPort (1), and LmInfos (TWA\_100050938.60904.AS-1-1029224850).

Figure 6.13: Webhook: interface.

The presented webpage allows a free-to-use service that is open to the public. However, the URL of each page contains UUID that must be known to a user to access a specific webpage. When performing a request from ThingPark the same URL with the UUID must be specified as a destination, no custom headers are required.

Content of the request is displayed as it is and can be examined visually (Figure 6.14). The webservice allows GET method to retrieve requests and DELETE to remove ones.

#### Raw Content

```
{
  "DevEUI_uplink": {
    "Time": "2022-11-08T20:22:38.145+00:00",
    "DevEUI": "0018B21000003BBF",
    "FPort": 1,
    "FCntUp": 8157,
    "ADRbit": 1,
    "MType": 2,
    "FCntDn": 1695,
    "payload_hex": "4ca000d42e",
  }
}
```

Figure 6.14: Webhook: part of POST request content display.

There is also an API available, using this user may specify how many requests and which to retrieve or delete [21].

## 6.3 Python Application

Python is a high-level interpreted general purpose programming language [22]. It has an advantage compared to other compiled languages such as C# that scripts may be easily modified or updated in a text editor and run immediately. The environment used by the language is preconfigured by the user for a particular machine before any code is executed. Therefore, there is no need to use separate settings for each project. There are also multiple extensions, modules, available in the official repository for the language, these may be easily integrated in each script after installation that is performed, in most cases, by the language API.

The overall application structure is presented in Figure 6.15 and flow chart in Figure 6.16.

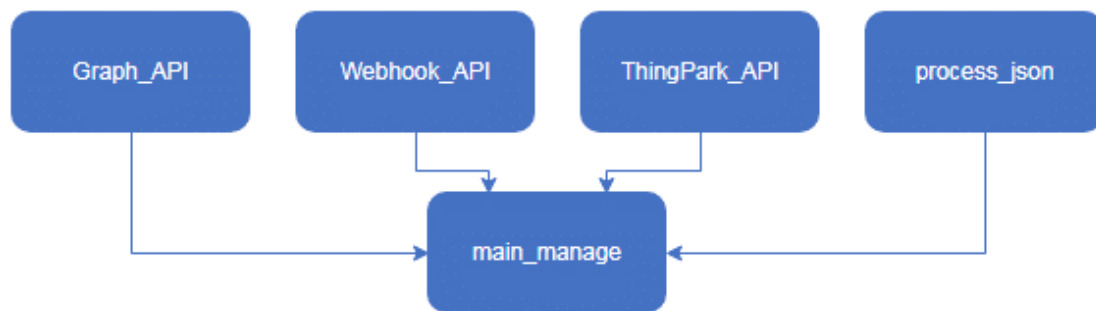


Figure 6.15: Python application: program structure.

Initialization of the main program starts with reading of necessary configurations, URL, and security keys from the provided configuration file.

Webhook\_API module provides functionality to retrieve and remove data from internet resources. The webhook.site is accessed to download data. If no data is available, then the process exits. Otherwise, all available requests stored on the website are downloaded.

In case data is available the application makes a request to Dimension Four IoT platform service and retrieves information about all available instruments registered for the available tenant using Graph\_API module.

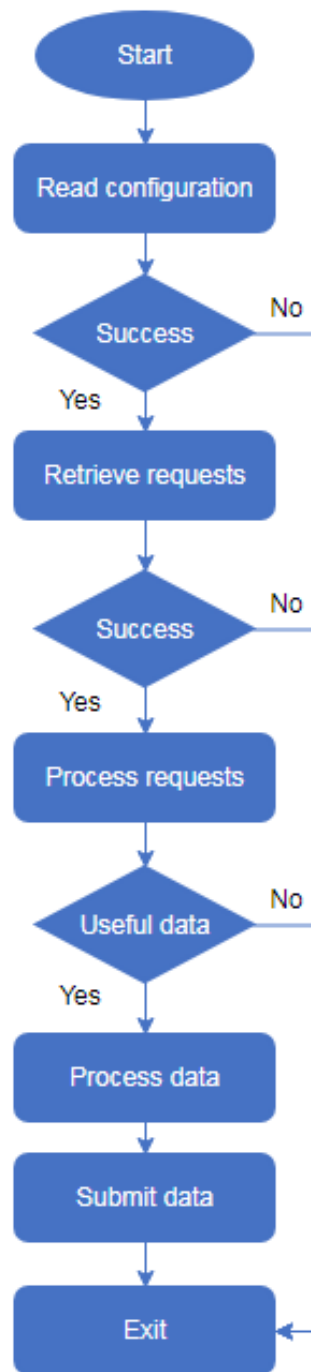


Figure 6.16: Python application: flow chart.

Then the main program parses the content of retrieved requests and assigns data to the corresponding instruments if they exist. In the end all data is registered in Dimension Four IoT platform in the associated points.

The application is based on usage of two main super classes: Sensor and Instrument (Figure 6.17).

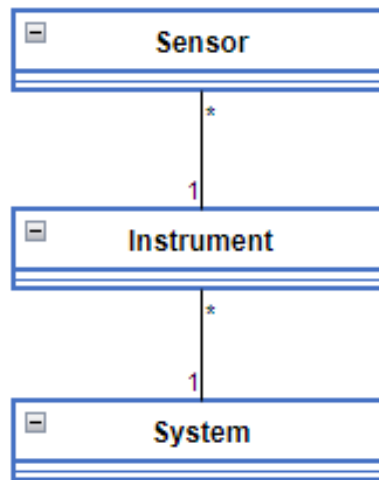


Figure 6.17: Python application: domain model.

It is considered from the description of LoRaWAN instruments that they may use several channels, or sensors, for performing measurements. Therefore, there may be associated multiple sensors that have similar attributes (Figure 6.18). One instrument may operate sensors of several types. One message with data submitted to webhook contains data for only one instrument.

```

class Sensor():
    def __init__(self):
        self.unit = None
        self.value = None
        self.type = None
        self.metadata = {}

class Sensor():
    def __init__(self):
        self.unit = None
        self.value = None
        self.type = None
        self.metadata = {}
  
```

Figure 6.18: Python application: Sensor class

Metadata assigned to a sensor may contain channel number which the sensor is connected to or other describing information.

The Instrument class (Figure 6.19) is responsible for extracting necessary metadata from the message and creation of Sensor, that are then included into a list of sensors for that instrument, together with assigning measured value to each sensor.

```

class Instrument():
    def __init__(self, message):
        self.message = message['DevEUI_uplink']
        self.DevEUI = None
        self.name = None
        self.payload = None
        self.appfl2 = 16
        self.appfl1 = 8
        if message:
            self.payload = self.message['payload_hex']

        self.msg_type = None
        self.periodic_df = None
        self.metadata = {}

        #Mandatory metadata
        self.metadata['Signal strength'] = ''
        self.metadata['SNR'] = ''
        self.metadata['point_id'] = ''
        self.metadata['imageUrl'] = ''
        self.metadata['timestamp'] = ''

```

Figure 6.19: Python application: Instrument class with attributes

From the superclass, specific classes are to be created. Each of them overrides super class methods and assigns proper metadata and values to sensors.

Methods that are not overridden are listed in Figure 6.20. Among those is `def_type()`, which examines if the provided message is a periodic data frame that contains actual measurement and not a maintenance data frame. To distinguish among the types, the instrument must decode a part of payload. The code varies between different instruments; therefore, it is assigned to `periodic_df` attribute depending on the type of the instrument. `bat_level` converts battery level if provided in message to number. Attributes `appfl1` and `appfl2` may be used by some of instruments to specify how many channels are in use.

After all instruments have been properly created, the main program prepares dictionaries from data from instruments and sensors to be sent to Dimension Four IoT platform. These dictionaries are then used by `Graph_API` module to register values with function `signal_create()`.

In the end, all retrieved requests must be removed from webhook, so they are not downloaded again and do not occupy memory on the server. Removal uses UUID assigned to each request registered by the webhook.

```

def fill_meta(self):
    if 'DevEUI' in self.message:
        self.metadata['DevEUI'] = self.message['DevEUI']
    if 'Time' in self.message:
        self.metadata['timestamp'] = self.message['Time']
    if 'LrrRSSI' in self.message:
        self.metadata['Signal strength'] =
            self.message['LrrRSSI']
    if 'LrrSNR' in self.message:
        self.metadata['SNR'] = self.message['LrrSNR']
    if 'BatteryLevel' in self.message:
        self.metadata['BatteryLevel'] =
            self.bat_level(int(self.message['BatteryLevel']))
    if 'BatteryTime' in self.message:
        self.metadata['BatteryTime'] =
            (self.message['BatteryTime'])
    if 'Frequency' in self.message:
        self.metadata['Frequency'] =
            (self.message['Frequency'])

def bat_level(self, value):
    return f"{{(value - 1)/(253)*100: .2f}}"

def def_type(self):
    if not self.payload:
        return None

    bfh = int(self.payload[:2], 16)
    if (bfh & self.periodic_df) == self.periodic_df:
        self.msg_type = "Periodic data frame"

```

Figure 6.20: Python application: common methods

## 6.4 Alternative Application Server

As an alternative to the presented webhook solution, one may implement other types of connections from the list given in Figure 6.3 such as a more secure custom webhook.

A custom webhook may be developed on a private server which may accept only particular HTTP methods. In addition, such server may use a specific interface or API with custom headers to limit type of messages to display or accept and to manipulate. The drawback of such system is that it must be developed from scratch.

One may also use other existing webhooks, such as MS Teams or Dimension Four IoT platform. Then it is necessary to pay attention to the facilities required for a datalogging system and the ones provided by these webhooks.

Among the critical functionalities to be considered when such webhooks are in focus are:

- Storage capacity
- API methods
- Security methods.

Another AS that is also widely used in relation with IoT is MQTT. MQTT connection has been set up under the current project, but not used in the definitive version of the system. This AS has been configured with a free broker provided by HiveMQ. The datalogging application, as a subscriber, must then regularly read latest messages from the broker.

The downside of the approach is that reading of the data must be synchronized with data publishing, so that the subscriber does not waste system resources. In the case with the equipment used for the project, instruments submitted messages with varying time intervals, and messages containing status information were mixed with the messages containing only data. That may cause more storage usage on broker by unimportant information that is not of interest, or useful messages will be overwritten by the status messages.

## 6.5 Message Transformation

ThingPark X offers facility to process messages from LoRa devices directly on the portal and send them to a destination. Such approach could help to avoid development of a data processing software for the system.

Transformation is implemented using a JMESPath query language. The language parses a provided JSON object and gives a result according to a user-defined pattern. As an example, below is presented a pattern to form another JSON object from the one as shown in Figure 6.21.

```
{
  DevEUI: DevEUI_uplink.DevEUI
  payload: DevEUI_uplink.payload_hex
  sensor_type: DevEUI_uplink.DriverCfg.mod.mId
  BatteryLevel: DevEUI_uplink.BatteryLevel
}
```

Figure 6.21: JMESPath pattern

This pattern will result in the object shown in the Figure 6.22.

```
{
  "DevEUI": "0018B21000003BBF",
  "payload": "4c6000d632",
  "sensor_type": "comfort",
  "BatteryLevel": null
}
```

Figure 6.22: Resulted JSON object

This technique relies on a rather static input message format. Any key defined in the pattern that does not find a value in the incoming object will be set to «null», as with BatteryLevel. While in general it is not be a problem, but for some cases the value may confusing. If



instruments may provide several measurements, such possibility must also be taken in account and fields for several sensor types should be included. But then instruments that provides only one measurement will have multiple fields set to «null». If the message does not have the key DevEUI\_uplink, then all fields will be set to «null». That may cause an error while storing data in Dimension Four IoT platform or just store data that wastes available storage.

Moreover, JMESPath does not allow to process values. Therefore, decoding of the payload should be done either by ThingPark or other part of the system.

While this technique may be useful in some cases, it must be used for simple non-varying messages. Considering the current project, JMESPath cannot produce a result of any form and therefore cannot be implemented to prepare queries for Dimension Four IoT platform.

## 7 Monitoring Application

A monitoring application that presents the data now stored in Dimension Four IoT platform has been created and deployed to Azure as a web application making it available to the public from anywhere. Also, an administration (admin) part has been created and included in the monitoring application, providing detailed view of the data when logged in and enabling full CRUD operations on the data. In addition to the tasks listed in the task description included in Appendix A, the main requirements are listed below.

- Deployment to cloud service and availability from anywhere
- The main display updates automatically when new data is available
- Dimension Four IoT platform used as data backend
- Create, Read, Update and Delete operations on the data stored in Dimension Four IoT platform using GraphQL API
- Create, Update and Delete operations only available when logged in with valid credentials
- Handling of any number of spaces, sub spaces, points and signals in the data structure
- Handling of any number of different units used in a point
- Historical data can be plotted by selecting point, start date and duration
- Historical data page has a button to export the plot to file

### 7.1 Main Technical Stack

#### 7.1.1 ASP.NET Core Blazor Server

ASP.NET Core Blazor Server is a cross platform and open-source framework used to build interactive web applications. ASP.NET Core 6 is the latest evolution of Microsoft's popular ASP.NET web framework. It enables the developer to use C# instead of JavaScript, and the existing .NET ecosystem and libraries can be leveraged.

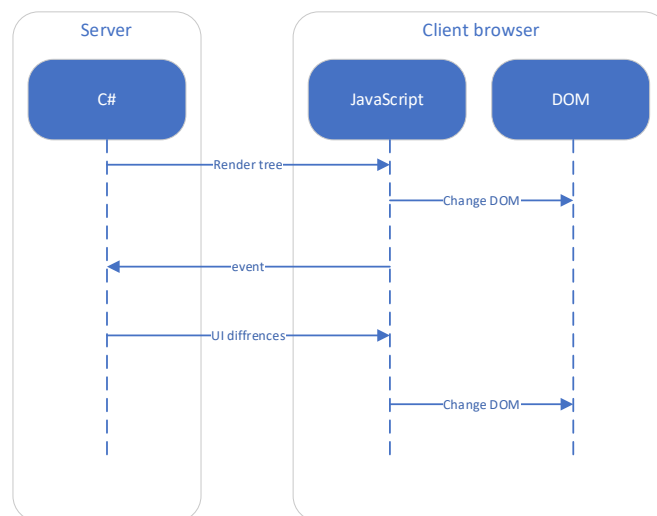


Figure 7.1: Blazor Server Side

Application logic can be shared between the client and the server. With Server side Blazor the render tree is built on the server and gets serialized to the browser using SignalR. This process is depicted in Figure 7.1.

### 7.1.2 SignalR

SignalR is a real-time, cross-browser, and open-source two-way Remote Procedure Calls (RPC) protocol. ASP.NET Core SignalR enables web applications to maintain a persistent connection making event notification functionality available to developers. This makes it possible for new web applications that require high-frequency updates from the server, such as real-time gaming or monitoring. Each side in the connection can invoke methods on the other side of the connection. SignalR uses WebSockets when available and will gracefully fall back to other techniques like Ajax long polling if WebSockets is not available; the application code remains the same whatever the fallback. As depicted in Figure 7.2 in the next section, SignalR makes Blazor Server work.

## 7.2 Architecture of the Monitoring Application

The architecture of the monitoring application is depicted in Figure 7.2 clients can connect to the server and use the monitoring application using a standard web browser. The monitoring part of the application has been created to be used on any modern computer, mobile or tablet.

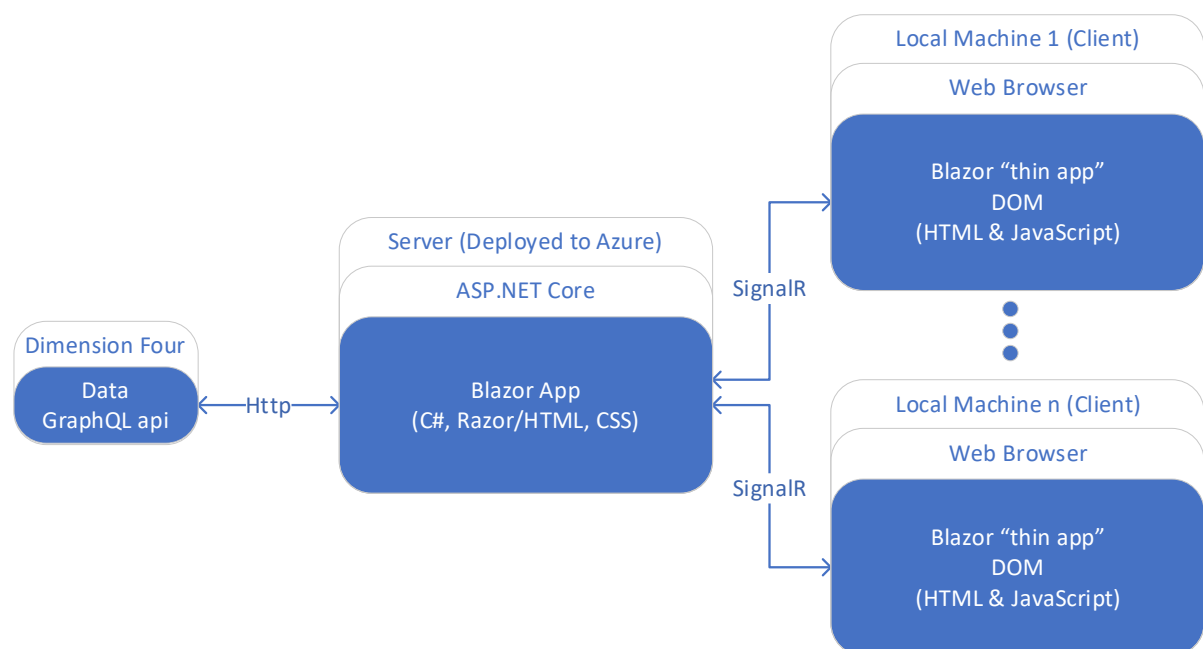


Figure 7.2: Architecture of the web monitoring app.

The solution created in Visual Studio 2022 contains three projects, one project containing the user interface, one project containing a library with pure C# functions and the last project

contains unit tests for functions implemented. The code is open source and can be found at the following link: <https://github.com/LoRaWAN-PT/MonitoringApp>

## 7.3 Access to Data in Dimension Four IoT Platform

### 7.3.1 HTTP

To access the data in Dimension Four IoT platform any HTTP client can be used [23], all GraphQL queries are posted to the same endpoint and authorization is done by providing two headers that include credentials. In C# Blazor Server this can be implemented using a HTTP client factory in program.cs file (on GitHub repository) to register a data service requesting an interface. This makes the service available for dependency injection [24] anywhere in the application, this has the added benefit of loose coupling to the concrete implementation. The actual implementation in C# where the base address and header details are retrieved from a setting (appsettings.json) file (on GitHub repository) is shown in Figure 7.3.

```
builder.Services.AddHttpClient<IDataAccess, GraphQLD4DataAccess>
    (dimensionFourDataService =>
    {
        dimensionFourDataService.BaseAddress =
            new Uri(
                builder.Configuration["DimensionFour:ApiUrl"]
            );

        dimensionFourDataService.DefaultRequestHeaders
            .Add(
                builder.Configuration["DimensionFour:Header1Name"],
                builder.Configuration["DimensionFour:Header1Value"]
            );

        dimensionFourDataService.DefaultRequestHeaders
            .Add(
                builder.Configuration["DimensionFour:Header2Name"],
                builder.Configuration["DimensionFour:Header2Value"]
            );
    }
);
```

Figure 7.3: Implementation of C# HTTP client factory used to register data service

### 7.3.2 GraphQL

From the perspective of a frontend consumer GraphQL is a query language for data APIs, it is also a runtime layer that needs to be implemented on the backend. One of the requirements for this project was to use Dimension Four's GraphQL API as a data backend for our IoT monitoring application. Posting to the GraphQL API is done inside the monitoring application using a HTTP Client, the request is a GraphQL query formatted as a single string with variables,

where the variables are passed in as arguments to a function that replaces the variables with the objects passed in. The query result is formatted in JSON.

An example of a C# function generating a string containing a GraphQL query to create a new space, by taking in name and parentId as parameters and return a string representing a GraphQL query is shown below.

```
public static string CreateSpace
(string name, string? parentId = null){
    return $"mutation{
        {
            space{
                {
                    create(
                        input:{
                            {
                                name: \"{name}\"
                                parentId: \"{parentId}\"
                            }
                        }
                    )
                    {
                        {id}
                    }
                }
            }
        }
    }";
}
```

Figure 7.4: C# Code snippet, function returning string representation of a GraphQL query to create a space in Dimension Four IoT platform

### 7.3.3 Serialize and Deserialize JSON Data in .NET

Data transfer object (DTO) classes using the System.Text.Json namespace [25] have been created to serialize and deserialize between JavaScript Object Notation (JSON) and proper C# objects.

## 7.4 Deployment

The IoT Monitoring app has been published to Azure using a free and shared service plan with limited resources available intended to be used for development and testing purposes. However, it can easily be upgraded to a paid plan when the traffic or production requirements increase [26]. The web application is publicly available at the following link: <https://fm4017.azurewebsites.net/>

## 7.5 User Interface

The graphical user interface consists of one part that is available to the public without requiring any authentication or authorization, and one part for administrators of the site giving a detailed view of the data and the possibility for full CRUD operations of the data in Dimension Four IoT platform.

### 7.5.1 Publicly Available Page

The main page shown on a computer and on an Android mobile phone is depicted in Figure 7.5 and Figure 7.6.

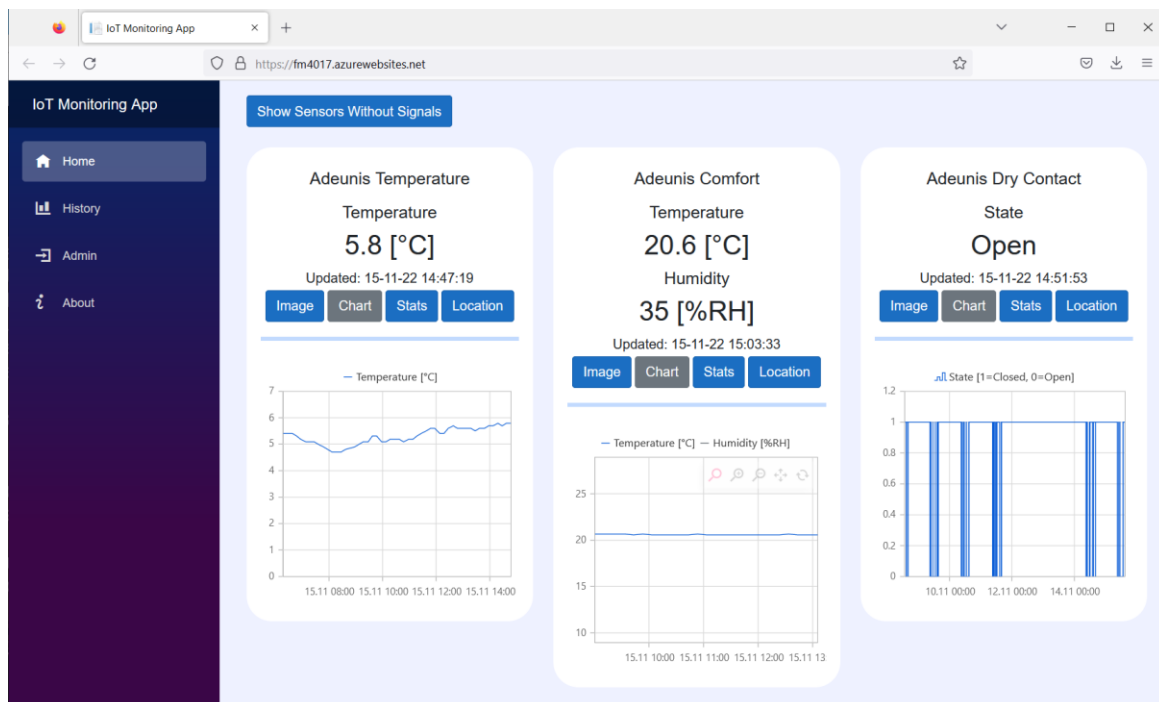


Figure 7.5: The main page display when opened on a computer

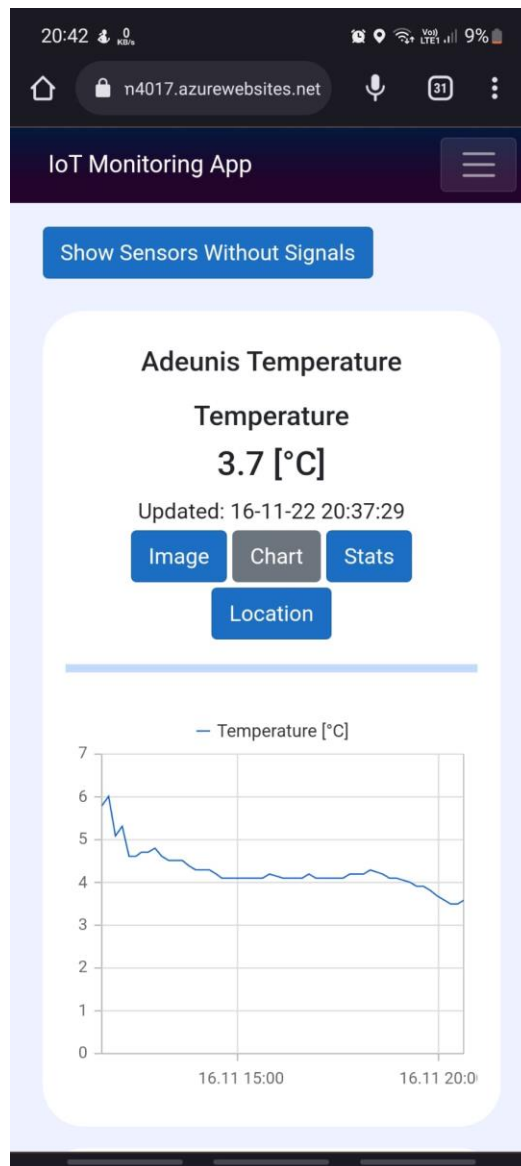


Figure 7.6: Look of the home page on an Android mobile phone

Historical data can be viewed on the History page, and available historical signals in Dimension Four IoT platform can be plotted in a chart for which an example is illustrated in Figure 7.7. Additionally, a button to export the chart to file is included to make it easy for the user to save any plot on a local PC.

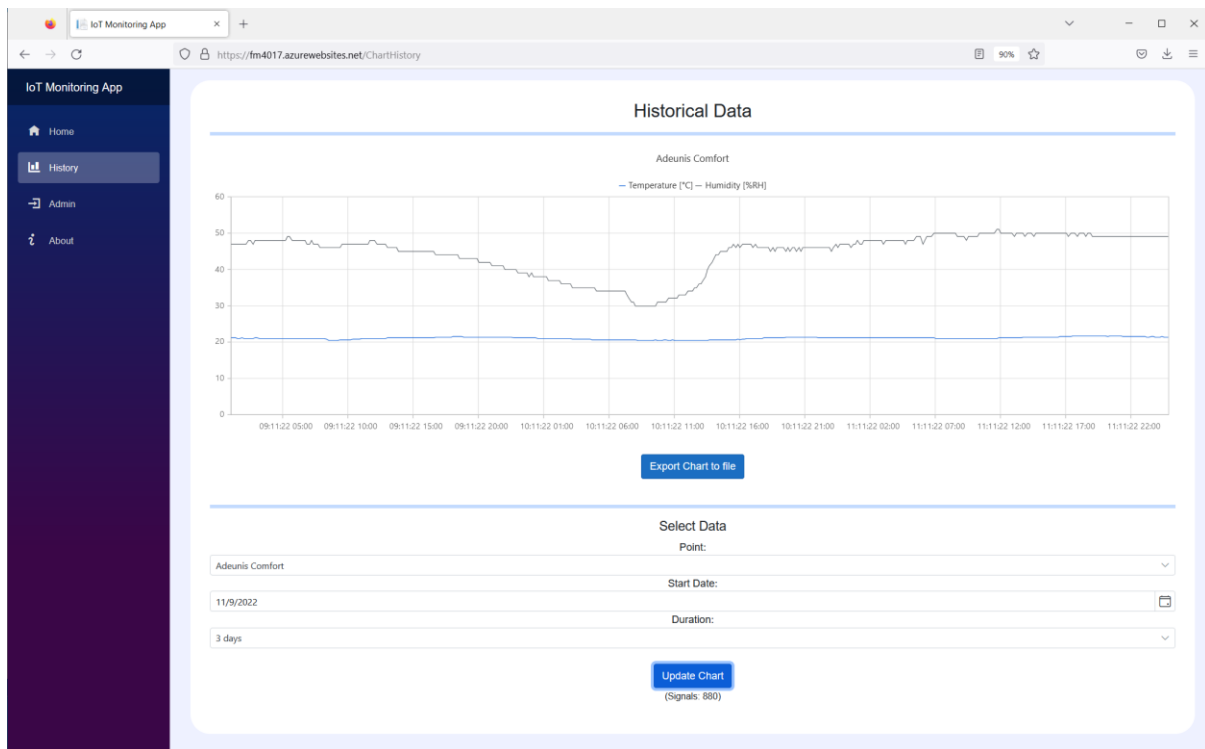


Figure 7.7: History page here displaying a plot of signals from 3 days for Adeunis Comfort sensor

## 7.5.2 Administration (Admin) Page

When authenticated and authorized as an admin by logging in with a valid username and password, the admin page is available. Here it is possible to browse the complete data structure in Dimension Four IoT platform including details like battery level and signal strength for signals. Additionally, full CRUD operations are available for Spaces and Points, signals are possible to create manually here. But editing and deleting of signals are not supported by Dimension Four IoT platform API [23].

An example of the admin page browsed into the space containing the Adeunis\_Comfort point is shown in Figure 7.8 where all the buttons for CRUD operations open a new pop up prefilled with relevant data. In Figure 7.9 it is shown the pop up appearing by clicking the edit point button in Figure 7.8. The input fields also contain input validation to avoid user mistakes such as forgetting to enter a name.



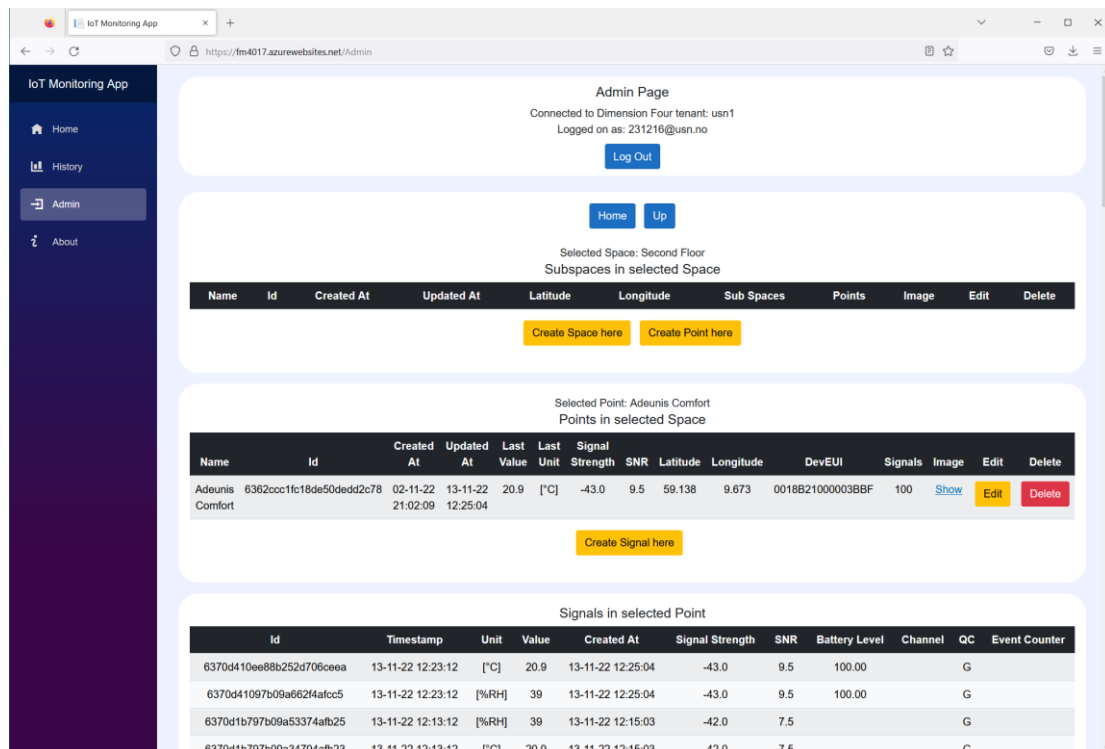


Figure 7.8: Admin page, here browsed into the space containing the Adeunis\_Comfort point

Edit: Adeunis Comfort
✕

Name:

Adeunis Comfort

Latitude:

59.138056

Longitude:

9.673333

Image Url:

https://www.adeunis.com/wp-content/uploads

DevEUI:

0018B21000003BBF

Close

Save changes

Figure 7.9: Example of pop up shown when clicking the button edit on Adeunis\_Comfort

## 8 Discussion

When LoRaWAN sensors are installed deeply indoors then usage of indoor gateway becomes imperative to get reliable and strong signals from the sensors.

The current datalogging application depends on an unsecure webhook site to fetch data from Altibox's ThingPark portal. The webhook URL is unique and not easy to guess. However, if it was exposed, it could lead to serious data loss. A secure webhook solution shall be used when developing the datalogging platform for commercial and production use.

The datalogging platform was built in Python to explore a variety of programming languages in this project. For deploying the datalogging platform as a production application, ASP .NET or similar programming languages can be employed. When using ASP .NET, it can be easily published on cloud in MS Azure portal as seen in the monitoring application of this project. Currently the datalogging application is running on a Raspberry Pi 3 as a proof of concept. As a further development step, it can be placed on a cloud server with Python script functionality to ensure uninterrupted and continuous reliability of the datalogging system.

Dimension Four GraphQL has imposed a limitation of 100 records per pull request. This number is too small when compared with the requirement of pulling lots of historical data. Due to this limitation, pagination and looping was needed to retrieve historical data. This can cause significant delays in the monitoring application. If this limit could be increased, the response time of the monitoring application can be improved.

There are a lot of existing frameworks such as ASP.NET, React, Angular to develop web applications. Blazor Server framework gave the flexibility to create a single page web application with ease, therefore it was selected to develop the monitoring web application for this project.

The monitoring application can further be extended by using machine learning algorithms on the historical data to predict temperature and relative humidity. Also, the user interface can be improved by using pictorial representations to represent weather conditions.

In Figure 8.1 it can be observed that the LoRaWAN temperature sensor provides reliable and timely information to the public as compared to information received from weather.com on Google.com and it is evidence of the quality of goals achieved in this project.

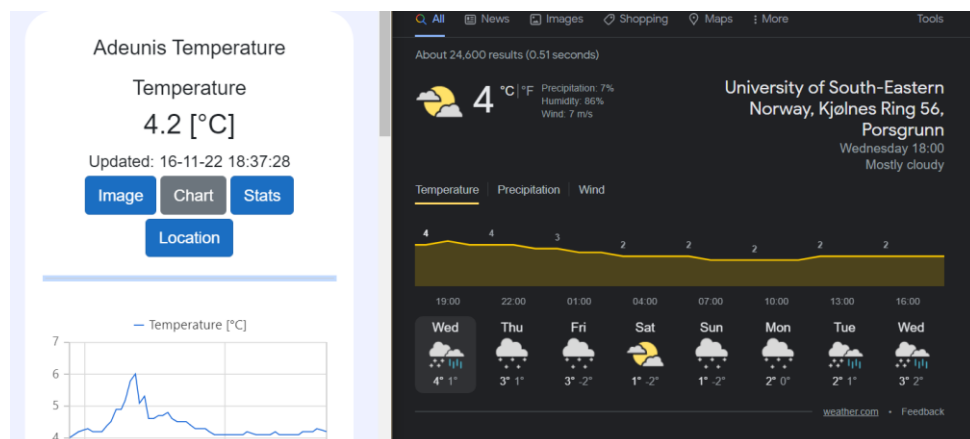


Figure 8.1: Comparison of monitoring application with weather.com

## 9 Conclusion

LoRaWAN coverage at USN, Porsgrunn and Porsgrunn city was tracked and documented. Three LoRaWAN sensors were installed at the weather station near Building C, Process Hall and Room B237a. An indoor LoRaWAN gateway was installed in the Process Hall. The data from the sensors was thoroughly analyzed and discussed. Datalogging of sensor signals from ThingPark portal to Dimension Four IoT platform was achieved through Python. Monitoring application utilized Microsoft's Blazor web framework to display current and historical sensors' data from Dimension Four IoT platform. The monitoring application was deployed on cloud. An administration application was also included as a part of the monitoring application capable of performing CRUD operations on Dimension Four IoT platform data structure. The entire developed system met all the objectives and scope defined for this project successfully.

The open-source source code for the datalogging platform and monitoring application can be found at the following link:

<https://github.com/LoRaWAN-PT/>

The monitoring web application and the sensor data is publicly available and can be seen on the following link:

<https://fm4017.azurewebsites.net/>

The overview of the entire project as a video presentation can be found at the following channel:

<https://www.youtube.com/channel/UCFrT1VOErNxltmLngVvw8Ow>

# References

- [1] Altibox, "Altibox and Dimension Four announces strategic partnership | Dimension Four," Altibox, [Online]. Available: <https://dimensionfour.io/blog/altibox-and-dimension-four-partner-to-support-academics-randd-institutions-and-startups-on-innovation>. [Accessed 12 10 2022].
- [2] DimensionFour, "About us | Dimension Four," [Online]. Available: <https://dimensionfour.io/about-us>. [Accessed 8 11 2022].
- [3] New Normal Group, "Chasing the New Normal | New Normal Group," New Normal Group, [Online]. Available: <https://newnormalgroup.com/about>. [Accessed 11 10 2022].
- [4] LyseKonsern, "Våre selskaper," Lyse Konsern, [Online]. Available: <https://www.lysekonsern.no/om-oss/vare-selskaper>. [Accessed 11 10 2022].
- [5] TheThingsNetwork, "LoRaWAN® | The Things Network," [Online]. Available: <https://www.thethingsnetwork.org/docs/lorawan/>. [Accessed 16 10 2022].
- [6] LoRaAlliance®, "What is LoRaWAN® Specification - LoRa Alliance®," [Online]. Available: <https://lora-alliance.org/about-lorawan/>. [Accessed 11 10 2022].
- [7] RadioBridge, "LoRaWAN Simplified: How do Wireless Sensors use LoRaWAN?," RadioBridge, [Online]. Available: <https://radiobridge.com/blog/wireless-sensors-using-lorawan>. [Accessed 17 10 2022].
- [8] TheThingsNetwork, "LoRaWAN Fundamentals in 5 Minutes," [Online]. Available: [https://www.youtube.com/watch?v=SmDza\\_\\_wAA](https://www.youtube.com/watch?v=SmDza__wAA). [Accessed 12 10 2022].
- [9] Lifewire, "What Is Bluetooth? The Ultimate Guide," [Online]. Available: <https://www.lifewire.com/what-is-bluetooth-2377412>. [Accessed 17 10 2022].
- [10] JuniperNetworks, "Understanding the IEEE 802.11 Standard for Wireless Networks - TechLibrary - Juniper Networks," [Online]. Available: [https://www.juniper.net/documentation/en\\_US/junos-space-apps/network-director4.0/topics/concept/wireless-80211.html](https://www.juniper.net/documentation/en_US/junos-space-apps/network-director4.0/topics/concept/wireless-80211.html). [Accessed 20 10 2022].
- [11] Qualcomm, "What is 5G | Everything You Need to Know About 5G | 5G FAQ | Qualcomm," [Online]. Available: <https://www.qualcomm.com/5g/what-is-5g>. [Accessed 19 10 2022].
- [12] LoRa Alliance, Inc, LoRaWAN™ 1.1 Specification, Beaverton: LoRa Alliance, Inc, 2017.
- [13] LoRa Alliance, Inc, LoRaWAN® Backend Interfaces 46 Technical Specification, Fremont: LoRa Alliance, Inc, 2020.

- [14] LoRa Alliance, Inc, LoRaWAN® Regional Parameters, Technical Specification, Fremont: LoRa Alliance, Inc, 2021.
- [15] DimensionFour, "Tenant, Space, Point, Signal Dimension Four Docs," [Online]. Available: <https://docs.dimensionfour.io/concepts/logical-structure>. [Accessed 12 11 2022].
- [16] Adeunis, "Test IoT network coverage | Sigfox, LoRaWAN | Adeunis," Adeunis, [Online]. Available: <https://www.adeunis.com/en/produit/ftd-network-tester/>. [Accessed 15 11 2022].
- [17] Adeunis, "TEMP: IoT temperature reading | Sigfox, LoRaWAN | Adeunis," Adeunis, [Online]. Available: <https://www.adeunis.com/en/produit/temp-temperature/>. [Accessed 10 11 2022].
- [18] Adeunis, "COMFORT: IoT temperature and humidity sensor | Adeunis," [Online]. Available: <https://www.adeunis.com/en/produit/comfort-temperature-humidity-2/>. [Accessed 11 10 2022].
- [19] Adeunis, "DRY CONTACTS, IoT digital sensor | Sigfox, LoRaWAN | Adeunis," [Online]. Available: <https://www.adeunis.com/en/produit/dry-contacts-0-1-status/>. [Accessed 11 10 2022].
- [20] ThingPark, "More about LoRaWAN® radio statistics," [Online]. Available: <https://docs.thingpark.com/thingpark-wireless/7.2/Content/DMUG/More-lorawan-radio-statistics.htm>. [Accessed 20 10 2022].
- [21] Webhook.site, "API Endpoints," 10 11 2022. [Online]. Available: <https://docs.webhook.site/api/tokens.html>.
- [22] M. L. Hetland, Beginning Python. From Novice to Professional, New York: Apress, 2008.
- [23] DimensionFour, "Dimension Four Docs," [Online]. Available: <https://docs.dimensionfour.io/>. [Accessed 14 10 2022].
- [24] Microsoft, "ASP.NET Core Blazor dependency injection," [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/core/blazor/fundamentals/dependency-injection?view=aspnetcore-6.0>.
- [25] Microsoft, "System.Text.Json Namespace | Microsoft Learn," [Online]. Available: <https://learn.microsoft.com/en-us/dotnet/api/system.text.json?view=net-6.0>. [Accessed 21 10 2022].
- [26] Microsoft, "Azure - App Service pricing," [Online]. Available: <https://azure.microsoft.com/en-us/pricing/details/app-service/windows/#pricing>. [Accessed 16 10 2022].

# **Appendix**

## **Appendix A: Project description**

# FM4017 Project

**Title:** Development of Internet of Things (IoT) solution using LoRaWAN Infrastructure for storing and monitoring sensor data

**USN supervisor:** Hans-Petter Halvorsen

**External partner:** Dimension Four AS, Altibox AS

## **Task background:**

Data from LoRaWAN sensors located at USN need to be stored and monitored. In this project the LoRaWAN Infrastructure from Altibox should be used and data should be stored within the Dimension Four IoT platform. Since the autumn of 2018, the Altibox and Altibox partnership has expanded the LoRaWAN Sensor Network in Norway and currently has coverage for more than 1,000,000 households in 100 municipalities. The LoRaWAN Sensor Network is a natural extension of the fiber network with major synergies of established infrastructure and the development continues. Altibox and Partners now offer IoT Access as a commercial service, so that more people can use the Sensor Network for their own sensors.

Dimension Four (<https://dimensionfour.io>), a local company in Grenland, Norway has developed a new IoT platform, which may be relevant to use by USN in the future. The IoT platform uses MQTT and GraphQL.

## **Task description:**

In this project the following activities should be performed:

- Give an overview of LoRaWAN in general and in context of this work.
- Give an overview of the Altibox LoRaWAN Infrastructure.
- Give an overview of the Dimension Four IoT platform and their GraphQL API.
- Create a proper data structure within the Dimension Four platform for storing data.
- Create a datalogging platform for transferring and storing sensor data through the Altibox LoRaWAN Infrastructure and into the Dimension Four IoT platform.
- Create a monitoring web application that displays the data (that are now stored in the Dimension Four IoT platform) in an intuitive and user-friendly way. This should preferably be a web application. The monitoring application should be deployed to a proper cloud platform and be publicly available when the project is finished, so the data is available for the public, including students and staff at the university and people all over the world.
- The system should be open-source and should be available at GitHub with proper documentation.
- GitHub should be used during development.
- The system should be properly documented in form of a technical report, documentation in GitHub and on YouTube.

**Student category:** IIA

**The task is suitable for students not present at the campus (e.g. online students):** Yes

**Practical arrangements:** None


**Signatures:**

Supervisor (date and signature):

Students (write clearly in all capitalized letters + date and signature):

Name: AMILA RUWAN GURUGE

Date: 30.09.2022

Sign: 

Name: EIVIND KNUDSEN

Date: 30.09.2022

Sign: 

Name: NOORAIN SYED KAZMI

Date: 30.09.2022

Sign: 

Name: VITALY DEKHTYAREV

Date: 30.09.2022

Sign: 