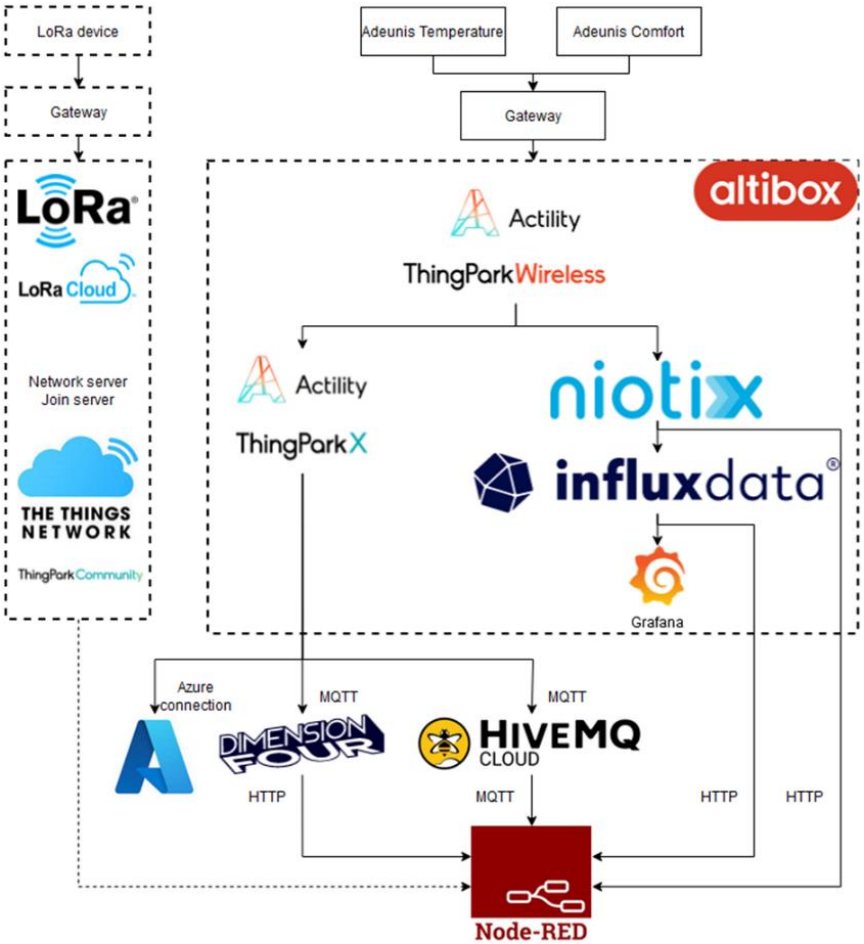
University of South-Eastern Norway

www.usn.no

FMH606-1 23V Master's Thesis

Development and Testing of LoRaWAN Sensor Network for Internet of Things Applications



MT-38-23

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606-1 23V Master's Thesis, 2022

Title: Development and Testing of LoRaWan Sensor Network for Internet of Things Applications

This report forms part of the basis for assessing the student's performance in the course.

Project group:	MT-38-23
Group participants:	Vitaly Dekhtyarev
Supervisor:	Hans-Petter Halvorsen
Project partner:	Altibox AS

Summary:

There are several wireless technologies that can be used for Internet of Things. Among these are Bluetooth, Cellular and Low Power Wide Access Network (LPWAN). Devices connected to LPWAN may work longer and be placed in remote areas. One of the implementations of LPWAN is LoRaWAN. In the current work, an infrastructure managed by Altibox AS has been applied for connecting sensors in one network using LoRaWAN protocol. A monitoring and logging application combining services provided by Altibox and open-source tool Node-RED has been setup to present measurements from sensors. Comparison of applications has shown that application from a network provider suits best for the devices under its management. However, it may be more convenient to use other solutions if data from unrelated providers are to be collected and presented in one place.

Preface

Our modern society can be characterised as highly technological. Appliances and applications work around us and produce a lot of information that may be surplus. This data can be created locally for individuals but is also available through the Internet. Nowadays it is possible to watch a web-camera located on another continent or check windspeed in the middle of the Pacific Ocean being far away from the measurement site.

However, the abundance of technological solutions also allows us to generate important data that could be vital for decision making in various fields, such as environmental monitoring or city management.

In practice, combination of the solutions has resulted in creation of the Internet of Things (IoT). User may no longer need to physically perform measurements but establish a network of devices that can communicate to each other and present the parameters of interest in a convenient way, or even perform a preliminary analysis of data.

At the same time, implementation and deployment of networks impose several restrictions on devices. These may be generalised as: power consumption, connection availability, data storage, data presentation. While for some implementations of IoT these restrictions are not critical, other applications would require a compromise to secure a stable performance.

Companies and institutions around the world work on development of compromises that suite their needs best. Meanwhile, there are solutions that will work for most situations where requirements to system components are similar across networks and implementations. One of these solutions is Low Power Wide Access Network, and LoRaWAN in particular. This is a great option for networks of smaller devices, low data payload and remote locations. Obviously, such properties would be beneficial for various tasks both in science and in industry.

By performing this work, I have found that, LoRaWAN could be a technological solution that is suited for establishing of networks that result in data-driven and sustainable development of human activity. That could lead to improved quality of life and better indoor and outdoor environment around the globe.

This work has been conducted at the University of South-Eastern Norway in cooperation with Altibox AS. I would like to thank my supervisor Hans-Petter Hansen from the University for guidance during the process. I am also thankful to Daniel Wathne Warholm from Altibox AS for giving me the opportunity to work with such a versatile technology and the infrastructure surrounding it, and clear and professional explanation of how it all works together.

Finally, I would like to thank my wife Alena for her support and patience.

Porsgrunn, 15.05.2023

Vitaly Dekhtyarev

Contents

Nomenclature

ABP – Activation By Personalization

ADR – Adaptive Data Rate

AES – Advance Encryption Standard

AS – Application Server

AppKey – Application Key

AppSKey – Application Session Key

CMAC – Cypher-based Message Authentication Code

CR – Coding Rate

CRC – Cyclic Redundancy Check

CSS – Chirp Spread Spectrum

DBPSK – Differential Binary Phase-Shift keying

DevAddr – Device Address

DevEUI – Device Extended Unique Identifier

DevNonce – Device Number Used Only Once

DSSS – Direct-Sequence Spread Spectrum

DL – Downlink

ED – End-devices

EUI - Extended Unique Identifier

FDMA – Frequency Division Multiple Access

FHDR – Frame Header

fNS – Forwarding Network Server

FNwkSIntKey – Forwarding Network Session Key

FPort – Port Field

FRMPayload – Frame Payload

FTD – Field Test Device

GW – Gateway

GFSK – Gaussian Frequency Shift Keying

hNS – Home Network Server

IoT – Internet of Things

ISM – Industrial, Scientific, and Medical

JS – Join Server

JoinEUI – Join Server Extended Unique Identifier
JoinNonce – Join Server Number Used Only OnceLoRa – Long Range
LoRaWAN – Long Range Wide Area Network
LPWAN – Low Power Wide Area Network
LR-FHSS – Long Range Frequency Hopping Spread Spectrum
LTE – Long-term evolution
MAC – Medium Access Control
MHDR – MAC payload header
MIC – Message Integrity Code
MoT –MAC on Time
NB-IoT – Narrow Band IoT
NetID – Network ID
NS – Network Server
NtwKey – Network Key
NwkSkey – Network Session key
NtwSEncKey – Network Session Encoding Key
OTAA – Over-The-Air Activation
PHDR – Physical header
PHYPayload – Physical payload
QPSK – Quadrature Phase Shift Keying
RPMA – Random Phase Multiple Access
RSSI – Radio Signal Strength Indicator
SaaS – Software as a Service
SF – Spreading Factors
SNR – Signal-to-Noise Ratio
sNS – Serving Network Server
TLS – Transport Layer Security
TOA – Time of Arrival
UNB – Ultra-Narrow Band
UL – Uplink
USN – University of South-Eastern Norway

1 Introduction

Nowadays, Internet of Things (IoT) is under continuous development and is available to various users. Devices used in IoT can be simple sensors or an integration of several sensors or actuators and perform different practical tasks. The main advantage of these devices is that they can operate without human's direct involvement, but supply a valuable information to a remote storage, Cloud, which may be later accessed by the users.

IoT solutions can be implemented in a form of an electrical socket controller that can be monitored or switched on and off by a user through a mobile application. A more complex example is a Smart Home system, that can control ventilation, heating and all other important elements comprising comfort in a living space. A connection to IoT devices can also be used in industrial or office buildings or as Smart City system for more efficient resource management at a larger scale.

However, purposes and tasks for IoT devices create also challenges such as possibility for long-range communication, available power capacity and data rate at which the devices can transmit information.

Currently implemented technical solutions cope with these challenges differently. Users and developers must always find a compromise. For instance, cellular and WiFi technologies have a good data rate, but consume a lot of power and require transmitters in the vicinity of devices. In turn, less power-demanding Bluetooth devices have a very short data transmission range, up to several tens of meters. Satellite dishes with long transmission range require clear sky, and they are power-intensive as well. Costs related to usage of these technologies may also vary highly, depending on the number of devices and network throughput.

One of the directions in wireless technology sector is usage of Industrial, Scientific, and Medical (ISM) radio transmission band and combination of hardware components in a way that it provides longest possible range for communication. The band is free to use, but with certain constraints. Also, the end-devices that are a part of the network must consume lowest possible power and be cost-efficient.

This technique is implemented among others in a Low Power Wide Area Network (LPWAN) that may be a good alternative to more expensive and demanding systems [1].

One of the implementations of LPWAN is Long Range Wide Area Network (LoRaWAN). The network is used in agriculture and industry, and there is a variety of instruments available in the market.

Several of devices for measuring meteorological parameters, such as temperature and humidity, and compliant with LoRaWAN standard have been deployed at the University of South-Eastern Norway (USN). These devices are connected into a network and supply data to a centralised system. The observations are available for monitoring, logging and visualisation. The devices and services for the network managing are provided by the external partner, Altibox AS.

In this work, I investigate this existing LPWAN network and LoRaWAN protocol with focus on improvement of data logging and visualisation and future combination with other Long Range (LoRa) devices from other network operators. The detailed project description is given in the Appendix A. The sketch of the resulting system is presented in Figure 1.1. LoRaWAN-supporting sensors transmit data to Altibox along with other network operators which provide

data to the same end-user from sensors connected to their networks, marked as dashed line. Therefore, several networks operating with LoRaWAN could be combined. Furthermore, the end-user interface has a functionality for displaying all data and saving it.

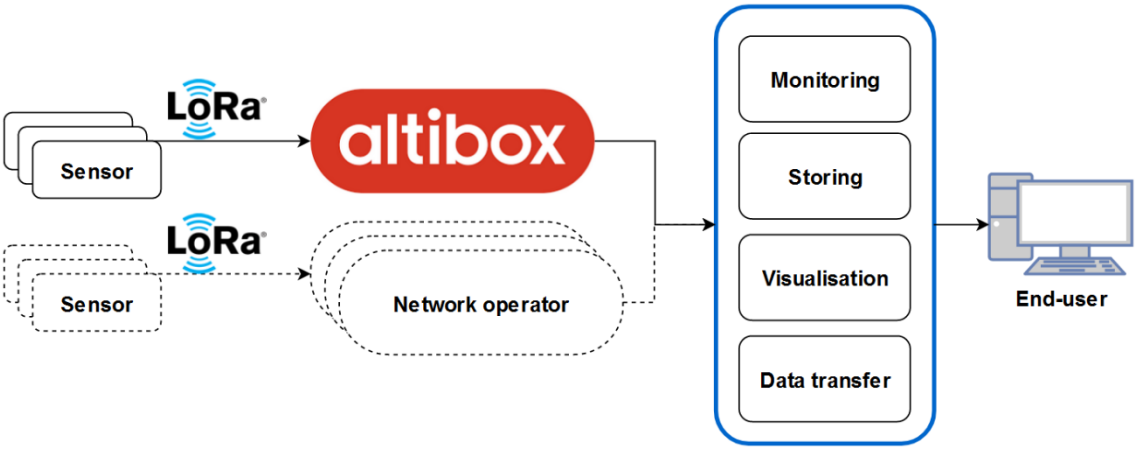


Figure 1.1: System sketch.

The objectives of the work are:

1. Explore LoRaWAN protocol and requirements for LoRa devices;
2. Investigate how the services provided by Altibox AS are utilised to manage devices deployed at the USN and operating with LoRa;
3. Examine what methods for data transmission from Altibox AS are available;
4. Develop a monitoring and logging system that collects data from available LoRa sensors in real-time;
5. Establish a way to transmit data from LoRaWAN management system directly to Dimension Four service;
6. Investigate security of the LoRaWAN protocol and give overview of the potential flaws.

The thesis consists of the following chapters: Introduction, Wireless technologies suitable for smaller IoT devices, LoRaWAN specifications and network elements, Device configuration, Altibox LoRaWAN system, Node-RED, Results, Discussion and Conclusion.

In Introduction, a background on the topic is given and objectives of the work are specified.

While the focus of the work is LoRaWAN, it is reasonable to compare alternative solutions to the chosen one. Therefore, other LPWAN are highlighted in “Wireless technologies suitable for smaller IoT devices” chapter. The chapter gives a general overview and comparison of the existing wireless technologies that can be used in IoT sector. Then, in LoRaWAN specifications and network elements, a description of LoRaWAN protocol, including topologies, elements and communication properties, is given. The chapter focuses on functionality of end devices and servers and the way they operate within the network.

Device configuration chapter provides description of the devices used in the project and how they can be setup to operate with the network. Altibox LoRaWAN system chapter discusses services, ThingPark and ThingParkX, with description of methods that user can utilise to observe measurements from deployed instruments. Besides, it is provided how measurements can be visualised using third-party services niotix and Grafana. After that, in Node-RED monitoring application, an open-source solution Node-RED is described. The tool can be used to build a monitoring application with built-in and additional modules for combining data from several operators in one end-user interface.

The Results chapter demonstrates several dashboards developed using the approaches described in previous chapters.

The Discussion and Conclusion parts provide overview of the achieved results, consider advantages and disadvantages of the chosen approaches and suggest further work.

2 Wireless technologies suitable for smaller IoT devices

This chapter gives a short overview of the existing wireless technologies that can be used in conjunction with IoT. Various technological approaches that influence performance of the devices connected to IoT and networks they are connected to, may be implemented. The technologies differ in usage of physical layer and protocols for transmission of data. The networks also have different accessibility for users: they may be private or open, with some restrictions. The network may also require an operator company to maintain connectivity. Therefore, IoT has different properties, which are hidden from the end users.

Deployment of devices that may be connected to the IoT is limited by location due to remoteness and power accessibility. The devices may not transmit signal far enough to reach a receiver. Also, to transmit a signal, the device may consume so much power that it needs a stable power source connected directly to the device. To overcome these limitations, LPWAN technology has been developed [2]. This technology is characterised by low power consumption which allows deployment of devices powered by batteries, and long range data transmission (the range may be over 10 kilometres in rural areas) [3].

Nowadays, there is a number of LPWANs such as SigFox, Ingenu RPMA, NB-IoT, Symphony Link and LoRaWAN.

Most of the technologies use unlicensed bands, which are free to use, as well as licensed ones. Use of unlicensed frequencies may cause interference, because of multiple transmitters that are online, that must be somehow overcome by providers and developers. Licensed frequencies may be more resilient, but it comes at extra costs. Nevertheless, use of all bands is regulated, and technology must comply with a specific standard defined for a geographical region the equipment is deployed in.

The presented LPWANs can be divided in two groups, those using LoRa modulation and those using their own or combination of other modulations.

2.1 Non LoRa-based networks

SigFox utilises proprietary technologies for data transmission. Available frequencies are in unlicensed sub-GHz ISM band. In Europe they are in the band from 868 to 868.2 MHz, for other regions from 902 to 928.MHz [3]. SigFox implements Differential Binary Phase-Shift keying (DBPSK) and Gaussian Frequency Shift Keying (GFSK) to modulate signals. As an improvement against collisions, SigFox uses Ultra-Narrow Band (UNB) of 100 Hz on a 192 KHz wide spectrum range and bit rate at 100 or 600 bits/s, the latter depends on the region [4].

Coverage of the SigFox in the World is shown in Figure 2.1.

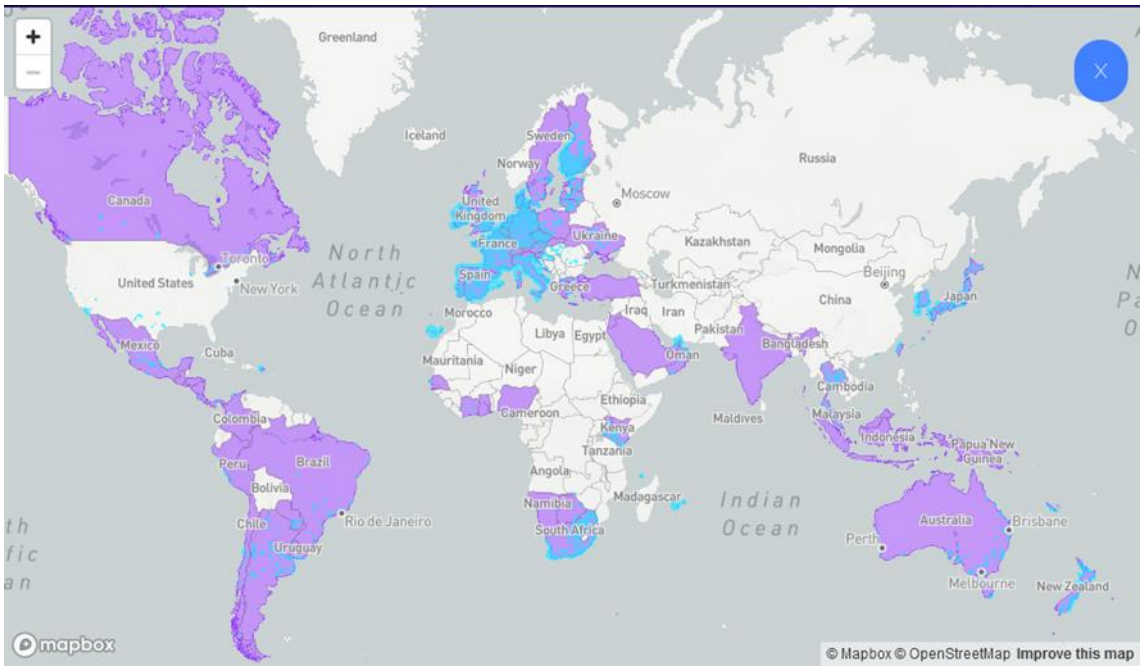


Figure 2.1: SigFox network coverage on 22.01.2023 [5].

To increase quality of transmission SigFox introduces redundancy by changing transmission frequency and repeating messages twice. After a base station, that demodulate signal, has received a message, it is forwarded for processing to the cloud provided by the SigFox Support System. No handshaking between device and base station is required. The cloud can be accessed with a web interface or a REST API (Figure 2.2).

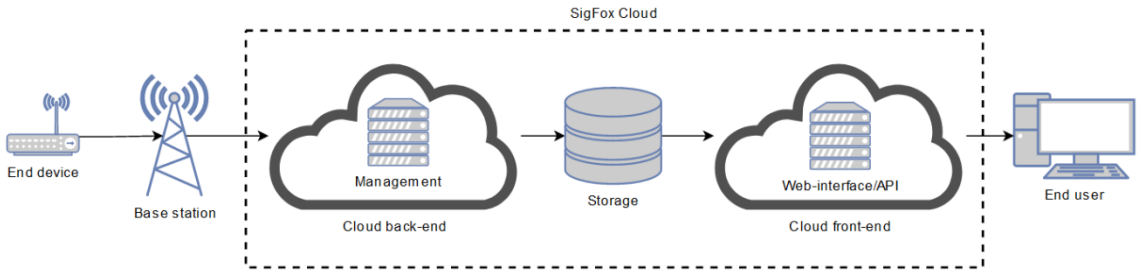


Figure 2.2: SigFox network architecture [4].

Nevertheless, SigFox has several limitations: paid activation of a device in the network through buying an account, 6 uplinks per hour, 4 downlinks per day, uplink payload is 12 bytes [4] and 8 bites for downlink [3], coverage in Norway is undefined (Figure 2.1).

Another technology in this group is Ingenu RPMA. It uses 2.4GHz band with Random Phase Multiple Access (RPMA) modulated using Direct-Sequence Spread Spectrum (DSSS) and increased range. It also allows multiple end-devices to be connected to the same access point. End-users obtain data using a RESTfull API. The band used by Ingenu is regulated, but it is also utilised in other communication methods, such as WiFi and Bluetooth. Therefore, it may

be prone to interference and collisions [2]. Range of data transmission for this band is not as long as for sub-GHz bands.

Narrow Band IoT (NB-IoT) technology uses licensed frequencies in bands from 700 MHz to 900 MHz with bandwidth of 200 KHz. It implements a reduced Long-term evolution (LTE) communication protocol suited for IoT network. Licensed spectrum with LTE provides a better Quality of Service than ones in an unlicensed spectrum. At the same time, transmitted messages with NB-IoT are smaller and are sent less often than usually in LTE. This allows to reduce power consumption on end-devices. However, the power usage will depend on how frequent transmissions take place. Signal is modulated using Quadrature Phase Shift Keying (QPSK) in addition to Single-Carrier Frequency Division Multiple Access (FDMA) for uplink messages and orthogonal FDMA for downlinks. Uplinks with maximum payload of 1.6 kB may be transmitted at the maximum speed of 200 kbps, and downlinks at 20 kbps with the same maximum payload. Transmission range is less than 10 km and may be performed only where LTE base stations are installed [3].

2.2 LoRa modulation

Long Range is a physical layer for data transmission that operates in an unlicensed sub-GHz ISM spectrum. These bands vary from country to country and are defined according to the region of usage. In Europe, these bands include 433 MHz and 868 MHz [6]. In addition to this, an implementation of LoRa on 2.4GHz band is under development [7].

From this perspective LoRa is similar to SigFox, which uses 868 MHz in Europe [7]. The signal is modulated using Chirp Spread Spectrum (CSS), and such signal may be characterised by low noise [3]. Chirp, Compressed High Intensity Radar Pulse, encoding has been used in communication with radars. It is represented as a transmission of a symbol, pulse, and each pulse contains a certain number of bits [7]. CSS is used to modify the electrical signal to the specific bandwidth and alter the signal properties, so that it can travel to longer range. Transmission with LoRa modulation is performed on channels 125 KHz and 500 KHz for uplinks and on 500 KHz for downlinks [8].

In Europe, there are legal requirements according to regional plan EU868. A signal modulated with LoRa can use up to 16 channels in general, 8 channels for transmission at data rate up to 5.5 kbps, 1 channel up to 11 kbps, 1 channel up to 50 kbps (FSK) and 3 fixed channels. Time occupied by transmission per defined time range, duty cycle, must not exceed 0.1% of the time range [9] [6].

2.3 LoRa-based networks

The LoRa physical layer is used in proprietary standard Symphony Link. The standard is oriented towards industrial solutions. Gateways, modulating and demodulating radio signal, may communicate to multiple end-devices, number of gateways may be increased to improve speed of transmission. The network may be equipped with repeaters to increase the range of a signal transmission. The standard provides several other benefits: 1) scanning for interference implemented on end-devices and gateways, 2) avoidance of limitation on duty cycle, by implementing frequency hopping [9], 3) firmware update over the air. During transmissions end-devices receive a beacon and specifications from gateways to adapt to changing

transmission conditions. Usage of a proprietary standard Symphony Link may cause additional costs [2].

Alternative to Symphony is experimental Medium Access Control (MAC) on Time (MoT) which, according to simulation, may provide several times higher network capacity [2]. However, this technique is not available in the market.

The third LoRa-based network is Long Range Wide Area Network (LoRaWAN). It is a widely used technique (Figure 2.3). LoRaWAN covers three layers above the physical one, LoRa, namely Data Link, Network and Session, which guarantees the end-to-end communication between devices (Figure 2.4). In the network end-devices are categorised as classes A, B and C device, and the frequency can be manipulated to choose the most suitable transmission option.

Components of the LoRaWAN ecosystem also provide certain level of security and instruments for managing devices and monitoring their status and acquired measurements.

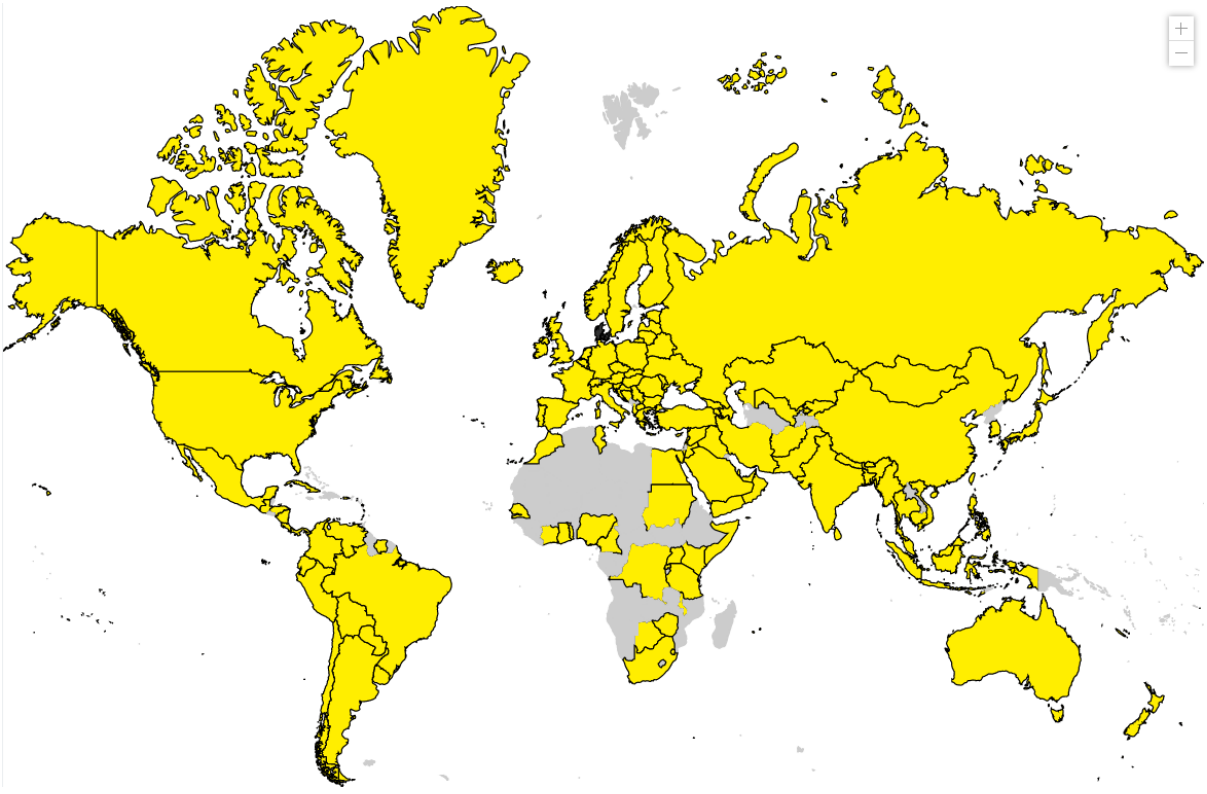


Figure 2.3: Global coverage of LoRaWAN on 22.01.2023 [10]

LoRaWAN may be established as a private, open or shared network, and may be managed by an operator, such as Altibox. Any operator should be certified by LoRa Alliance to be able to establish an open network.

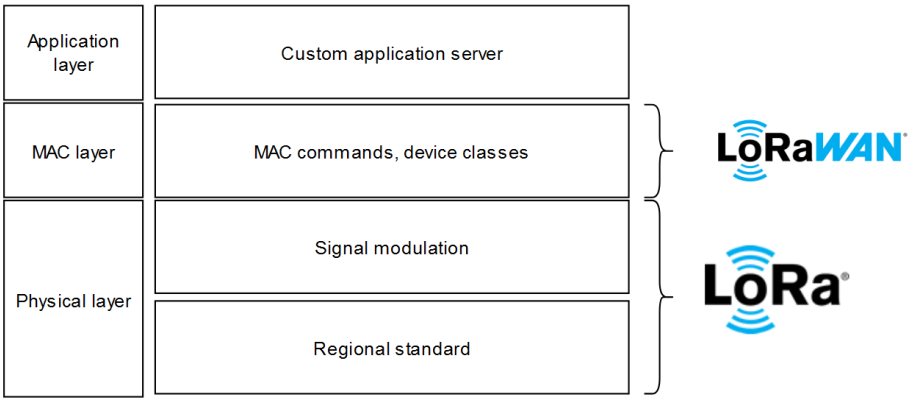


Figure 2.4: LoRaWAN stack [8].

3 LoRaWAN specifications and network elements

The work is focused on use of LoRa compatible sensors connected to LoRaWAN. This chapter covers in details how structure of the network is built, and how elements of the network function. The main components are:

1. End-devices (ED) that perform measurements or actions in the field, identified by a unique DevEUI, where EUI is Extended Unique Identifier;
2. Gateways (GW) demodulate LoRa signal received from devices and modulate messages from Network Server (NS). When ED is roaming GW forwards messages to the home NS which the device is assigned to. Home NS can receive messages on several channels at the same time;
3. Join Server (JS) is identified by a unique JoinEUI. JS allows to register new devices and keeps necessary security keys;
4. NS manages communication between EDs, JS and Application Server (AS);
5. Application Server presents data from EDs to the user and allows to send messages to devices.

Network operations will slightly differ depending on the ED. If a device is stationary it is considered that ED is at Home (Figure 3.1). If ED moves and can leave the range of one stack of JS and NS and enter area managed by another NS, the ED is considered as a roaming ED (Figure 3.2) [11].

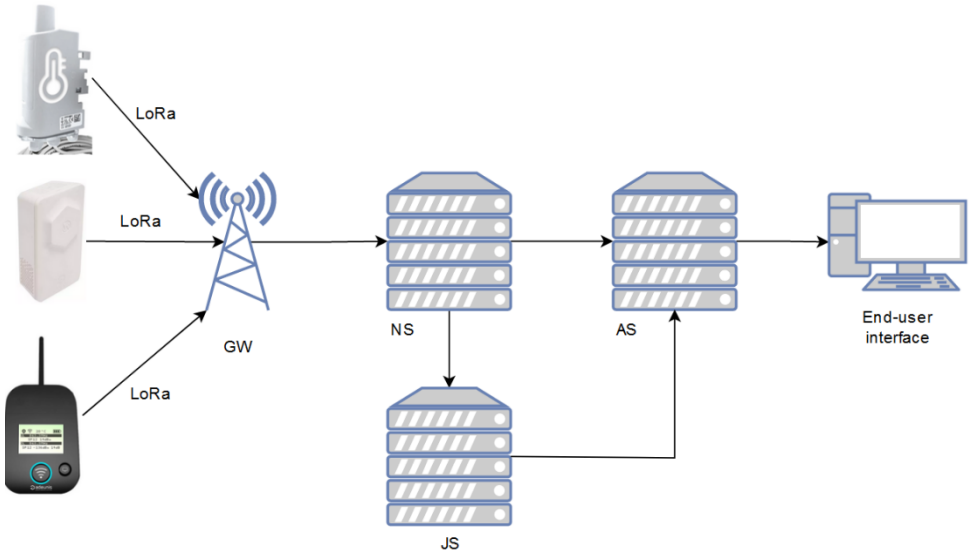


Figure 3.1: LoRaWAN with ED in Home network.

When ED changes its position, roaming, it passes through areas in range of different GWs. Then the NS, which registered ED in the network and stored ED description, profile, will be considered as a home NS (hNS). The new GWs, that reach the ED, and the ED MAC layer

will be controlled by another NS, serving NS (sNS). That sNS may manage the GWs and forward the packets to the hNS. However, sNS may not have functionality for forwarding the packets. Then there will be another NS that manages GWs and forwards packets to sNS, such NS is a forwarding NS (fNS), while sNS will still control MAC layer of the ED [11].

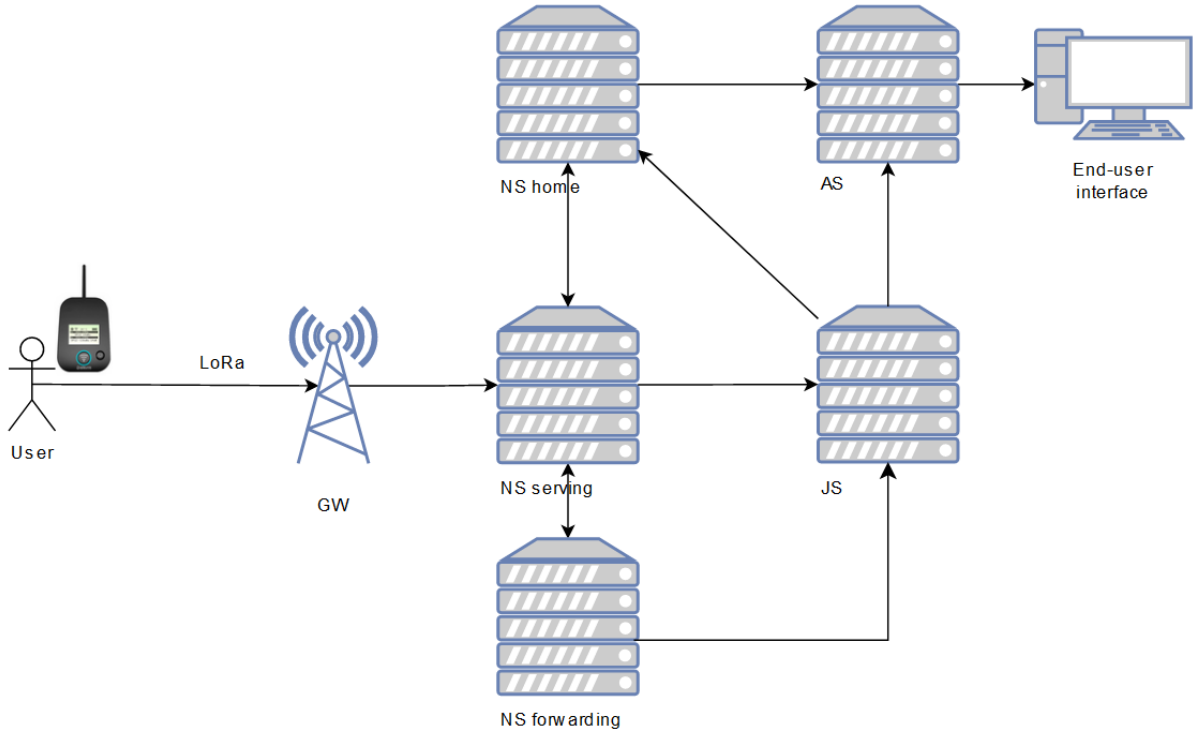


Figure 3.2: LoRaWAN with roaming ED.

EDs belong to three classes in LoRaWAN. Difference between the classes is evaluated in terms of power consumption, and how often user can send commands to the device. An estimated required power for each class may be averaged as:

- Class A – 5uA;
- Class B – 30uA;
- Class C – 10mA.

As one can see Class A refers to the least demanding device, while Class C is the most power-intensive one. An ED cannot operate in two or more class modes simultaneously.

3.1 Class A end device

This class must be supported by all devices to be certified for use in LoRaWAN.

Energy efficiency for the class A comes from the fact ED is in a deep sleep most of the time and sends a message (uplink (UL)) only when is triggered by an internal process.

While in sleep these devices do not accept any incoming messages (downlink (DL)) from GW, therefore, if a NS is sending some information, it will be queued [12].

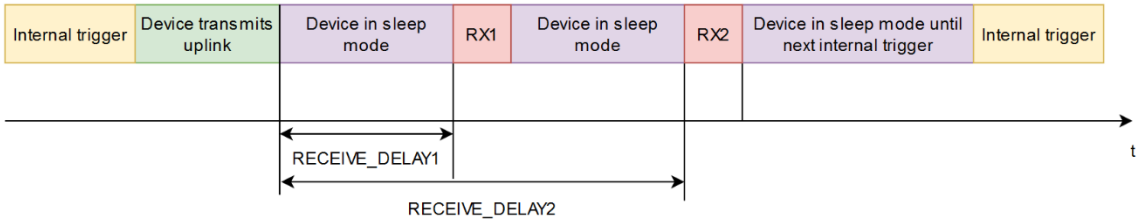


Figure 3.3: Class A end device communication [8].

Receiving of ULs takes place during receive windows (RX1 and RX2). Receive windows are open twice after the device has transmitted an uplink (Figure 3.3). The device goes in sleep mode for RECEIVE_DELAY1 seconds, which is 1s by default. Then he first receive window is open for a time period long enough to register an incoming message preamble [13]. If no downlink is received, the sleep mode is activated for RECEIVE_DELAY2 seconds, which is RECEIVE_DELAY1 + 1s by default. The second receiving window is then open for the same period of time.

In absence of any incoming message, ED switches to deep sleep mode until next internal trigger. Both delays (RECEIVE_DELAY1 and RECEIVE_DELAY2) are configurable [6] [12]. ED has lowest power consumption when it is asleep.

3.2 Class B end device

When ED is configured to class B it still has functionality of the class A device, but in addition it opens RX windows periodically on a deterministic basis. GWs transmit a beacon synchronously on the same channel and at the same frequency. Therefore, all devices of class B in a range of any GW will receive beacons at the same time, every 128s (BEACON_PERIOD) [14] (Figure 3.4).

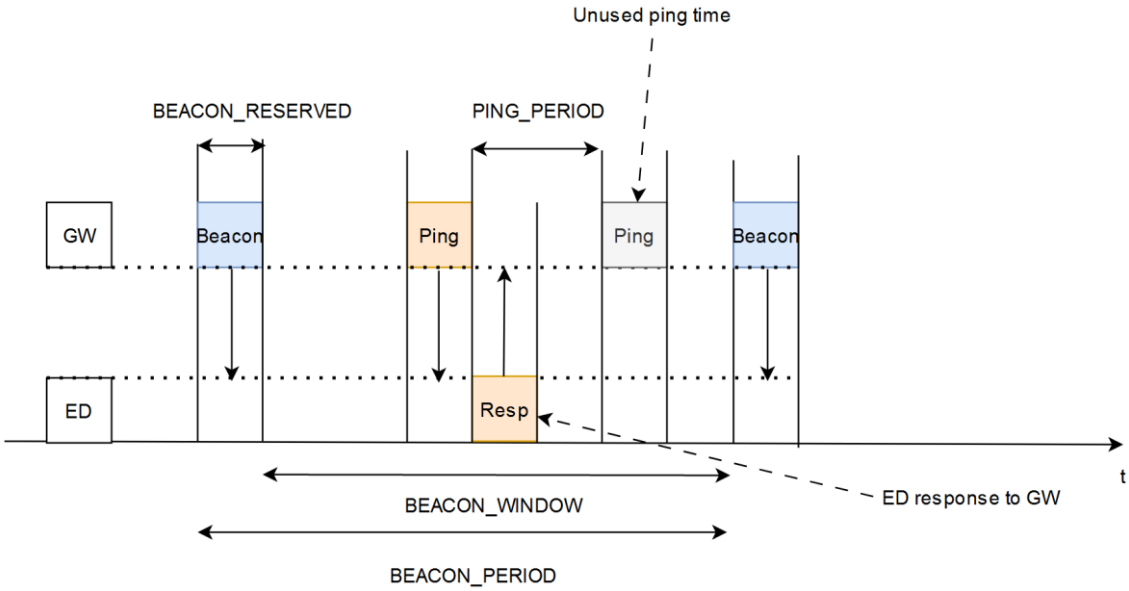


Figure 3.4: Class B end device communication [15].

This beacon is used by ED for synchronisation of internal clock with the network. After synchronisation the device adjusts the time period between ping slots. A ping slot is the time when receive window for a downlink is open on the device. The time period between beacons is BEACON_WINDOW, the time that can be used for receiving downlinks and is a bit shorter than BEACON_PERIOD (Figure 3.4), 122.880s. BEACON_WINDOW is divided equally in 4096 ping slots, 30ms long each [15]. While ping slot specification may be modified by user, beacon parameters are region specific.

If Class B device stops receiving beacons, it switches to a beaconless mode for 120 minutes. In case during this period no beacon arrives, the device switches to Class A operation [15].

Because the ED needs to open ping slots and listen to beacons, it is less power efficient than Class A device.

3.3 Class C end-device

EDs are set to class C mode when user requires frequent message exchange between the device and GWs, and power source is not of concern for the user.

In this mode the ED follows the pattern of Class A device. After transmission of an uplink, RX1 is open in RECEIVE_DELAY1 seconds, and RX2 in RECEIVE_DELAY2 seconds. However, during the delays the end device is not going into a sleep mode as Class A device does, but opens another channel for receiving downlinks (Figure 3.5) [16].

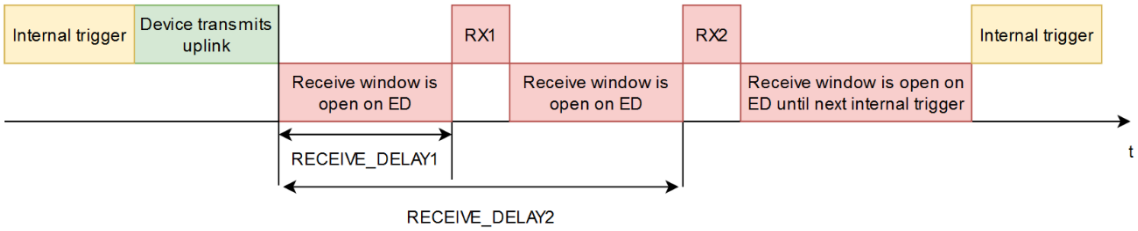


Figure 3.5: Class C end device communication [16].

In this case the ED is almost always available for the network and consumes much more power than in Class A or B mode.

This class of devices must also support all commands for Class A devices [16].

3.4 Gateway

Gateway is a message forwarder. It is able to receive, modulate and demodulate LoRa signal and is used for communication with EDs. GW receives all LoRa modulated signals that it can capture, similarly it transmits the radio signal to all LoRa devices. Also, GW is connected to the Internet using any available method and sends messaged to NS. At this stage, no specific security or encryption mechanism is implemented. GW only examines Cyclic Redundancy Check (CRC) of a message, if it is correct the message is forwarded further, if not – it is discarded [8].

When forwarding messages GW adds metadata which is used by other parts of the network:

- Radio Signal Strength Indicator (RSSI)
- Signal-to-Noise Ratio (SNR)
- Time of Arrival (TOA)
- Frequency Channel
- Data Rate

GW types vary in complexity and resilience to environmental conditions. They may be outdoor or indoor, also they may listen on 8, 16 and 64 channels, depending on implementation requirements [17] [8].

3.5 Join server

This server is responsible for allowing new devices to join the network the JS belongs to, and distribution of the security parameters within the network, so that ED can communicate to NS and AS. JS receives a Join-request message from ED and responds with Join-accept frame (message) which signals that the ED is allowed to join the network. Otherwise, Join-request frame is discarded.

Join-accept frame contains derived keys that ED uses to encode and decode frames sent to and from NS and AS. In addition, JS manages roaming ED and sends the required

information to the corresponding NSs. JS may also disconnect ED from the network if it is requested, or it may reconnect the device to the network if ED of some reason misses the received security keys.

3.6 Network server

NS is responsible for several tasks. First, NS receives messages from GWs. Because all GWs in range of ED transmission receive the same message, they will send the same message to the available NS. If a NS receives multiple copies of the same messages, it must remove duplicates [7]. Also, the server analyses metadata arriving with the messages, i.e. RSSI and SNR, and computes which GW is more suitable to use with a particular ED in terms of signal strength and transmission rate [11].

For Class B devices NS calculates the time to schedule ping slots and orders sending of downlinks to a GW [18].

During activation of ED, NS forwards Join-request to a JS if the device is static. If ED is roaming and has just reached the area, where the current NS is operating, this NS forwards Join-request to a NS where device had been registered first, home NS (hNS).

When analysing messages from EDs, NS may implement a technique that improves the device performance and optimises its battery usage. The technique is Adaptive Data Rate (ADR). NS, by evaluating Time on Air parameter, estimates which Spreading Factor (SF) should be used by the ED. This changes the data rate and duration of signal which end-device will use to transmit messages next time [7].

3.7 Data rate and transmission

ADR is activated by NS by setting a special bit in a MAC command which is communicated between NS and MAC layer of ED. If ADR is not activated, ED uses the lowest data rate, if ADR is activated, ED must process MAC command specification to adjust the bit rate corresponding to the recommended SF [13].

When ED sends data, it transmits symbols. Each symbol includes several bits of data. SF indicates the number of bits encoded in one transmitted symbol. There are several SF available in LoRaWAN, from SF7 containing 7 bits in one symbol to SF12 containing 12 bits. The higher SF, the higher transmission distance the signal can travel, but data rate is lower [3]. This comes from the specification of Chirp signal modulation. To distinguish symbols, devices start transmission of each symbol from a defined frequency. That means that the higher coefficient of SF, the more intermediate frequencies between the lowest and highest, must be used [19](Figure 3.6).

For example, if SF2 is used, there are 2 bits in one symbol. This results in 4 possible combinations of the two bits ($2^2 = 4$): 00, 01, 10 and 11. The bandwidth would be split into four equal steps. For SF2, this would count $2^7 = 128$ possible combinations or frequency steps.

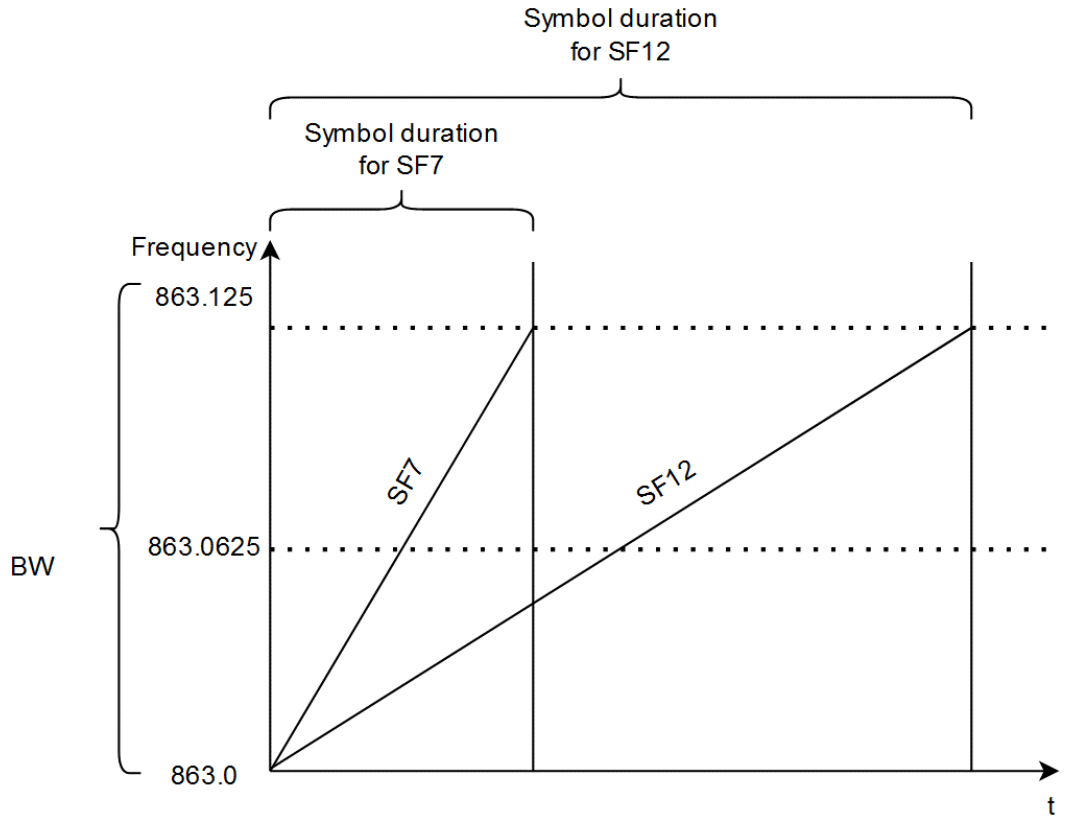


Figure 3.6: Illustrative difference between SF7 and SF12 as an example.

The data rate may vary from several hundred bits to several thousand bits per second, the exact value depends on SF and bandwidth [7]. However, the actual data rate that ED can use, is determined by regulations applied in a particular geographical region (channel plan). The plan relevant for the current work is Europe, the channel plan is EU863-870. There is also available EU433 plan, but it is not used in the project.

The list of available data rates and properties is shown in the Table 3.1.

Table 3.1: Data Rates available for EU863-870 channel plan.

Data Rate ID	Modulation	Spreading Factor	Bandwidth	Bit rate [bit/s]	Symbol transmission time, T_{symbol}, [ms]	Maximum useful application payload size [octet]
DR0	LoRa	SF12	125 kHz	366	32.768	51

DR1	LoRa	SF11	125 kHz	671	16.384	51
DR2	LoRa	SF10	125 kHz	1220	8.192	51
DR3	LoRa	SF9	125 kHz	2197	4.096	115
DR4	LoRa	SF8	125 kHz	3906	2.048	222
DR5	LoRa	SF7	125 kHz	6836	1.024	222
DR6	LoRa	SF7	250 kHz	13671		222
DR7	FSK			50000		222
DR8	LR-FHSS CR 1/3		137 kHz	162		50
DR9	LR-FHSS CR 2/3		137 kHz	325		115
DR10	LR-FHSS CR 1/3		336 kHz	162		50
DR11	LR-FHSS CR 2/3		336 kHz	325		115

Any ED shall be able to operate on at least data rates from DR0 to DR5 to be certified to participate in LoRa network [6].

In the table LR-FHSS is Long Range Frequency Hopping Spread Spectrum modulation, and CR is coding rate. CR is used for error detection and correction. The value indicates how many additional bits to be transmitted relatively to the original number of bits. CR 1/3 indicates that there will be transmitted 3 bits per each bit, and CR 2/3 indicates that 3 bits to be transmitted per each 2 bits. Therefore, CR increases the size of a message to guarantee successful reception.

Importance of data rate for transmission is not only related to the latency, but also for meeting region specific requirements. EU863-870 channel plan limits Duty Cycle (DC) to less than 1%.

DC is a relation of cumulative duration of transmission, time on air, (T_{on_cum}) to an observation time interval (T_{obs}), which is by default 1 hour, for a specified bandwidth F_{obs} (Equation 3.1) [20]

$$DC = \left(\frac{T_{on_{cum}}}{T_{obs}}\right)_{F_{obs}} \tag{3.1}$$

That means that ED is allowed to use 36 seconds during an hour for continuous transmission of data

$$T_{on_{cum}} = \frac{1\ hour}{100} = \frac{3600\ seconds}{100} = 36\ seconds$$

Alternatively, ED can stop transmitting for a time equal to the last time used for transmission multiplied with 99. That is, if a transmission took $50ms$, ED could safely send a new uplink in $50ms * 99 = 5445ms = 5.445s$.

To estimate how much time will be used during transmission, one can use equation for Time on Air from equation 3.1 [7].

$$Time\ on\ Air = n_{symbol} T_{symbol} \tag{3.1}$$

where n_{symbol} is number of symbols to be transmitted and T_{symbol} is time required to transmit a symbol.

T_{symbol} can be found from SF and used bandwidth:

$$T_{symbol} = \frac{2^{SF}}{Bandwidth} \tag{3.2}$$

Finding n_{symbol} is more complicated as the actual number of symbols in one message depends on several factors and can be calculated using equation 3.3 [7].

$$n_{symbol} = n_{preamble} + 4.25 + \\ +8 + \max\left(ceiling\left(\frac{8Payload - 4SF + 28 + 16 - 20H}{4(SF - 2DE)}\right)(CR + 4), 0\right) \tag{3.3}$$

In the equation above $n_{preamble}$ is the first part of a message used by GW and ED to initialise a message reception, H is an indicator showing if a header in the message is used or not, has values of 1 or 0, DE shows if a low data rate optimisation is enabled, has values of 1 or 0.

This specification is, however, based on usage of a calculator developed for SX1272 module by Semtech. Therefore, for better estimation of Time on Air, a dedicated software or documentation for hardware must be used.

Bit rate of transmission may also be estimated for a specific bandwidth using the equation 3.4 [7].

$$BR = SF \frac{Bandwidth}{2^{SF}} \tag{3.4}$$

Messages transmitted between ED and NS are also of different types and sizes and can be classified as:

1. Join-Request sent by ED to NS to join a network;
2. ReJoin-Request sent by ED to NS to join a network again;
3. Join-Accept sent by NS to ED to confirm that ED can join a network;
4. Uplink sent by ED to NS under normal operation;
5. Downlink sent by NS to ED under normal operation.

General structure of a packet with data that can be received from ED (uplink) is shown on Figure 3.7.

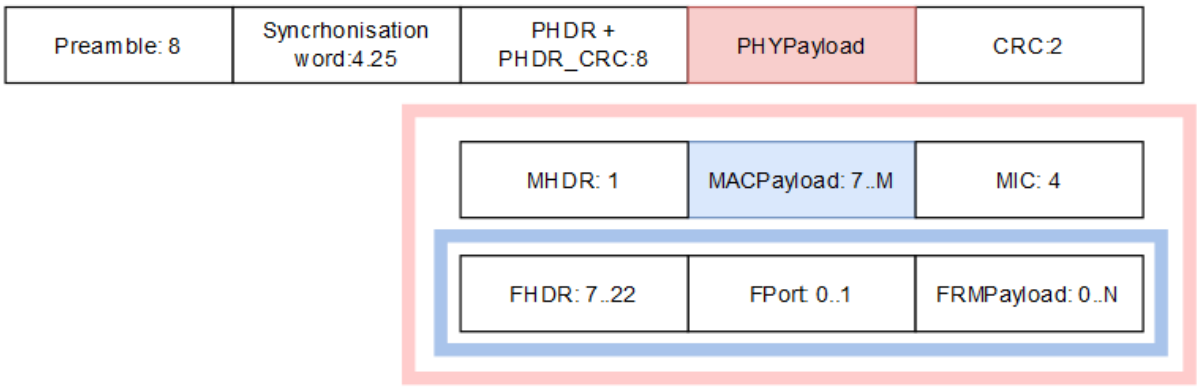


Figure 3.7: LoRa physical layer packet structure, uplink [13] [6]. Values M in MACPayload and N in FRMPayload represent the maximum velues.

Structure of the packet may also vary depending on the LoRa mode, which can be explicit or implicit. In explicit mode Physical header (PHDR) and CRC for the header (PHDR_CRC) are included, while in implicit one they are absent.

CRC on the end of the packet is only used with packets from ED to GW, uplinks, and not with ones from GW to ED, downlinks.

MAC payload header (MHDR) and MAC payload must always be integrity-checked what is indicated in Message Integrity Code (MIC). MHDR specifies what type of message arrives and how it must be processed.

Frame Header (FHDR) contains the ADR bit which gives the ED instruction to change its data rate.

Port Field (FPort) indicates if the content of Frame Payload (FRMPayload) is a MAC command, that requires a special treatment, or an application specific data [13].

Maximum size of physical payload (PHYPayload) is dependent on the size of FHDR, FPort and FRMPayload, and maximum size of MACPayload is dependent on the size of FRMPayload. Interdependency between N and M parameters is shown in equation 3.5 [13]

$$N \leq M - 1 - (length\ of\ FHDR) \tag{3.5}$$

Both N and M are region specific, so for EU863-867 channel plan these values varies according to the data rate, FHDR content and compatibility to repeaters (Table 3.2), the latter are not described here. From the table one can see that, in general, size for only four data rates is different (in bold) if repeaters are considered.

In addition, during estimation of Time on Air without using dedicated applications, user must take into account that the size of packet parts is indicated in different units. Preamble, Synchronisation Word, PHDR and PHDR_CRC are measured in symbols and not in octets, bytes, that means that actual length of the parts must be recalculated to octets using SF.

Table 3.2: Maximum size of payload in octets [6].

Data Rate ID	Repeater compatible payload		Not repeater compatible payload	
	M	N	M	N
DR0	59	51	59	51
DR1	59	51	59	51
DR2	59	51	59	51
DR3	123	115	123	115
DR4	**230**	**222**	**250**	**242**
DR5	**230**	**222**	**250**	**242**
DR6	**230**	**222**	**250**	**242**
DR7	**230**	**222**	**250**	**242**
DR8	58	50	58	50
DR9	123	115	123	115
DR10	58	50	58	50
DR11	123	115	123	115

Besides, structure of packets transmitted with different techniques, as shown in Table 3.1, (LoRa, FSK, LR-FHSS) comprises different components, specified in documentation.

By defining SF and data rate, it is defined for how long ED stays online to transmit data. These factors have significant influence on performance of ED. Higher SF allows to place ED further from GW because lower frequencies have better penetration properties than the higher ones. However, that also makes ED to use more time on the transmission before it goes into sleep, and consequently use much more power than with lower SF.

3.8 Application server

Application server may be considered as an end point in LoRaWAN ecosystem. Uplinks from ED are forwarded to this server and are received by the end-user. NS use DevEUI specified in message to forward it to the corresponding AS. There may be several AS communicating with one NS, and there may be several NS communicating to one AS. Similarly, one AS may communicate to one or several JS, and vice versa [11].

After receiving and decoding payload, AS can send useful data to other servers outside LoRaWAN infrastructure. Besides, functionality of AS allows end-user to send downlinks to ED to manage its behaviour.

3.9 LoRaWAN security and end-device activation

At the physical layer, the network is considered robust because the CSS modulation technique is resilient to degradation and interference. However, it is possible to use existing off-shelf devices to disturb the signal. At the same time, such actions can be identified by system administrator and be taken care of [21].

At MAC level, the network supports several levels of security, which varies between versions of protocol, but all utilise Cypher-based Message Authentication Code (CMAC) with Advance Encryption Standard (AES), or AES-CMAC, with 128 bits key size to protect information [13]. This algorithm detects intended as well as accidental data modifications [22].

If a new ED is supposed to join a network, LoRaWAN considers two ways of secure inclusion of the ED: Activation By Personalisation (ABP) or Activation Over The Air (OTA). However, even before any device can participate in the process of activation, it must be configured. Besides, depending on the chosen way of activation, other components of the network must also be configured in advance to be able to process uplinks from the ED.

Configuration includes providing a number of security keys to ED for encoding and decoding messages, but the list of keys to be used depends on version of LoRaWAN.

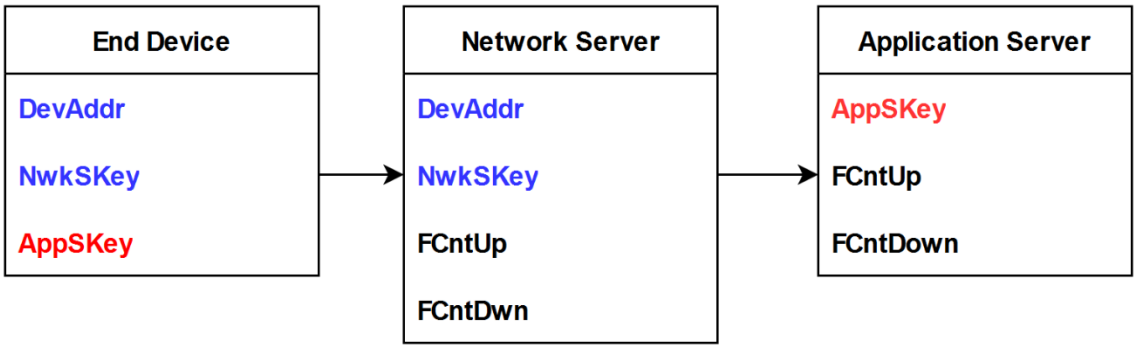


Figure 3.8: LoRaWAN 1.0 Session context. Keys in blue colour are used on ED and NS only, the key in red is used on ED and AS only.

When ED is operating in LoRaWAN 1.0 (Figure 3.8) it encodes application data using Application Session Key (AppSKey) and then encodes the message using Network Session key (NwkSkey) before sending to NS. On reception, NS examines the message with Device Address (DevAddr) and checks MIC with NwkSKey. If the message passes the check, it is sent further to AS. AS uses stored AppSKey to decode and encode payload received from NS [14].

In case of LoRaWAN 1.1 (Figure 3.9), ED requires three keys instead of one to communicate with NS. ED uses Forwarding Network Session Key (FNwkSIntKey) to calculate MIC for uplinks, Serving Network Session Integrity Key – for calculating MIC for downlinks, Network Session Encoding Key (NtwSEncKey) – for encoding and decoding all MAC commands. AppSKey is used in the same manner as for LoRaWAN 1.0 [13].

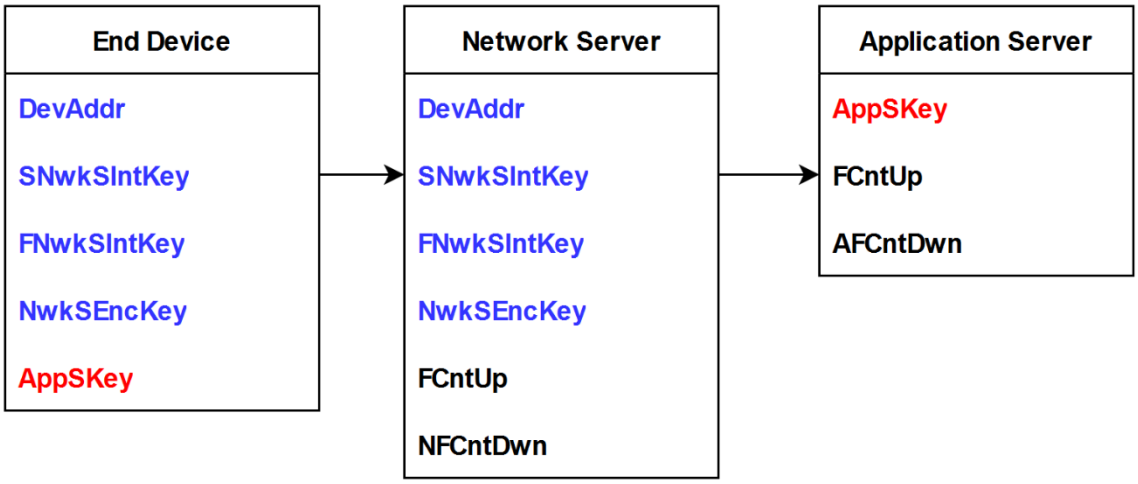


Figure 3.9: LoRaWAN 1.1 Session context. Keys in blue colour are used on ED and NS only, the key in red is used on ED and AS only.

None of these keys is transmitted between network components during normal data exchange. They all are stored locally on each device and are ED specific.

When ABP is implemented, the keys are uploaded by network administrator. They all must be known and manually saved on the corresponding network components.

OTA process is more sophisticated and is mainly managed by JS. Join procedure is initiated by ED that is to join the network. As the number and type of keys is different between LoRaWAN 1.0 and 1.1, the join procedures also involve different number of steps.

Before activation can start, both ED, JS, and in some cases AS, must be configured properly. ED and JS have JoinEUI, DevEUI and root keys stored on the devices, for 1.0 version the root key is only Application Key (AppKey), for 1.1 the root keys are AppKey and Network Key (NtwKey). To start the procedure ED sends a JoinRequest message to NS (Figure 3.10).

This message includes both ED and JS identifiers and a Device Number Used Only Once (DevNonce) value. The value is created by the ED and used only once during activation with a given JoinEUI. DevNonce starts from zero and is incremented each time it is used for activation. DevNonce is stored then on both ED and JS.

After receiving JoinRequest, NS forwards the message to JS for generation of session context keys. JS generates a Join Server Number Used Only Once (JoinNonce), that is associated with the only one ED. JoinNonce behaves in a similar manner as DevNonce. After that, JS generates the keys using stored Network ID (NetID), AppKey, received DevNonce and generated JoinNonce.

After keys generation, JS sends AppSKey to AS and responds to NS sending JoinNonce. The NS forwards the message as JoinAccept-message to ED, it will contain JoinNonce, NetID and DevAddr assigned to the ED by NS. ED derives NwkSKey and AppSKey using aes128_encryt algorithm as defined in RFC4493.

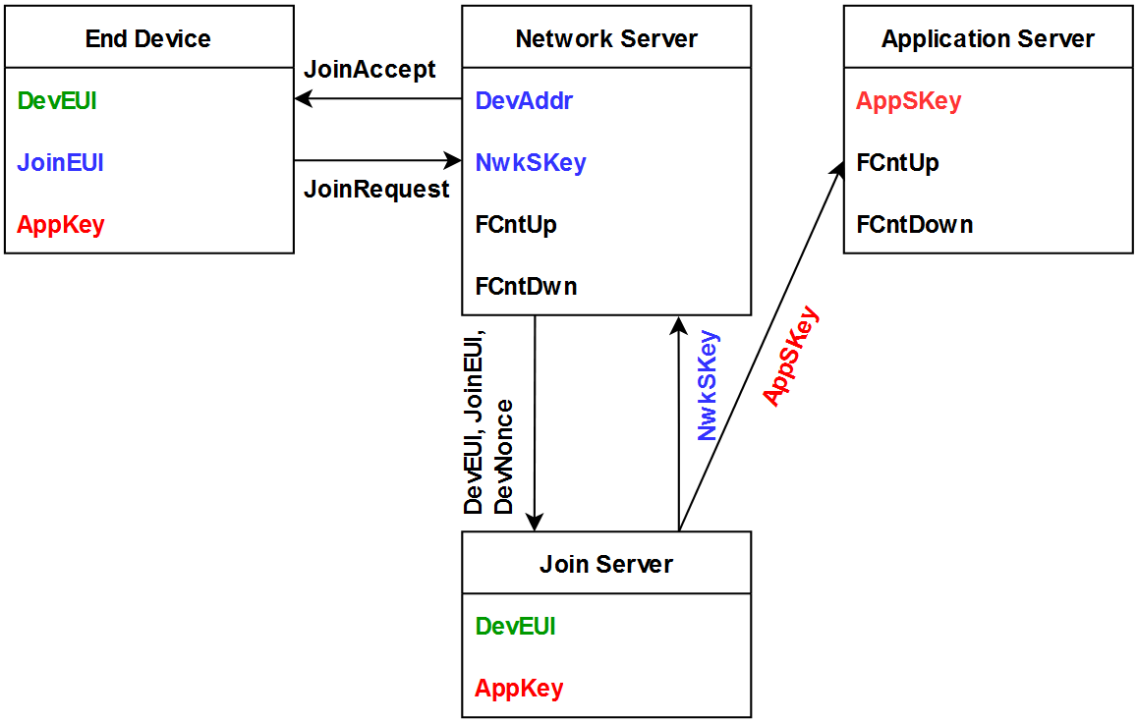


Figure 3.10: LoRaWAN 1.0 OTA. Keys in blue colour are used on ED, JS and NS only, the key in red is used on ED, JS and AS only, the key in green is used by ED and JS only.

In LoRaWAN 1.1, ED and JS are configured with two root keys (Figure 3.11).

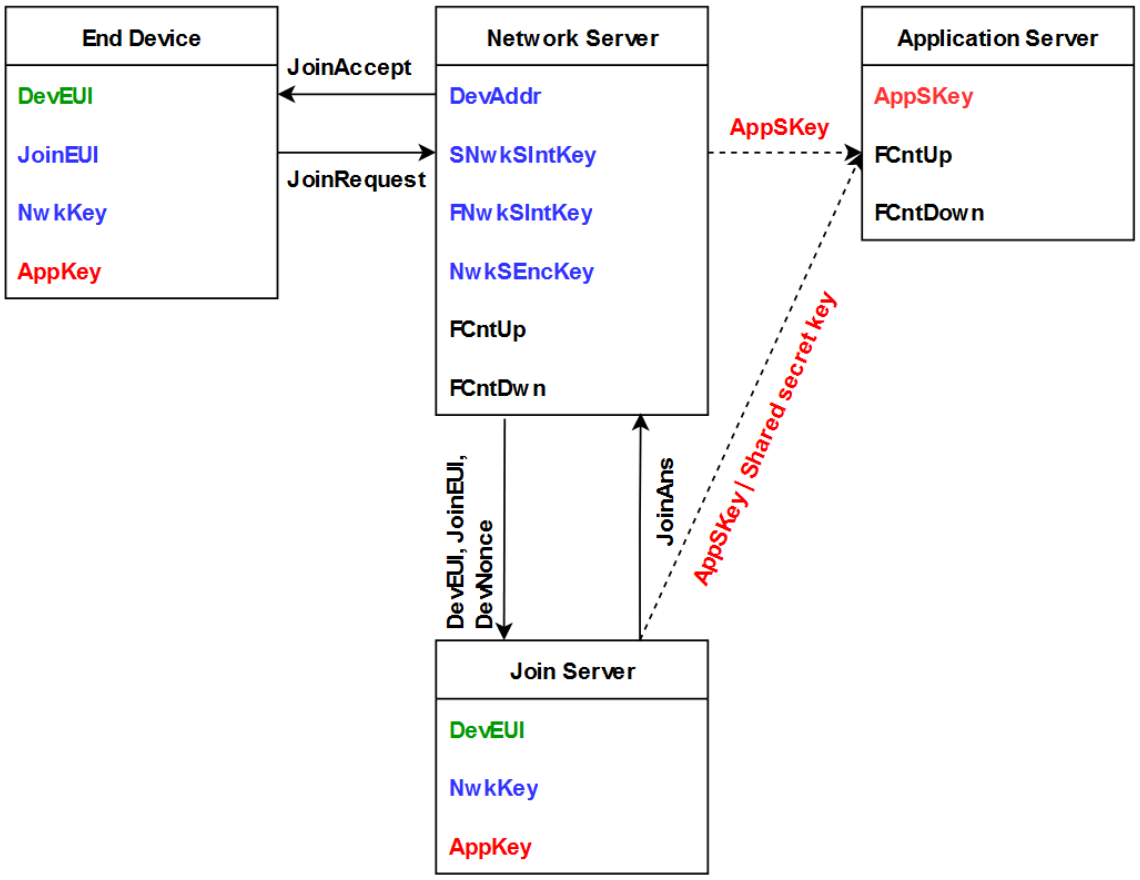


Figure 3.11: LoRaWAN 1.1 OTA. Keys in blue colour are used on ED, JS and NS only, the key in red is used on ED, JS and AS only, the key in green is used by ED and JS only.

In case of protocol version 1.1, the join procedure starts in the same way as for LoRaWAN 1.0 with ED sending JoinRequest. In the end of the procedure, NS sends the same JoinAccept to the ED, but the ED derives keys FNwkSIntKey, SNwkSIntKey, NwkSEncKey from NwkKey using JoinNonce, JoinEUI and DevNonce. AppSKey is derived from AppKey using the same additional parameters.

In this process, however, there is an alternative way to transmit AppSKey to the application server. An important prerequisite in this case is that there is an additional secret key shared only between AS and JS. When ED sends its first uplink, it also includes AppSKey. On reception, AS decodes the AppSKey using the secret key shared with JS. Alternatively, AS may request application session key from JS, then AS decrypts the AppSKey using the shared key [13, 11].

It is also important to specify that during data exchange NS cannot decrypt content addressed to AS because it does not have AppSKey to do it.

In general, as the procedure involves association of secret keys to unique devices and advanced encoding algorithm, the data transmission can be considered as safe. Besides, using of “Nonce” values makes exchange protected against replay attacks and use of ED duplicates. Therefore, if of some reason an ED is compromised, it will not give attackers access to other devices and network credentials.

Nonetheless, the process of activation is secured only in case if attackers do not get access to the root keys and JS. Besides, distribution of the shared between JS and AS key is not described in the documentation and may be source of threat for network.

3.10 Planning of LoRaWAN deployment

Before deployment of equipment, one must consider factors related to the performance of the network, specifically properties of the physical layer of LoRaWAN.

Distribution of radio signal may not be as desired if the terrain or other physical obstacles, as buildings, do not allow the signal to reach receivers. Antenna hight of GW and antenna quality of ED also play significant role in communication. The higher antenna of GW, the better coverage it provides.

Besides, if there is a need to have geolocation of ED that are not equipped with GPS, it can be estimated by the network using TOA. For this approach an uplink must be received by three or more gateways. Then using time difference between reception of the packet NS can estimate location of the ED. This approach requires precise clock on GW as well as on ED [23].

4 Device configuration

During this project, three LoRa compatible instruments were available. Two of them are sensors and one is a network test device. All the instruments were supposed to be used on the USN campus. However, because the study program is online, and most part of the work has been performed in another location, the test device and one of the sensors have been moved to another city for convenience. Still, they were considered as located on the campus, where they will continue working after the project is finished.

There is a big diversity of sensors using LoRa modulation. They come from different manufacturer and have various configurations and interaction interfaces. This chapter describes how user can recognise and setup sensors provided by Altibox and manufactured by Adeunis.

The test device is Adeunis Field Test Device (FTD) (Figure 4.1). FTD main purpose is to examine quality of signal in the location of the device and provide additional information, such as temperature, coordinates and condition of the device [24].

Figure 4.1: Adeunis Field Test Device: 1. Change screen button, 2. Trigger UL button.

To analyse signal from GW in the area, one can read values from the screen. It shows Up- and Downlink characteristics on separate lines, “UL” and “DL” are numbered and frequency for

their reception and transmission is displayed. Other values indicate spreading factor (SF), power used for transmission for UL and Received Signal Strength Indicator (RSSI) for DL. The last value is SNR. For UL it is provided by the GW to the device, for DL – FTD makes analysis itself.

To see other values on the screen, user pushes button marked with 1 in the figure above. Besides, it is possible to trigger an UL sending from FTD by pushing button marked with 2.

Another device used in this project is Adeunis Smart Building Comfort sensor (Figure 4.2). The device measures temperature and relative humidity in range from -20 to +60 °C and from 10% to 90%, respectively, and is designed for indoor use [25].

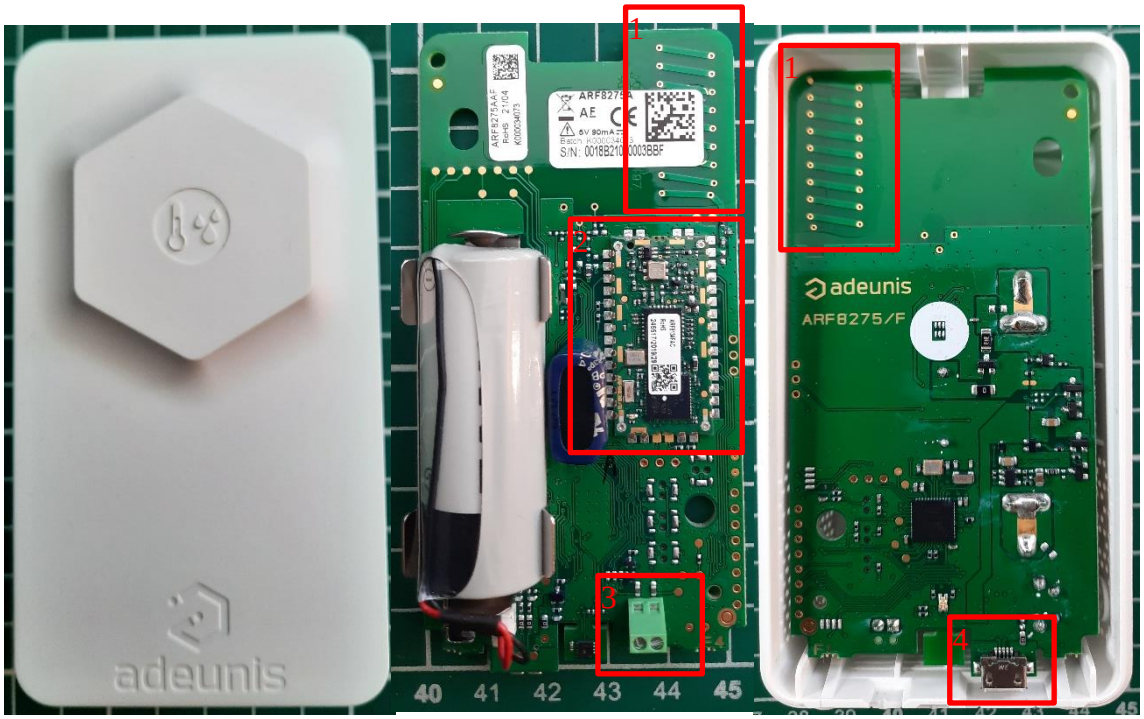


Figure 4.2: Adeunis Smart Building Comfort sensor: 1. Antenna; 2. Radio module; 3. Terminal block; 4. Micro USB interface. Other parts of the device include microcontroller and sensors.

The Comfort sensor is also equipped with a button on one side and a terminal block. Both elements may be programmed to send UL when change state.

Another device connected to the network is Adeunis TEMP LoRaWAN. It has two available contacts for connection of probes [26], but only one is currently in use. This sensor can be mounted outdoor.

All the mentioned devices operate in LoRa version prior to 1.1.

For configuration of devices, a dedicated software “IoT Configurator” distributed by Adeunis is used (Figure 4.3). The application works with Adeunis devices designed for both LoRaWAN and SigFox protocols. There is a list of available models to set up and options for connecting the application to a device. Connection is performed with USB- micro interface.

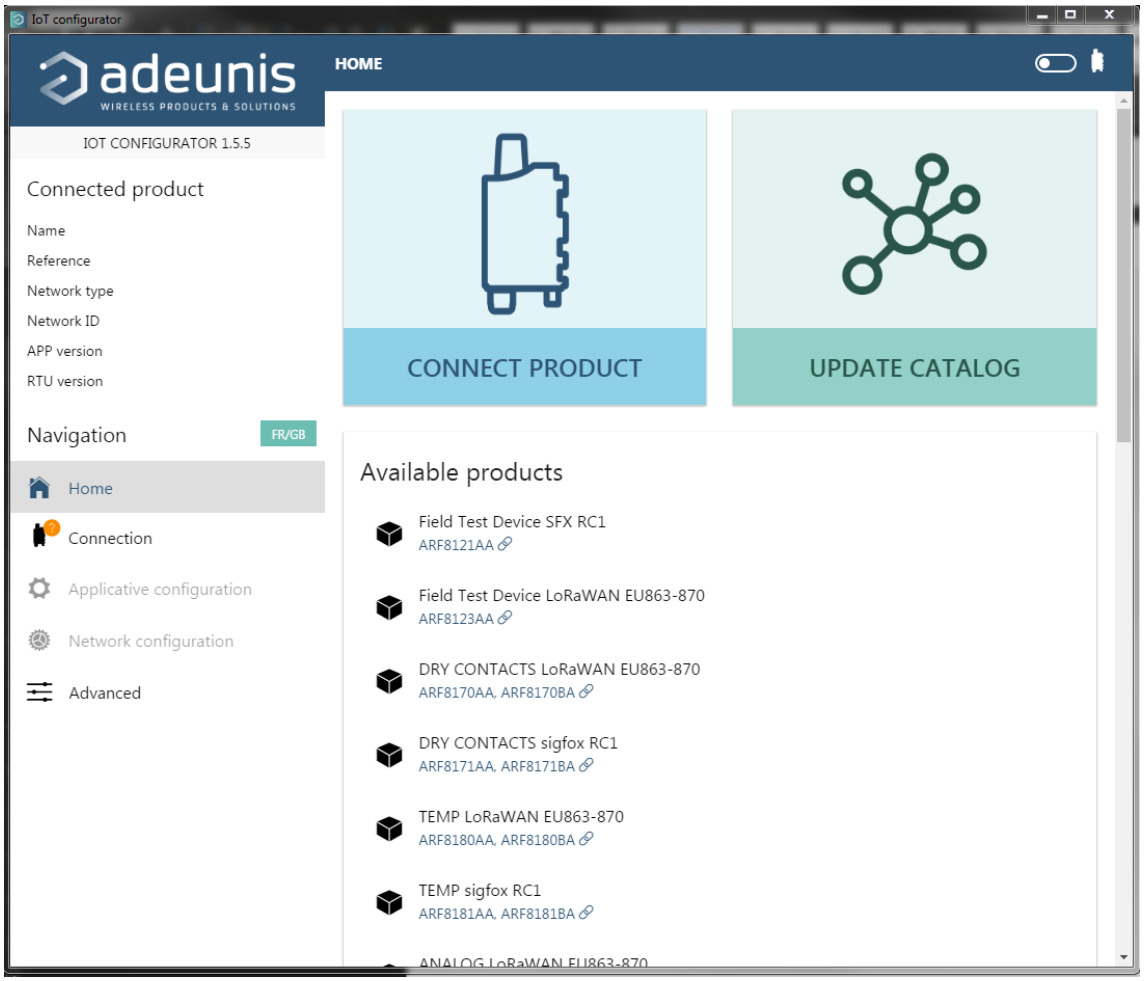


Figure 4.3: IoT Configurator: Home window.

After successful connection, one can observe parameters describing the device. Among those are name, network type and DevEUI, which is presented as a “Network ID” (Figure 4.4).

Figure 4.4: IoT Configurator: FTD successful connection.

In this state, the specific configurations related to network layer and application layer can be investigated and modified.

Functionality at application layer depends on the type of the device and the type of data it may provide. In case of FTD (Figure 4.5), one can activate or deactivate transmission of location (GPS), accelerometer data and a specific message. One can also specify the class of the device and how often FTD should send data. The period is specified in seconds., and the maximum value is 24 hours (86400 seconds). FTD in this project had been set up to send UL with its location every 600 seconds. When UL is translated, it also includes ambient temperature measured by the device.

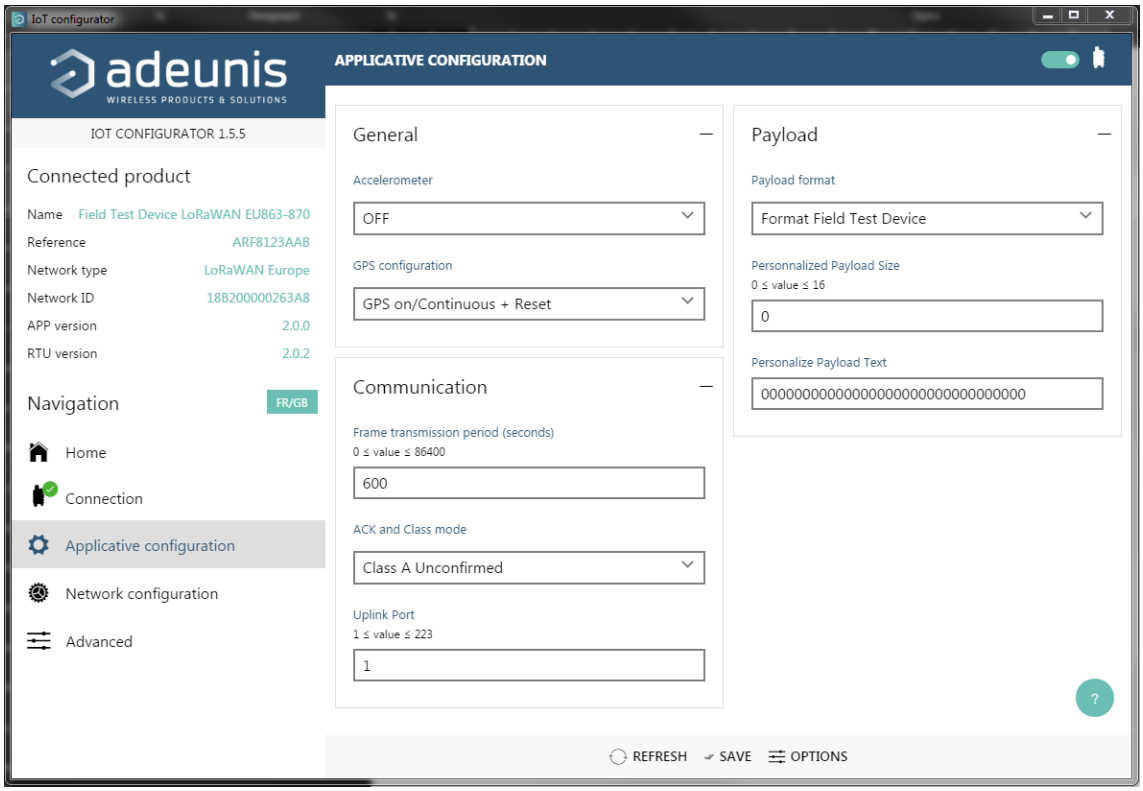


Figure 4.5: IoT Configurator: FTD application configuration.

In the network configuration tab, a user may choose the way of activation of the device, activation over the air in this case, and use of ADR. Then one specifies AppEUI, that is JoinEUI in the newer LoRaWAN version, and AppKey (Figure 4.6). These keys will be used to derive session keys as described in chapter 3.9. The session keys also may be provided manually if the device will use ABP and skip join procedure.

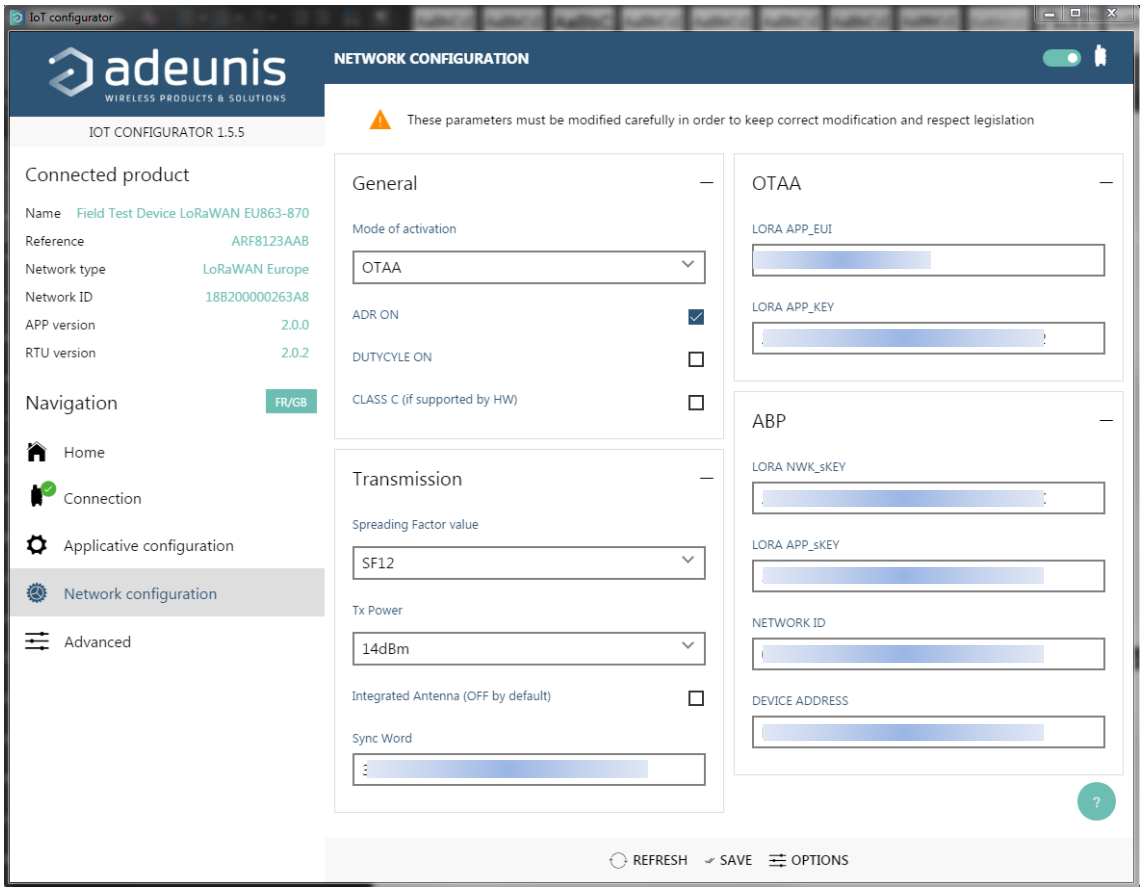


Figure 4.6: IoT Configurator: FTD network configuration.

“Advanced” tab allows do view common and specific information about the device (Figure 4.7). “Version” button returns devices firmware versions, and “List” button returns content of the functional registers that control devices behaviour [24]. It is also possible to view the log saved on the device, the log is continuously updated and provides information about UP such as SF, frequency, power used for transmission and SNR. For DL there is SF, frequency, RSSI and SNR. In the end, the log contains Packet Error Rate (PER), which demonstrates a relationship between transmitted and received packets.

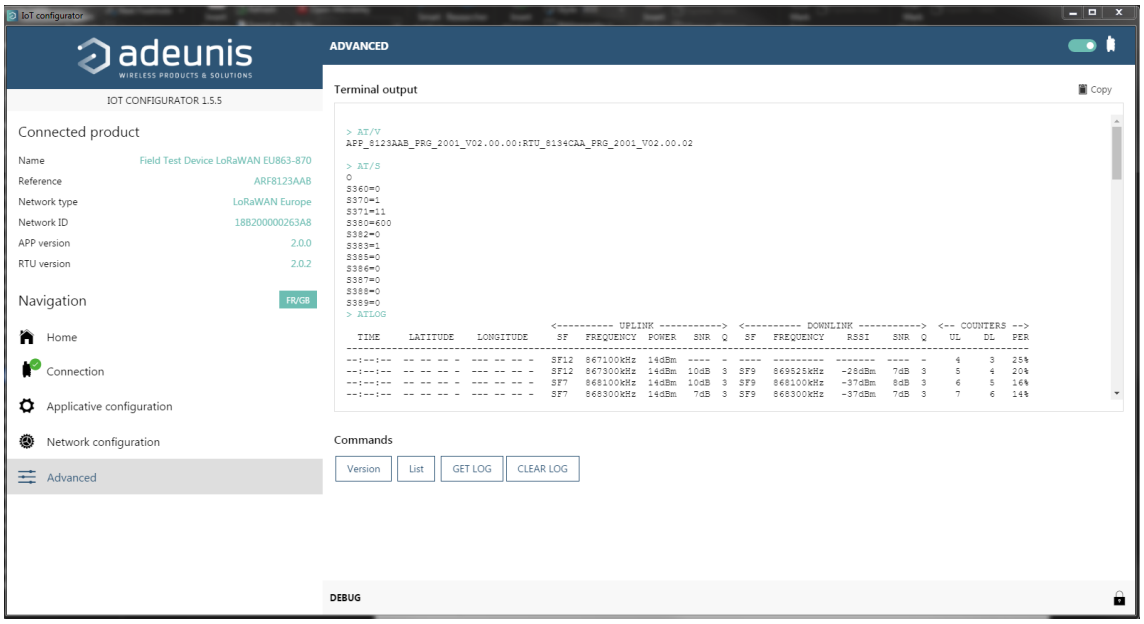


Figure 4.7: IoT Configurator: FTD advanced tab.

The application configuration for Comfort sensor is significantly different from FTD (Figure 4.8). In General settings, one can choose between three modes: Park, Production and Command. Park mode turns the device into a standby state. In Command mode, registers of the device can be accessed and modified. This is performed when the device is connected to a computer with USB cable, then, configuration is done with IoT Configurator or AT commands. In Production mode, the device operates as normal, taking measurements and sending data [26]. As one can see, the Comfort sensor in this case is in Production modetaking samples every $300\ seconds * 2 = 600\ seconds$, or once in 10 minutes.

Further, a Dry contact sensor may be connected to the terminal block shown in Figure 4.2. No time zone configurations have been implemented, and timestamp is displaying time in UTC.

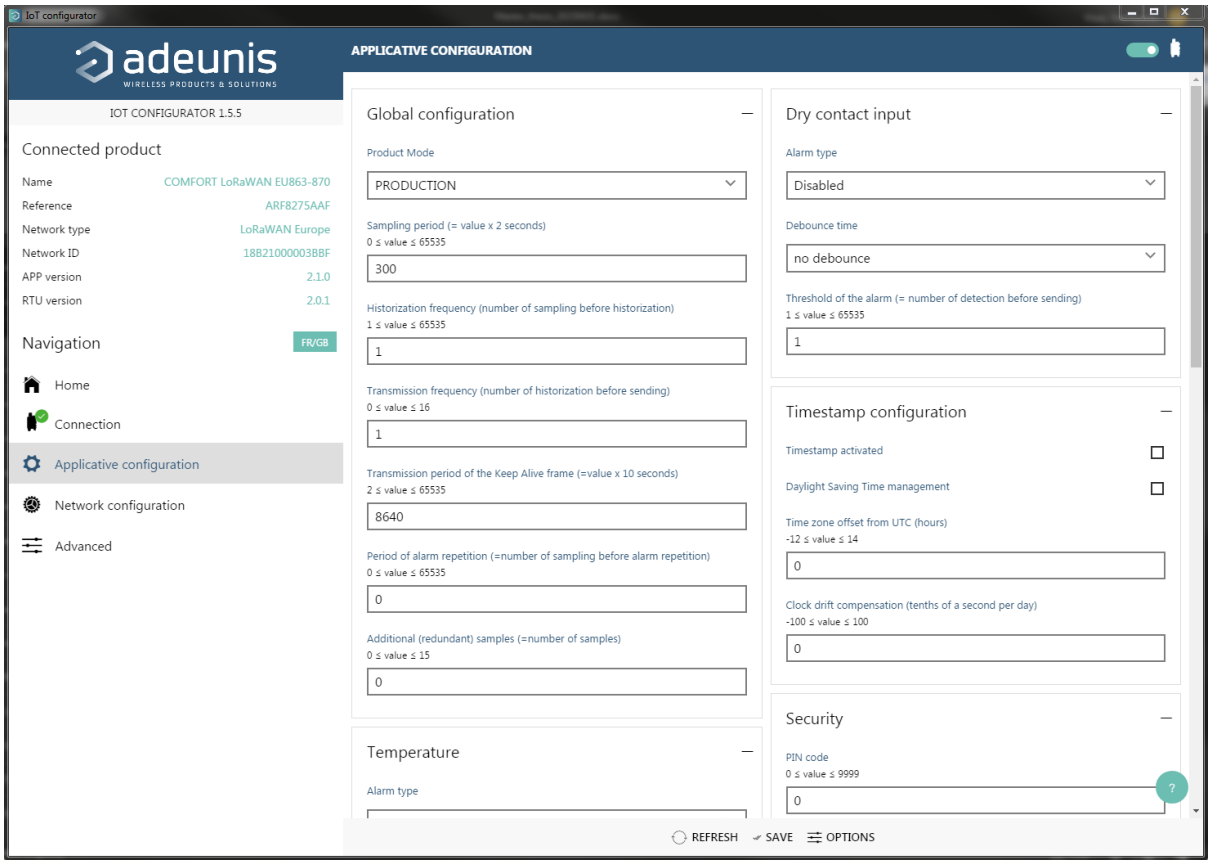


Figure 4.8: IoT Configurator: Comfort application configuration.

The device may be configured to send UL when a level of measured parameter exceeds or fall below a defined threshold, what triggers an alarm. In the current configuration, the temperature measurement are not set to trigger any alarm (Figure 4.9), similar for relative humidity (Figure 4.10). These data are only sent following the time schedule. There is also possibility to set a PIN code that will be used with AT commands to configure the device and increase security level. With zero value, the PIN code is deactivated.

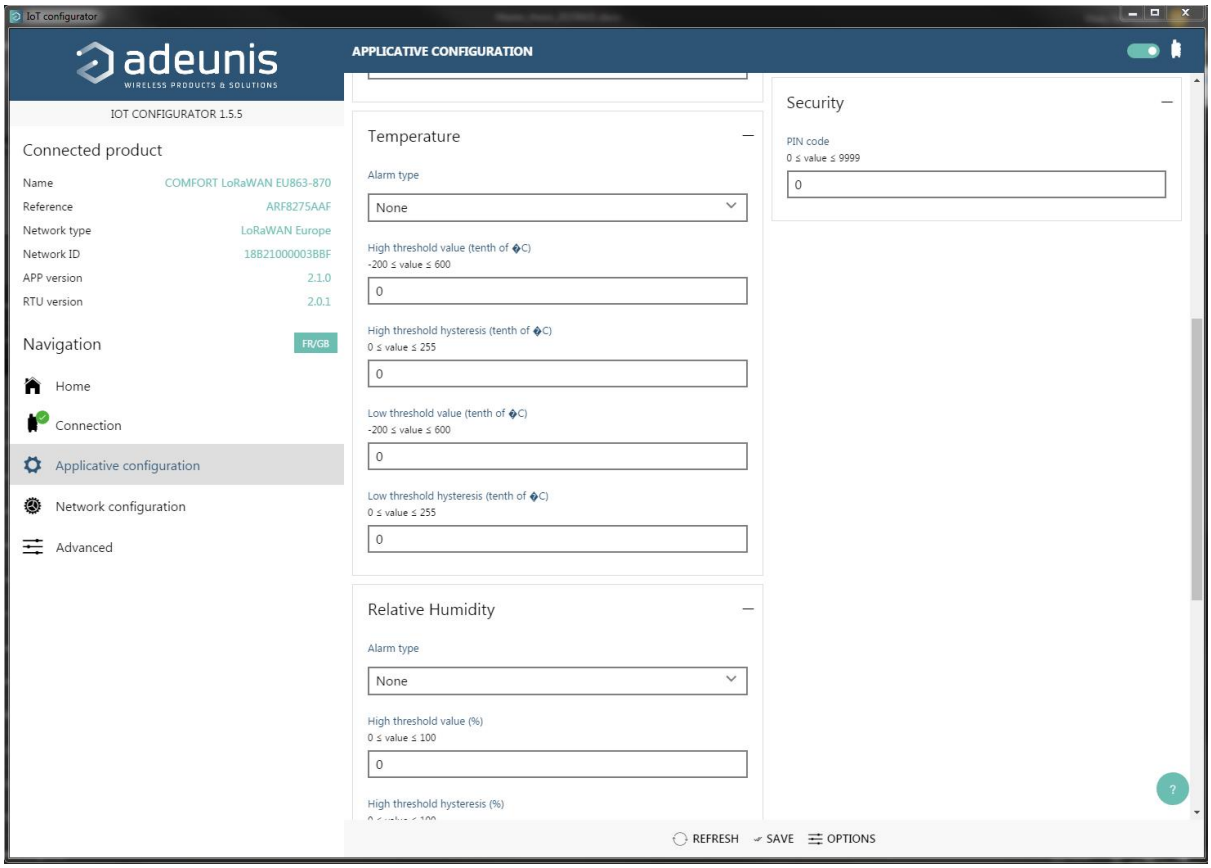


Figure 4.9: IoT Configurator: Comfort application configuration, Temperature, Security.

Another available for interaction component of the Comfort sensor is button. The button is located on the side of the body and triggers sending of UL after the defined number of activations. In this case the sensor sends a message every time the button is pushed (Figure 4.10).

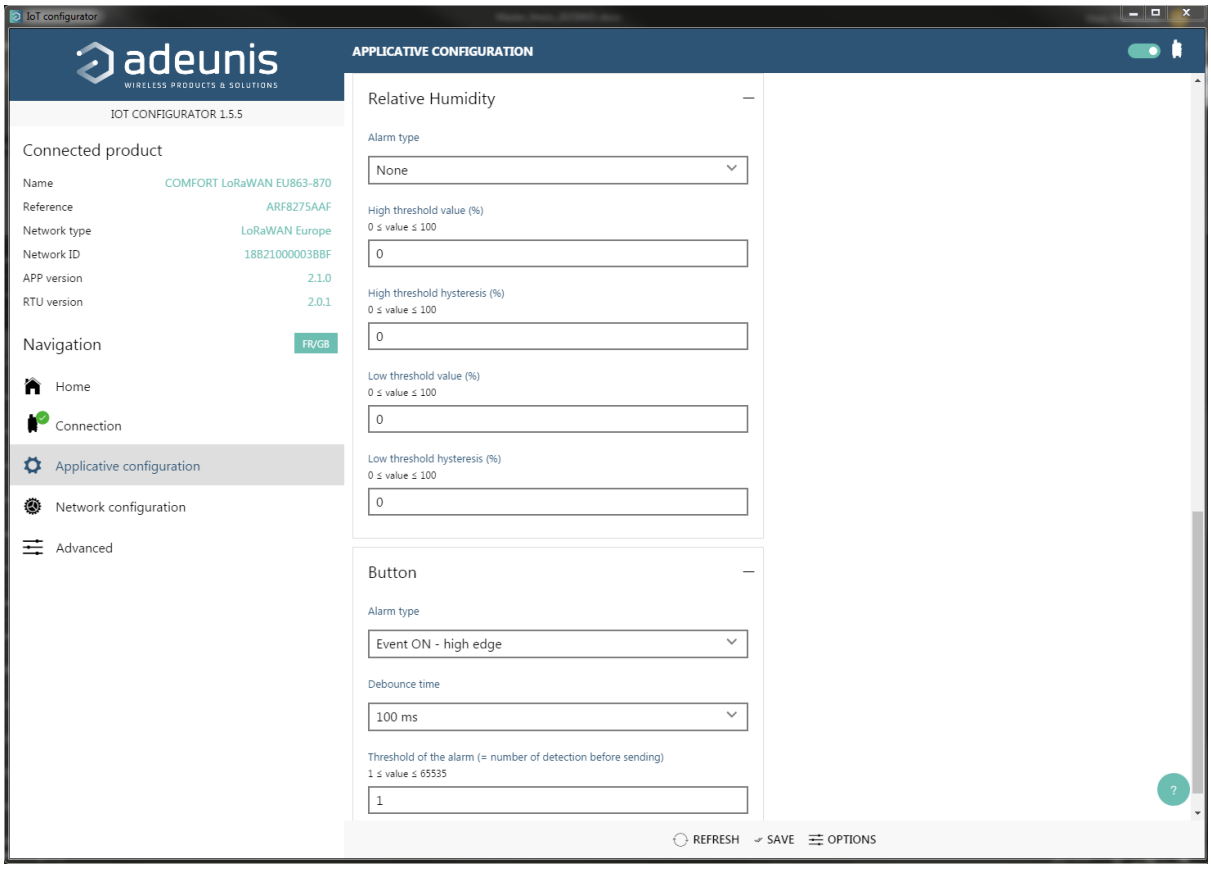


Figure 4.10: IoT Configurator: Comfort application configuration, RH, Button.

Network configuration of the Comfort device slightly differs from the FTD (Figure 4.11). In this case, Over-The-Air Activation (OTAA) is enabled and values for AppEUI (JoinEUI) and AppKey are defined. However, now session key is stored in the device, that means that at the moment no ABP is available at all.

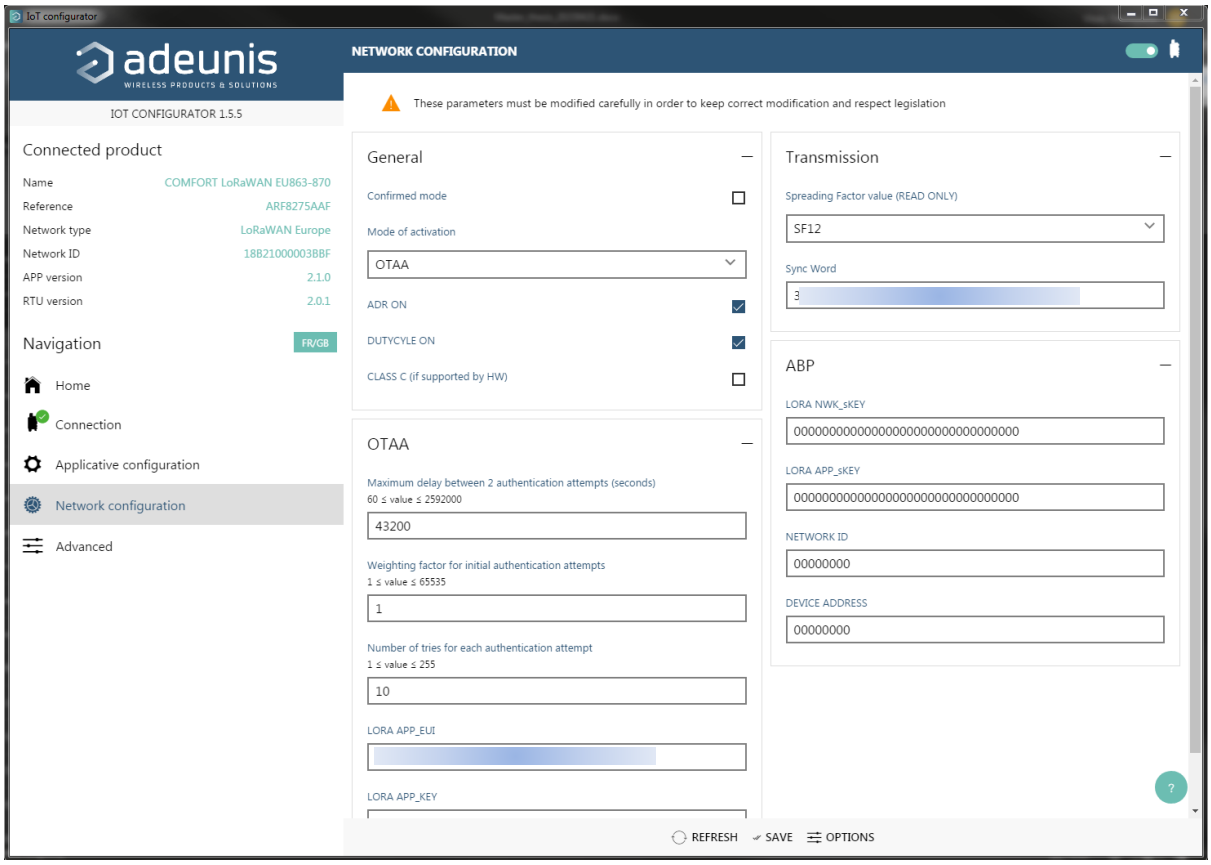


Figure 4.11: IoT Configurator: Comfort network configuration.

The last tab, advanced, provides also device specific functionality (Figure 4.12). One can check the version of firmware and values stored in functional registers as previously. In addition, user can check and set time on the device. However, no logging is available.

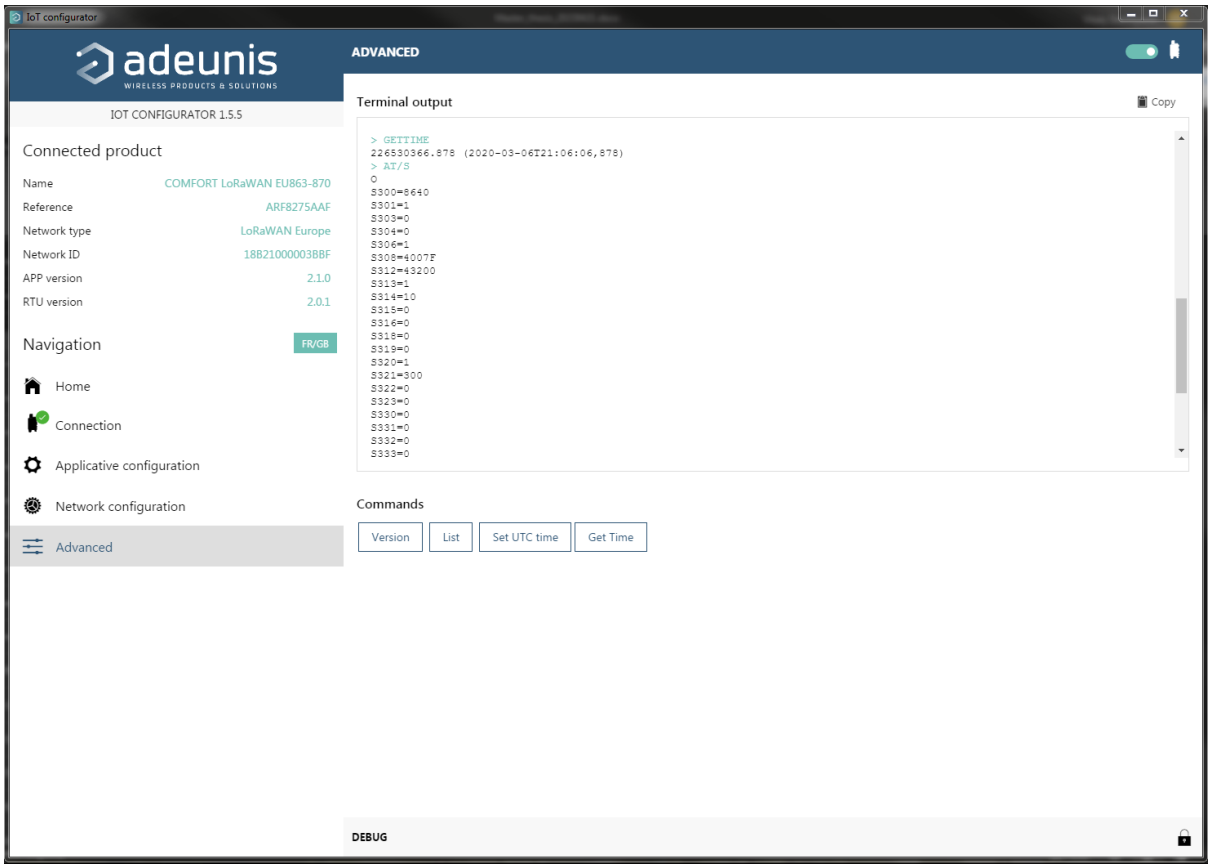


Figure 4.12: IoT Configurator: Comfort advanced tab.

5 Altibox LoRaWAN system

As it has been shown in chapter 3, LoRaWAN consists of several servers in addition to EDs and GWs. All the elements are connected into a network that is managed by an operator. There are public, shared and private networks. The operator may decide what type of the network it will be. For example, there is a network server provided by LoRa Allicance for free, one can use own GW and ED with it. In this case, the user should register devices in the service and manage them self.

For the current project a public network provided by Altibox has been used. One requires a subscription to access it. In addition, all EDs and GWs must be first approved and registered by administrator of the network, before they can be used.

5.1 ThingPark platform

The available LoRa compatible devices have been managed using platform provided by Altibox AS. The platform, ThingPark, has been developed by Actility and includes several functional blocks (Figure 5.1).

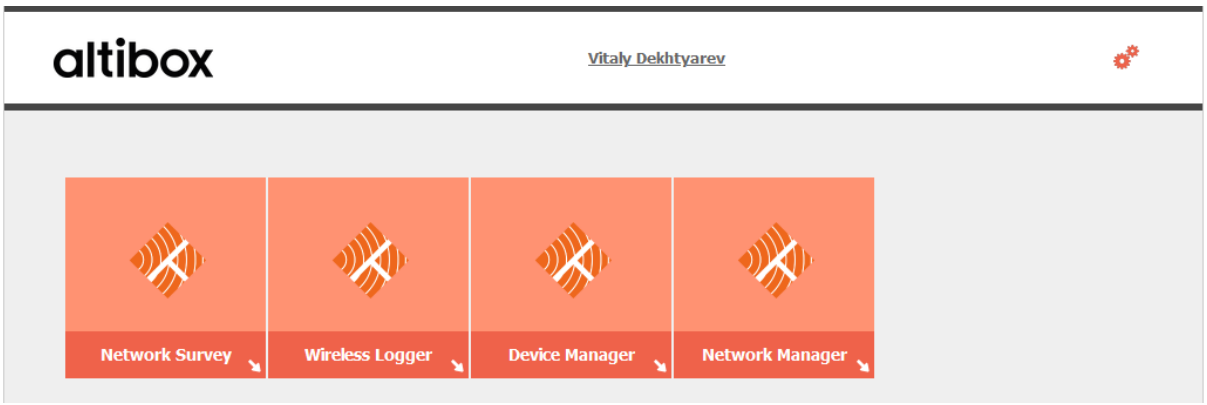


Figure 5.1: ThingPark main window.

Network Survey depicts location of the active devices and properties of the signal of the area they operate in, if ED provides own location. Wireless logger gives overview and content of packets transmitted within the network for available devices. Device manager (Figure 5.2) provides functionality for registering, editing and removing devices and their specification from the network. Network manager gives an overview of the registered GWs.

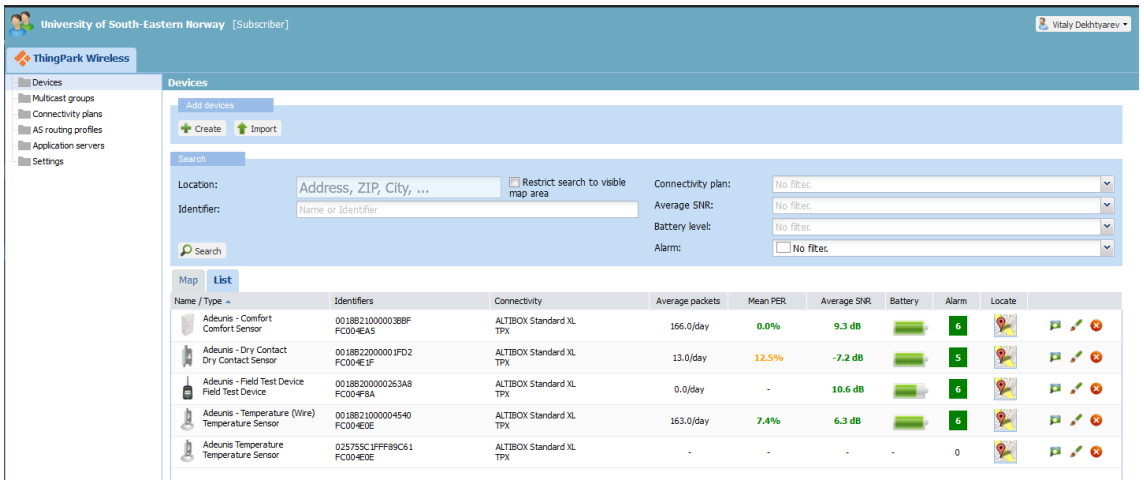


Figure 5.2: ThingPark: Device manager.

When ThingPark is investigated from LoRaWAN structure perspective, this system may be viewed as a combination of JS, NS and AS. This way the network is easy to maintain.

Wireless logger (Figure 5.3) provides options for filtering messages with respect to device number or type of messages. It is also possible to investigate particular packets with MAC commands or transmitted during join procedure. In addition to the raw content, user may observe decoded content defining the type of decoder. Each packet also shows network properties, so that it is possible to examine performance of ED in different locations and move them or add GWs to improve radio transmission.

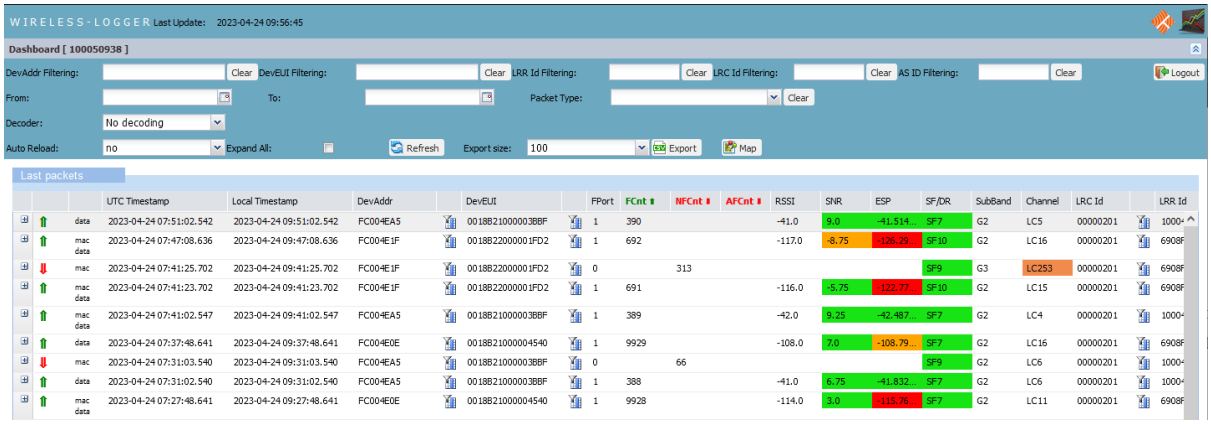


Figure 5.3: ThingPark: Wireless Logger.

While the Wireless logger may decode a message from ED and provide technical insight, nevertheless, the full content must be retrieved using other AS. An alternative AS may be specified in Device manager. In “Application servers”-tab, the user gives URL of a server that receives packets and format of the packets. One may also specify additional custom HTTP headers, e.g. related to security check (Figure 5.4). Example of a simple AS maybe a webhook. The created AS should then be activated in “AS routing profiles”-tab.

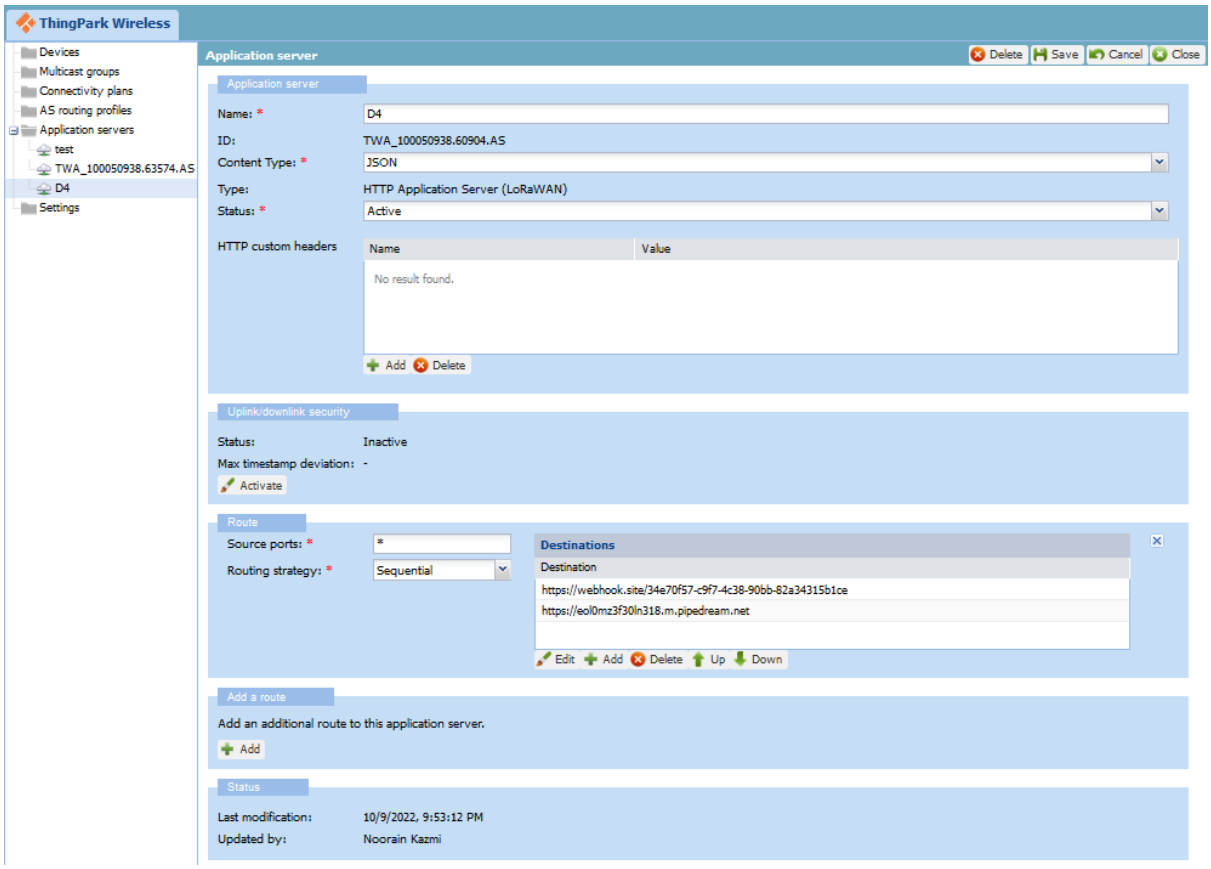


Figure 5.4: ThingPark: Application Server specification.

A message that will be received by webhook in JSON format is shown in Figure 11.1 **Error! Reference source not found.**in Appendix B. One can observe network parameters, such as DevEUI, if ADR is activated, DevAddr, RSSI, SNR, SF (SpFact), Frequency and data from the ED in payload part. The data is decoded, but may also be retrieved from payload_hex attribute and decoded by user according to the technical documentation for ED.

5.2 MQTT broker

One of the protocols often used with IoT is MQTT. To investigate interaction between LoRaWAN infrastructure and the protocol, an account on publicly available MQTT server HiveMQ has been created. The server provides broker for creating topics, subscribing and publishing messages in them. Detailed setup of the broker and the application is presented in Appendix C.

5.3 ThingParkX platform

Altibox AS has provided access to an additional functional element that can be used in LoRaWAN ecosystem managed with the services from Acility - ThingParkX. ThingParkX (Figure 5.5) is a platform that is used instead of AS profile specification in Device Manager (Figure 5.4). One should specify TPX as an AS profile to manage data exchange with own servers through connections of various types. Existing connections and option to create a new one is available from the connections tab (Figure 5.6 and Figure 5.7).

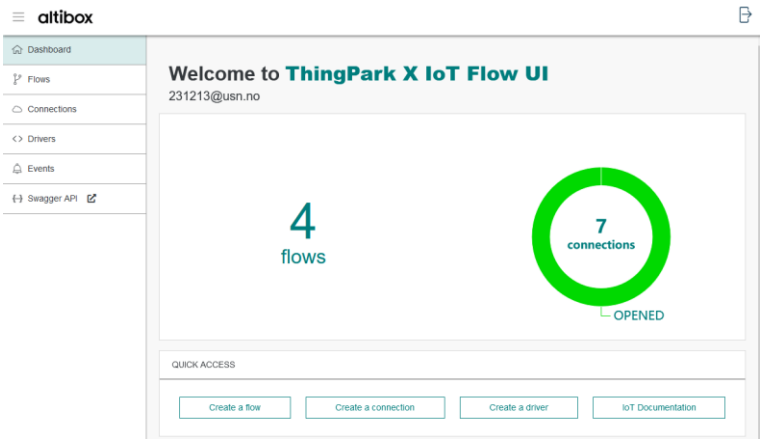


Figure 5.5: ThingParkX: home window.

When connection is created it is added to a Flow, the Flow in turn may be assign to a particular ED identified by DevEUI. Moreover, several connections may be combined in one Flow, such that one ED may transmit data to several AS. Nevertheless, it is necessary to pay attention to settings made for a connection, it may not be applicable to other ED than it is setup for.

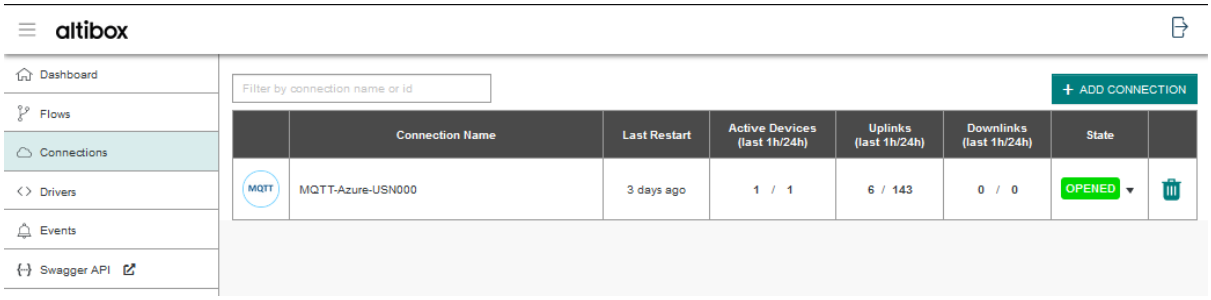


Figure 5.6: ThingParkX: list of created connections.

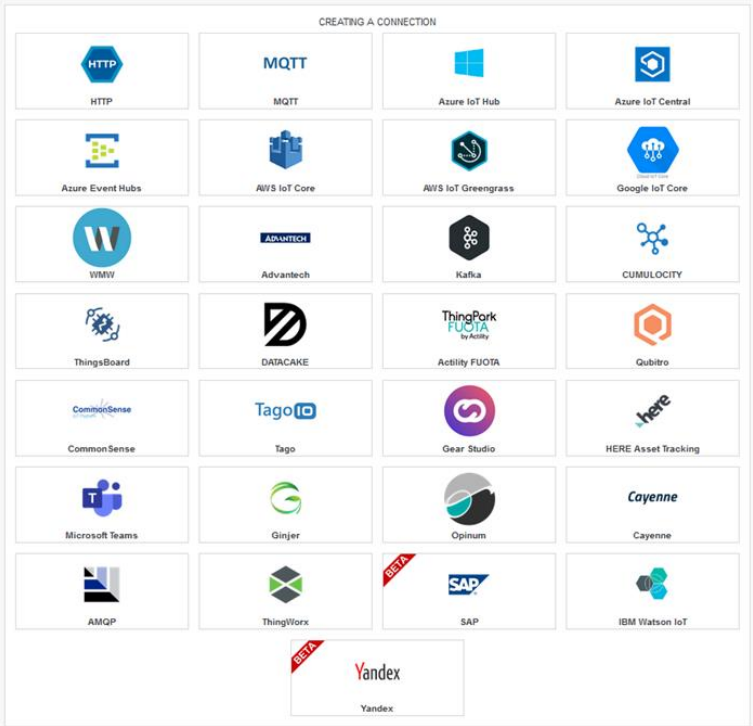


Figure 5.7: ThingParkX: connections.

5.3.1 MQTT connection

Connection can be created using button “+ Add connection”, or an existing connection can be modified. As an example, it is chosen a MQTT connection. User chooses MQTT from the list of available connections and specify required parameters (Figure 5.8). It is necessary to give a name to the connection.

Then “Basic Settings” include credentials of the MQTT broker: URL combined with port number for communication with TLS, username, and password.

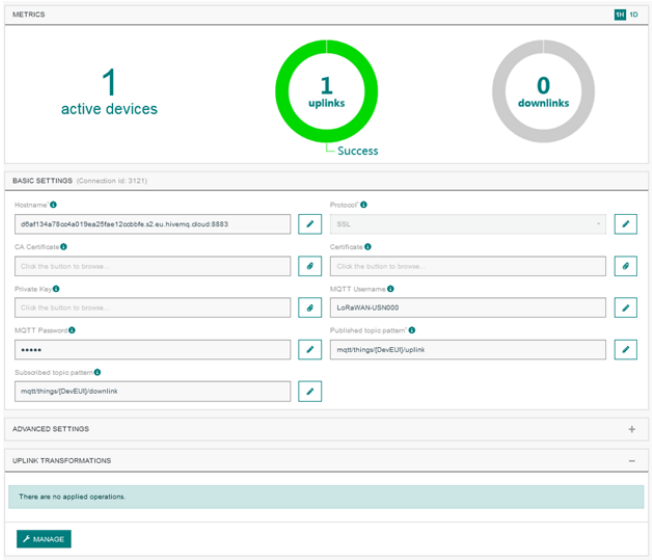


Figure 5.8: ThingParkX: editing existing MQTT connection setup.

Topics for subscription and publishing are given by the platform but may be modified.

In the top part of setup section there is a statistical information showing how many devices use that connection and how many successful and unsuccessful transmission have been performed for UL and DL. This statistic is updated only after the connection had been assigned to a Flow and used for transmission, i.e. activated.

5.3.2 Message transformation utility

A significant advantage of the platform is a possibility to modify packets transmitted by devices before the packets are forwarded. This may be done in “Uplink transformation” section (Figure 5.8). After activating the “Manage”-button, user is given a pop-up window with several options for transformation (Figure 5.9). A recommended option has been chosen, operation with JSLT that is a language for transformation of JSON objects.

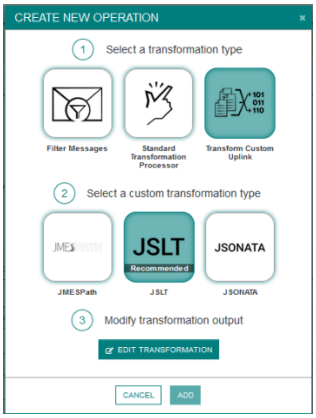


Figure 5.9: ThingParkX: UL transformation options.

After choosing the mentioned option a new webpage appears. It contains three sections: “Raw uplink”, “Transformation”, and “Result” (Figure 5.10). User can write the required operations in “Transformation” section of the new page. If it is critical to check that result is as desired, one must provide sample of UL in the “Raw uplink” section.

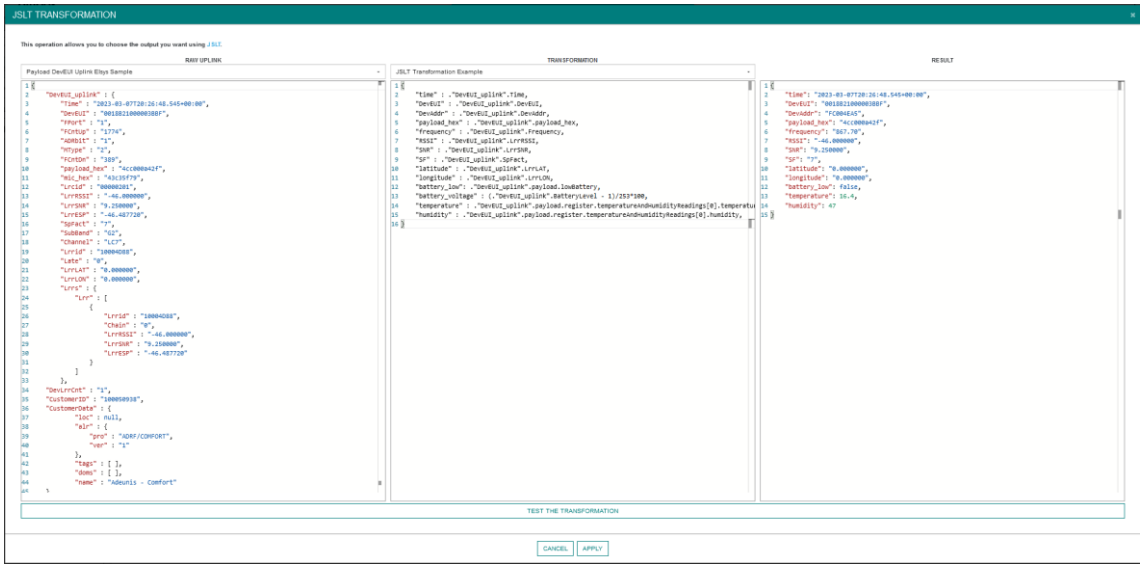


Figure 5.10: ThingParkX: JSLT interface.

In Figure 5.10 a sample UL from Figure 11.1 is used. Example of transformation of UL from temperature sensor to be published on Hive MQ broker is shown in Figure 5.11. The output message is split into the part with value, type of the value and measurement units, while another part is metadata with content of payload hex string, transmission properties, and battery charge converted to percentage.

```
{
    "DevEUI" : .DevEUI_uplink.DevEUI,
    "signals" : [
        {
            "value" : .DevEUI_uplink.payload.temperature,
            "unit" : "CELSIUS_DEGREES",
            "type" : "Temperature",
            "timestamp" : .DevEUI_uplink.Time,
            "metadata": {
                "DevEUI" : .DevEUI_uplink.DevEUI,
                "DevAddr" : .DevEUI_uplink.DevAddr,
                "payload_hex" : .DevEUI_uplink.payload_hex,
                "frequency" : .DevEUI_uplink.Frequency,
                "RSSI" : .DevEUI_uplink.LrrRSSI,
                "SNR" : .DevEUI_uplink.LrrSNR,
                "SF" : .DevEUI_uplink.SpFact,
                "latitude" : .DevEUI_uplink.LrrLAT,
                "longitude" : .DevEUI_uplink.LrrLON,
                "battery_low" : .DevEUI_uplink.lowBattery,
                "battery_voltage" : (.DevEUI_uplink.BatteryLevel - 1)/253*100
                }
        }
    ]
}
```

Figure 5.11: ThingParkX: temperature sensor UL transformation for HiveMQ.

5.3.3 Flow

After creation of connection one creates a Flow to transmit the data. For that purpose user chooses Flow from the menu to the left and adds a new one with the button “Add” to the right (Figure 5.12).

Flow creation comprises five steps. The first one is giving a name and description to the flow (Figure 5.13). In this case, flow is created for Comfort sensor to publish data on HiveMQ broker.

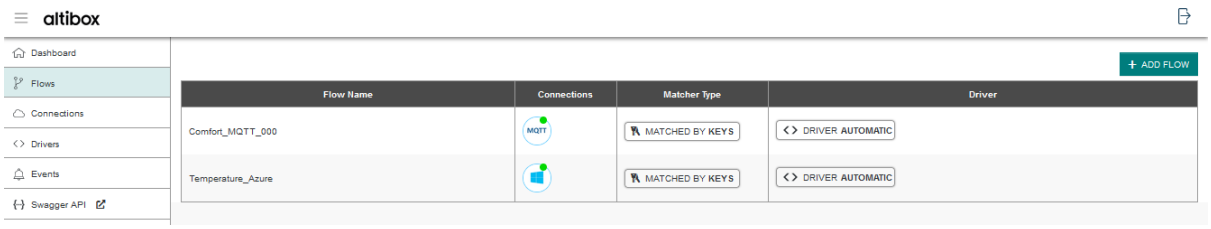


Figure 5.12: ThingParkX: list of existing flows.

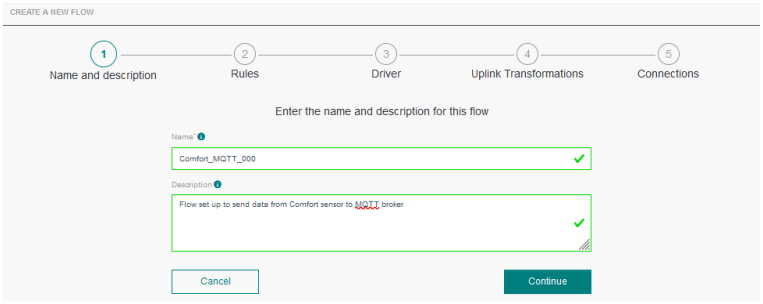


Figure 5.13: ThingParkX: Flow creation step 1.

In step two, it is specified how the flow is assigned to a device. In this case, "Keys" means DevEUI of ED, and the value corresponding to Comfort sensor is specified (Figure 5.14). It is also possible to specify several keys for different devices to be used with the same flow if desired.

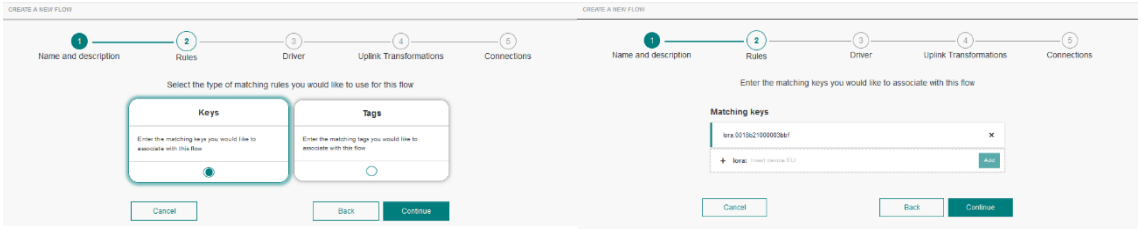


Figure 5.14: ThingParkX: Flow creation step 2.

In the next step, user may choose a specific driver for the Flow (Figure 5.15). Driver is used for decoding UL from ED. If there is no specific requirement to decoding, the user may rely on the drivers provided by Actility.

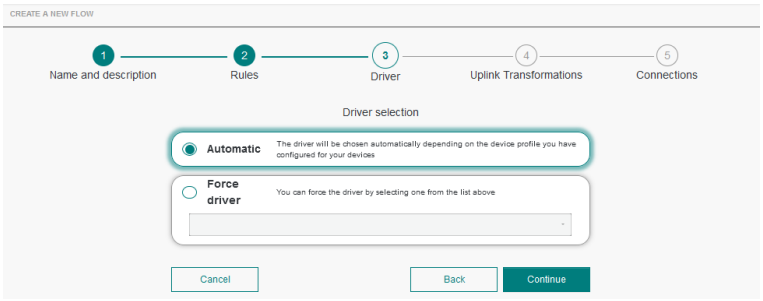


Figure 5.15: ThingParkX: Flow creation step 3.

Further, user may add an UL transformation, but it is recommended by Actility to perform this at connection level.

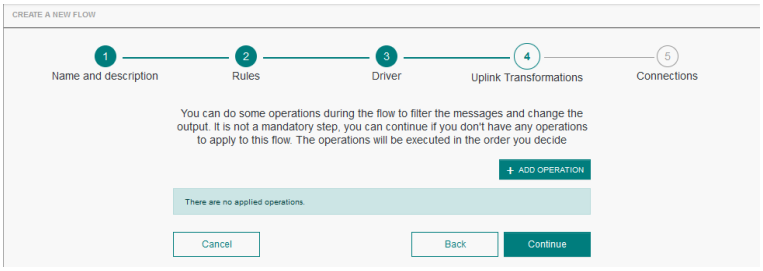


Figure 5.16: ThingParkX: Flow creation step 4.

In the last step user chooses the desired connection, and after confirmation the flow is active. User may choose several connections if it is relevant.

5.4 MQTT connection to Dimension Four

One of the destinations for data from available ED is Dimension Four platform. The platform provides Software as a Service (SaaS). User can register account for free and setup a tenant that has a unique Id (Figure 5.17).

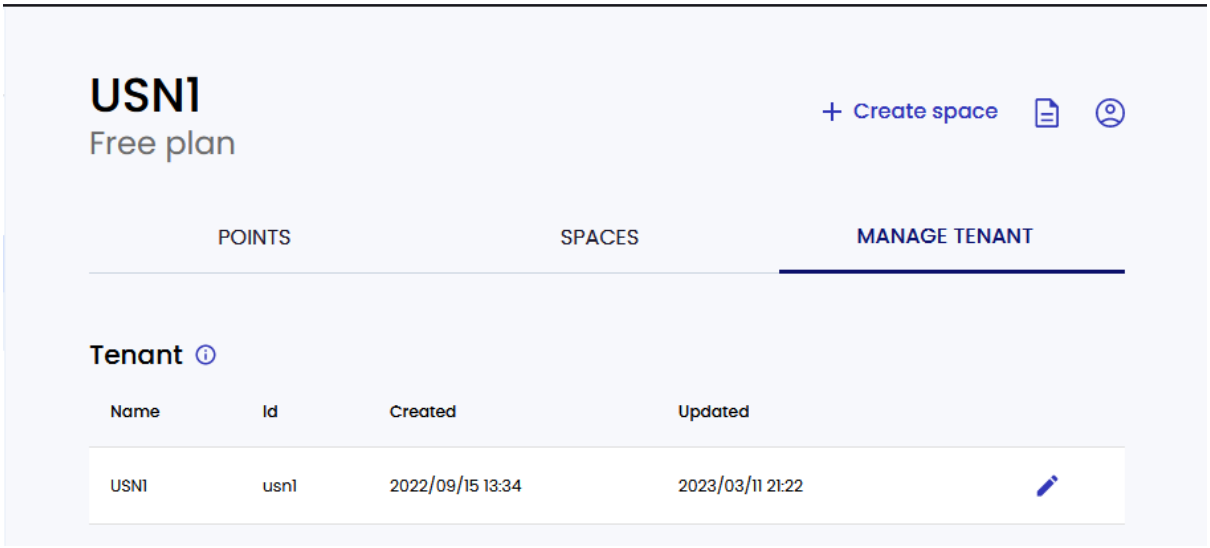


Figure 5.17: Dimension Four: Tenant overview.

The tenant may be further divided into spaces (“Campus Porsgrunn”, “Weather Station”) and each space into points (“Adeunis Comfort”) (Figure 5.18).

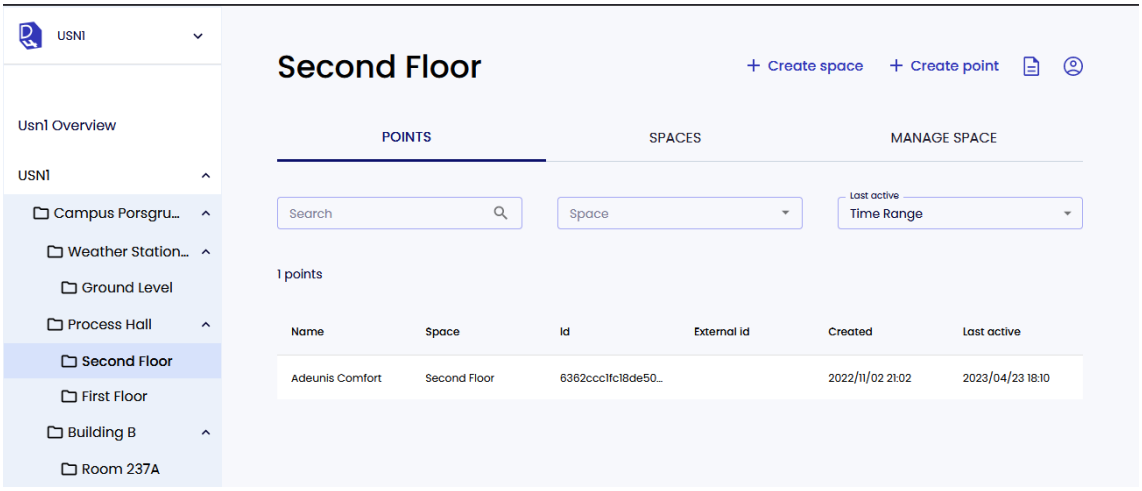


Figure 5.18: Dimension Four: Tenant structure.

A point has a unique id and may represent a device, in this case measurements from the physical device will be stored as signals in the point.

Dimension Four platform provides option to upload measurements to the storage using MQTT protocol (Figure 5.19).

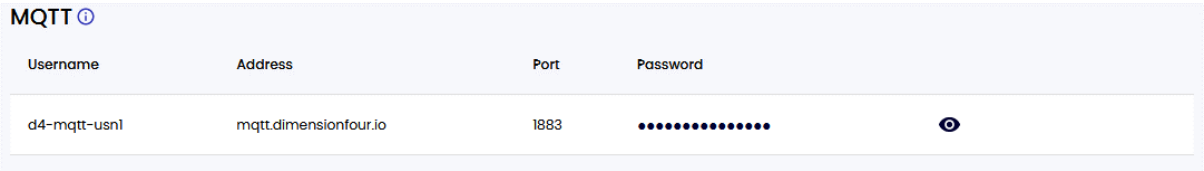


Figure 5.19: Dimension Four: MQTT parameters.

These credentials have been used to establish a connection from ThingParkX to the storage. Detailed setup is described in Appendix D.

One must pay attention to the difference between transformation for HiveMQ and Dimension Four brokers. Transformed messages for HiveMQ must contain “DevEUI” at the root level of JSON object, otherwise, they will not be delivered, while messages for Dimension Four do not need this member.

5.5 Azure IoT Hub connection

Another connection available in ThingParkX that has been investigated is Azure IoT Hub. The IoT Hub is an online service that may perform management of message transmission between applications and devices [27]. The service also provides facilities to monitor particular events on connected devices or implement artificial intelligence algorithms to process incoming data.

Procedure for establishing a connection from ThingParkX to the IoT Hub is described in Appendix E.

To examine functioning of the Azure IoT hub one can use an Azure IoT Explorer application developed by Microsoft and freely available on github.com (Figure 5.20). After straight forward installation one needs to go through several steps to start reading measurements, see Appendix E.

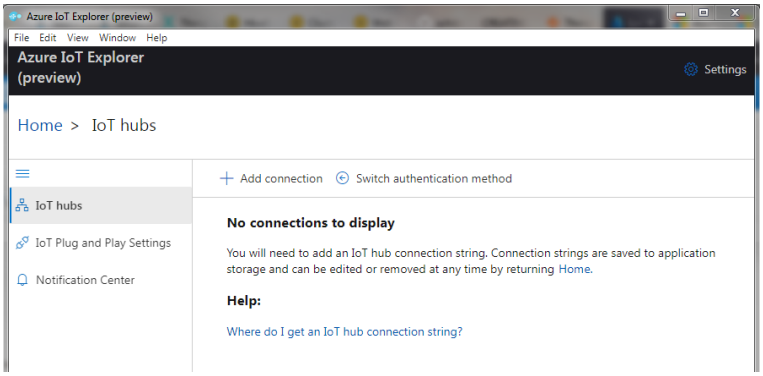


Figure 5.20: Azure IoT Explorer: home window.

5.6 Altibox application server

In the frame of this project, Altibox has provided access to AS developed by niotix (Figure 11.28). The server may be used as an alternative to the described above connections. It provides storage for measurements, visualisation tool and communication with sensors.

The AS implements physical devices in software as Digital Twins, that increases level of abstraction when working with sensors and hardware.

Digital Twins may be nested and configured as one of several type: such as “Default”, “Building”, “Sensor”, “Room”, “Door”, “Zone”, “Floor” or “Trackable”. Each of these may be defined on the provided World map using coordinates.

When one of these types has a defined position on world map, its actual dimensions may be marked as a polygon. Further, in settings for type “Zone”, a precise image of an area may be added to the application. This facilitates easier identification of a polygon by users and relation of it to a real floor or room in a building.

Approximate floor plan of a processing hall at USN has been created (Figure 5.21) during this work.

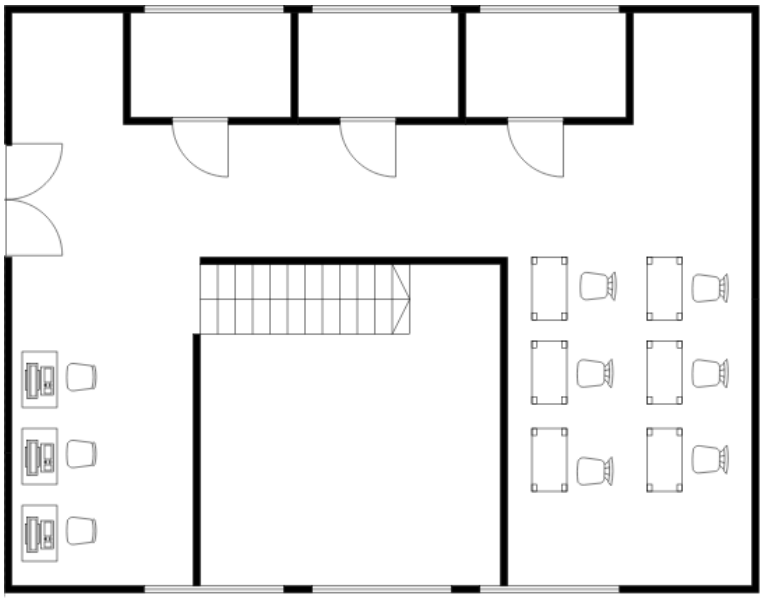


Figure 5.21: Processing hall approximate floorplan.

Access to the niotix functionality may also be managed for different types of users from niotix. There are several permissions available that can be assigned individually to users (Figure 5.22). The permissions include management and reading of separate components of the niotix, such as accounts, Influx database, and data states.

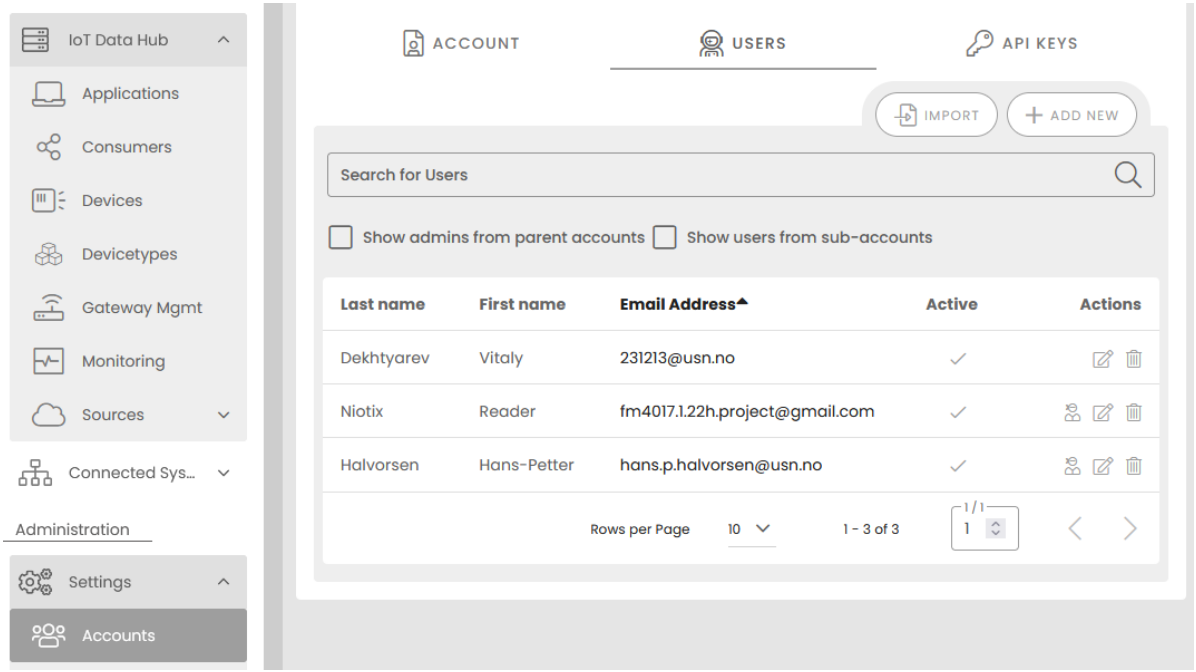


Figure 5.22: niotix: account management.

From the account management tool, it is also possible to generate API keys that may be used with niotix web API for querying data stored by the service. niotix provides a documentation on how to use web API on its official web page.

Detailed configuration of niotix service is described in Appendix F.

5.7 Dashboard with Altibox and Grafana

GrafanaLabs is a company offering various solutions for querying and visualisation of data streams (Figure 5.23). The service is also available in conjunction with niotix as an external tool for visualisation. One may use it in addition to the existing services in niotix.

A significant prerequisite for this service is that it must be enabled by administrator at Altibox for a particular niotix user account.

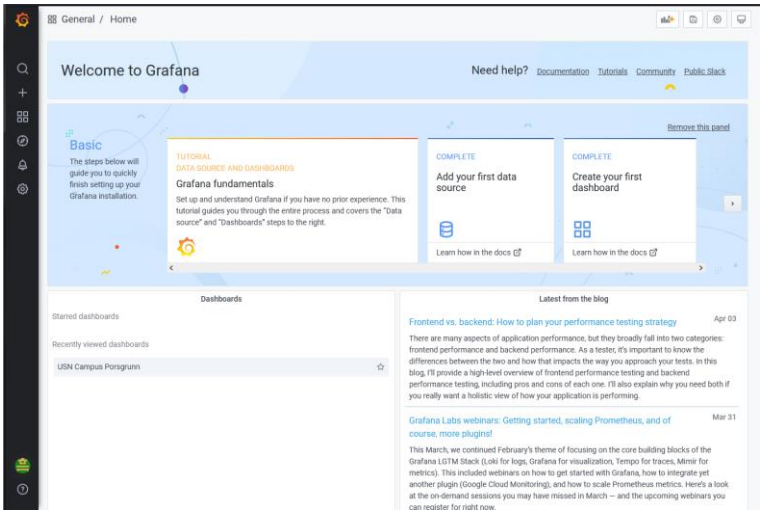


Figure 5.23: Grafana: home window.

The basic building blocks of visualisation in Grafana are dashboards, which can be further divided into panels and rows (Figure 5.24)

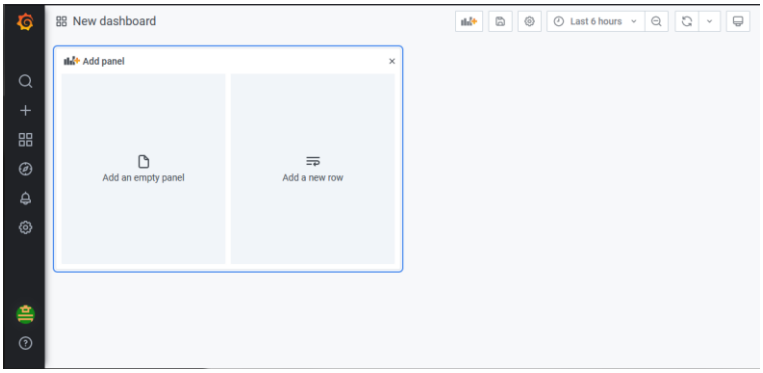


Figure 5.24: Grafana: dashboard creation.

When user choses to create a panel there appear several options for type of visualisation and settings (Figure 5.25).

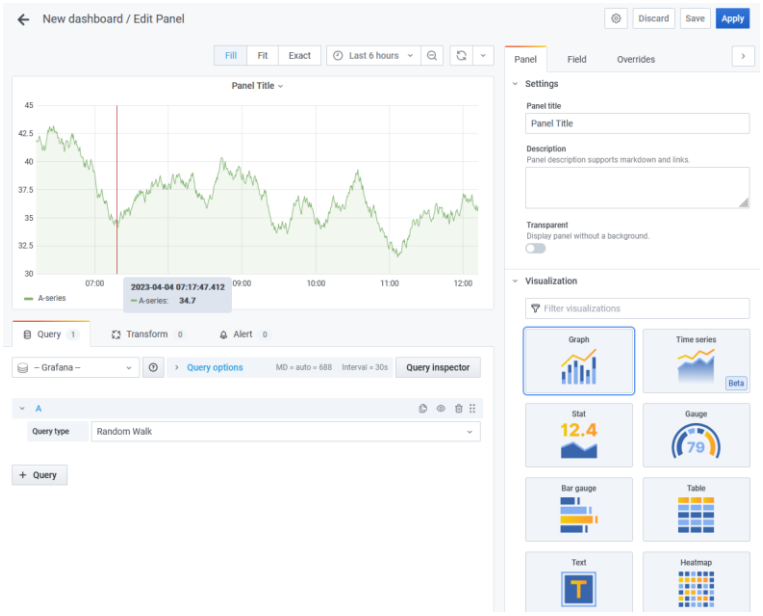


Figure 5.25: Grafana: panel creation.

Each of the provided visualisation methods has several settings for customisation. The most relevant of the used type of data is Graph-visualisation. Detailed description of how to setup a dashboard Grafana is presented in Appendix G.

6 Node-RED monitoring application

If the data forwarded from ThingParkX to connections described in chapter 5 ends up on servers that are not managed by Altibox and do not provide functionality for monitoring measurements as niotix, one may need to set up an additional application. To observe data published to MQTT broker, it is required a subscriber and visualisation tool. Dimension Four allows to store data, but still not to visualise or download it directly. Azure IoT Hub works in a similar way as MQTT broker. For the purpose of retrieving data from those connections, one needs a service allowing the necessary communication and other operations. Node-RED has been chosen to investigate the possibility of alternative to niotix monitoring application. It was chosen because it comes preinstalled on Raspberry Pi, which can be easily integrated in IoT, and contains building blocks for visualisation dashboard.

Node-RED represents a browser-based programming tool. The tool itself is built on event-driven JavaScript runtime Node.js and is suitable for network operations. Because the tool is light-weighted, it may run on low-cost machines. Node-RED has been installed on Raspberry Pi and runs as a server in local network. Interaction with editor is performed in a browser by opening of the webpage with IP of the server and port 1880.

Programming in Node-RED is performed by moving nodes from a palette to the workspace and wire the nodes together (Figure 6.1). The workspace is then representing a flow. Each flow can be exported in JSON-format document and can be saved as a file locally. The flow can then be imported into a new flow from the file using import command in menu.

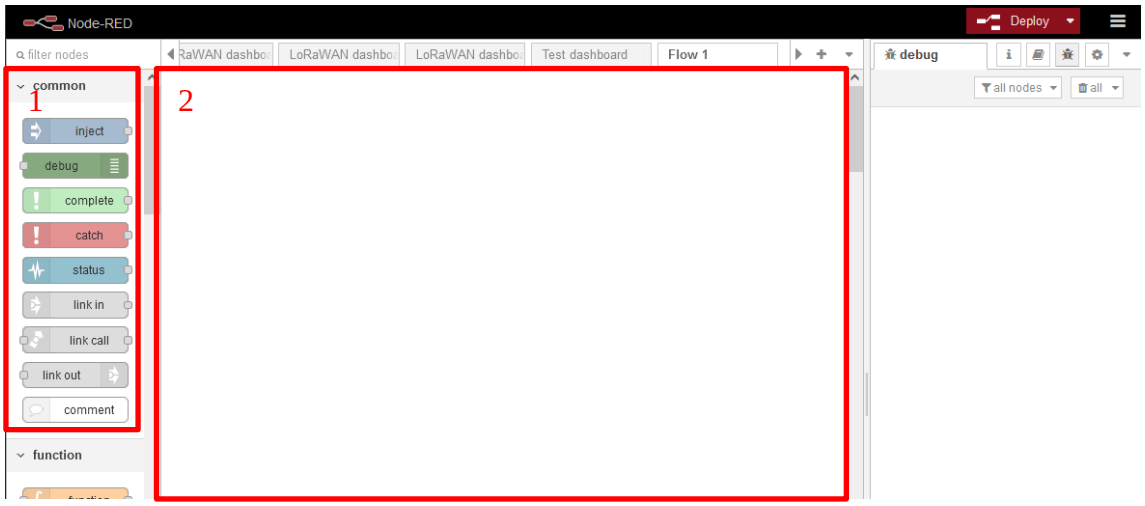


Figure 6.1: Node-RED: interface: 1. palette; 2. workspace.

Most of the nodes are designed to perform a particular procedure, however, there is a “function”-node that can be used for custom programs in JavaScript.

In addition to existing built-in nodes, there are multiple additional tool sets provided by developers of Node-RED and community. One of such set is Node-RED dashboard, that allows to perform simple visualisation of incoming data.

In the following sub-chapters, it will be demonstrated how to build monitoring applications for data from sensors connected to LoRaWAN and managed by services provided by Altibox. The concept of this approach is to consider alternative way of creating user interface, dashboard, if any of the Altibox services is not available and compare their performance.

However, it is assumed that ThingParkX is working properly and forwards data to the established connections as planned.

6.1 MQTT protocol

The first dashboard is associated with the connection to MQTT broker established with ThingParkX. The flow consists of two MQTT-in nodes that are subscribed to MQTT broker hosted on HiveMQ (Figure 6.2). The obtained values are processed and displayed on chart.

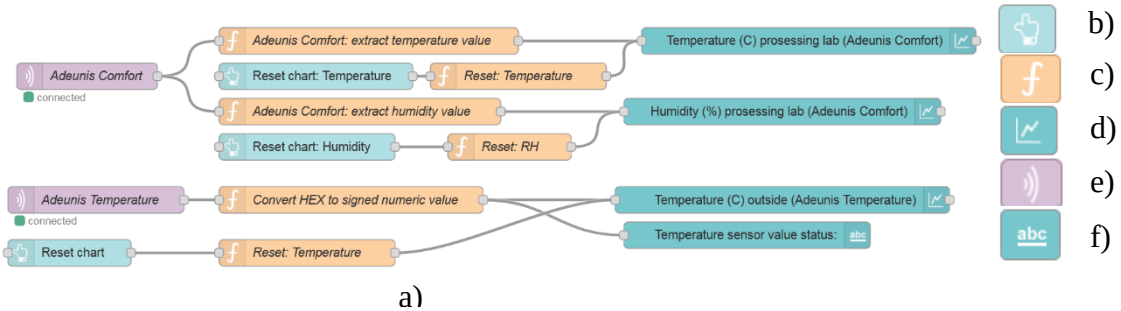


Figure 6.2: Node-RED: MQTT dashboard design: a) flow overview; b) button node; c) function node; d) chart node; e) MQTT-in node; f) text field node.

In the flow two MQTT subscribers are listening to the broker and plot the arrived values in a chart after pre-processing.

Detailed description of content of the nodes and setup is described in Appendix H.

6.2 HTTP to Dimension Four interface

One of the established connections forwards data using MQTT protocol to Dimension Four service. The service stores data into a database, then data can be retrieved using HTTP requests and GraphQL language queries (Figure 6.3).

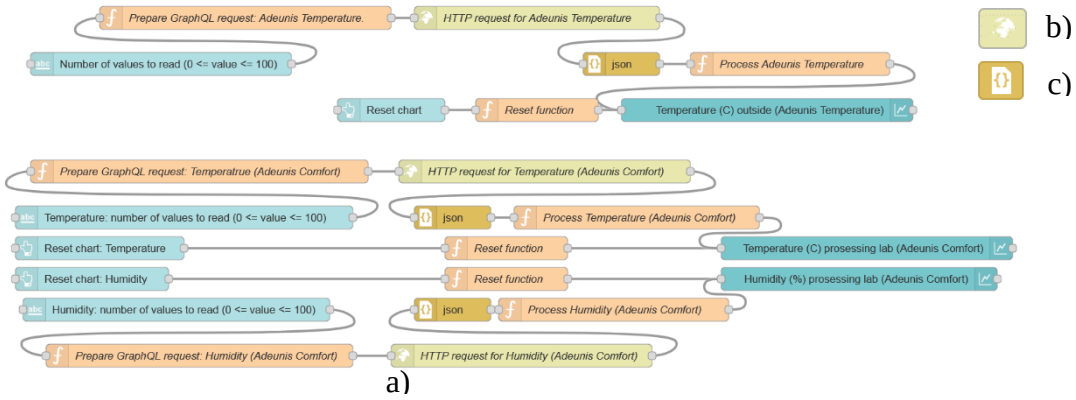


Figure 6.3: Node-RED: Dimension Four dashboard design: a) flow overview; b) HTTP request node; c) JSON node.

User can specify the number of datapoints for each measured parameter, which are to be retrieved. The obtained datapoints are then displayed in the chart.

Detailed description of setup and specification of the nodes is presented in Appendix I.

6.3 IoT hub

Application server niotix provides several APIs. The most relevant is API for interacting with Digital Twins. Functionality of the API allows user to monitor, create, delete and update accounts, users, digital twins, virtual devices and states. From which the most interesting case for the current project is monitoring of states. The overview of the flow is shown in Figure 6.4.

The flow is designed in a way that user initiates reading by activating a button. Then a payload for a specific Digital Twin is built and sent using POST HTTP method. The response is processed and displayed as a time series combining data from the new request together with data from all previous requests. Additionally, the new response is displayed as a gauge.

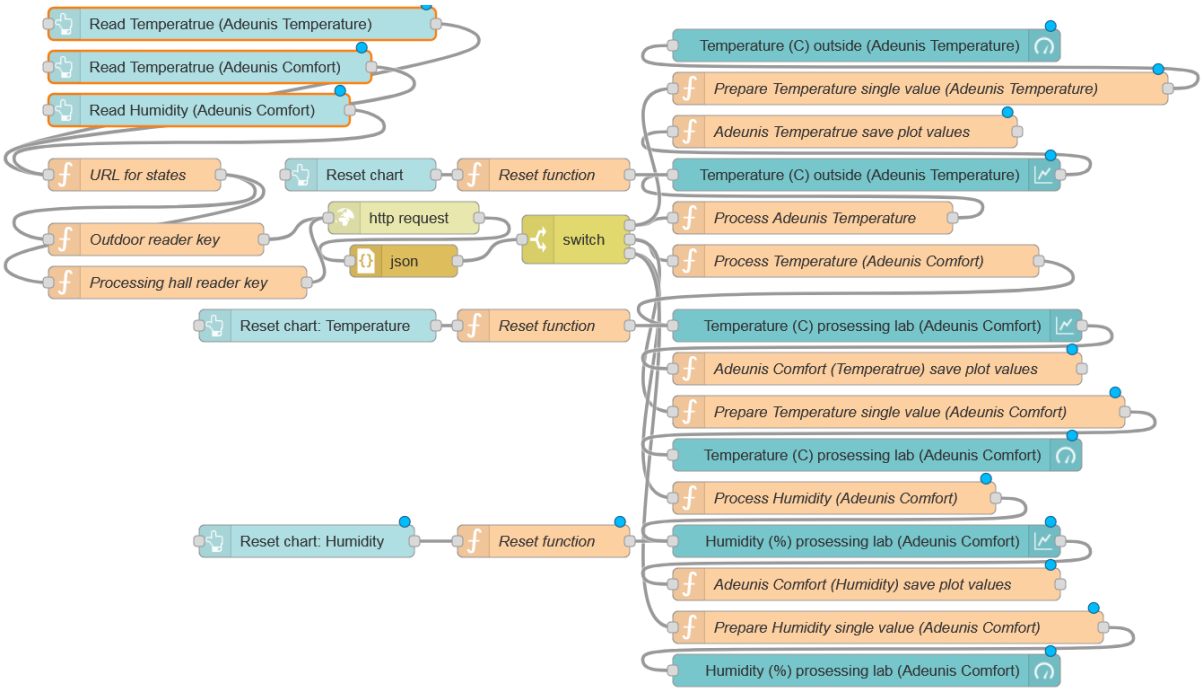


Figure 6.4: Node-RED: IoT Hub dashboard design.

The detailed description of the flow and all its components is presented in Appendix J.

6.4 InfluxDB

Another useful API from niotix is the connection to InfluxDB. This method allows user to request data for time range of interest for a particular state. To implement that functionality, the flow has been designed with adding “Date picker” for each of the measured parameter (Figure 6.5). It has also been added possibility to save data into a file in .csv format. The retrieved values are plotted in dedicated chart.

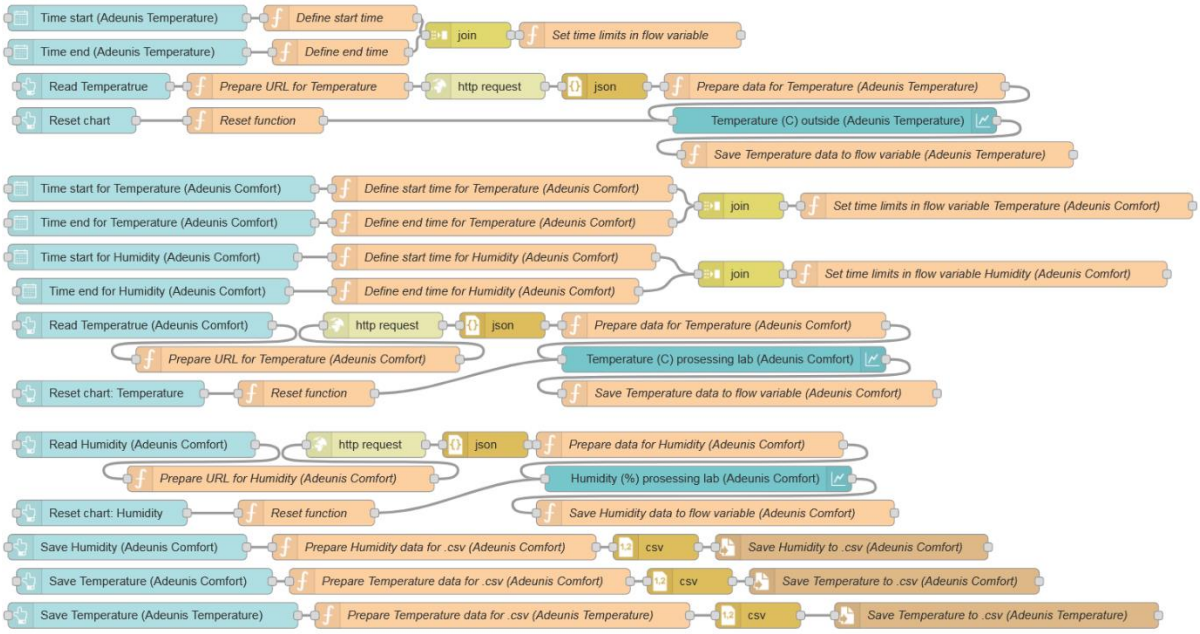


Figure 6.5: Node-RED: InfluxDB flow overview.

Detailed description of setup nodes and functions is presented in Appendix K.

7 Results.

This chapter presents results achieved during implementation of the methods described in the previous chapters.

7.1 MQTT connection

The MQTT connection established with ThingParkX is similar for both HiveMQ broker and Dimension Four broker. Successful delivery of messages has been examined by using MQTTX desktop client subscribed to the HiveMQ broker (Figure 7.1), and by visual observation of the list of signals registered on Dimension Four platform (Figure 7.2).

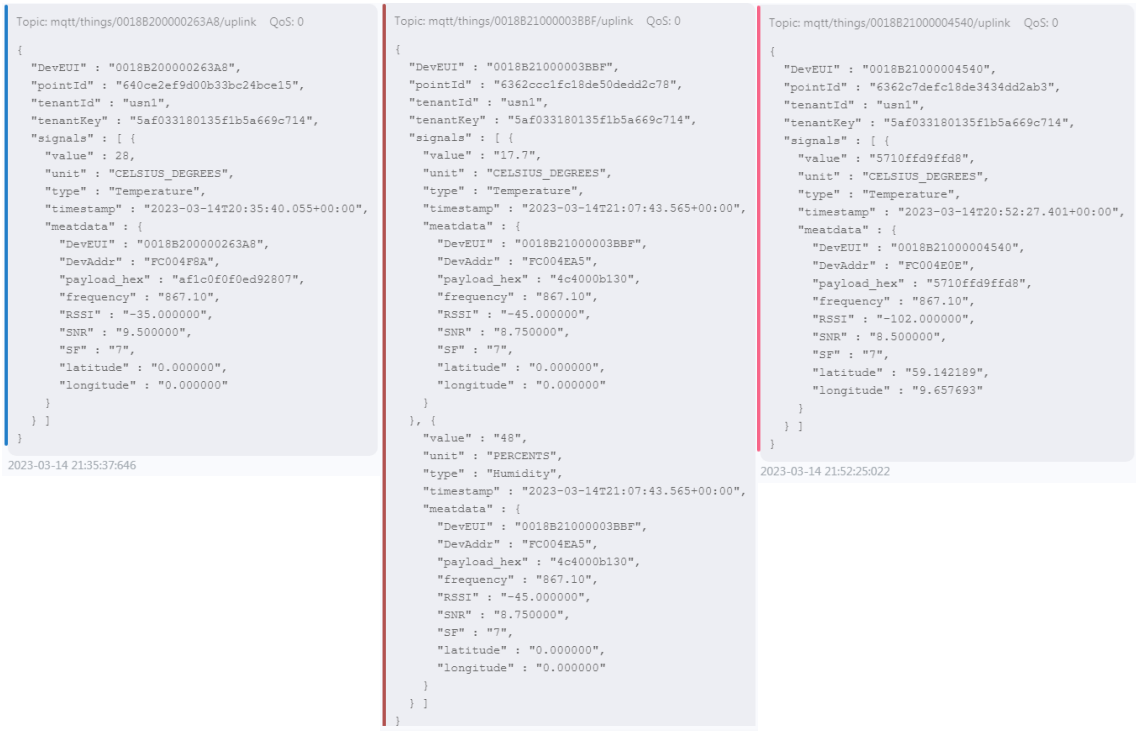


Figure 7.1: MQTT publishing on HiveMQ, observed in desktop client.

There is an error with the temperature sensor related to the driver implemented in ThingParkX. This error causes a failure during the value conversion and consequently erroneous data saved in the data storage. Nevertheless, metadata containing raw data in “payload_hex” has also been saved on Dimension Four, while not presented on the webpage.

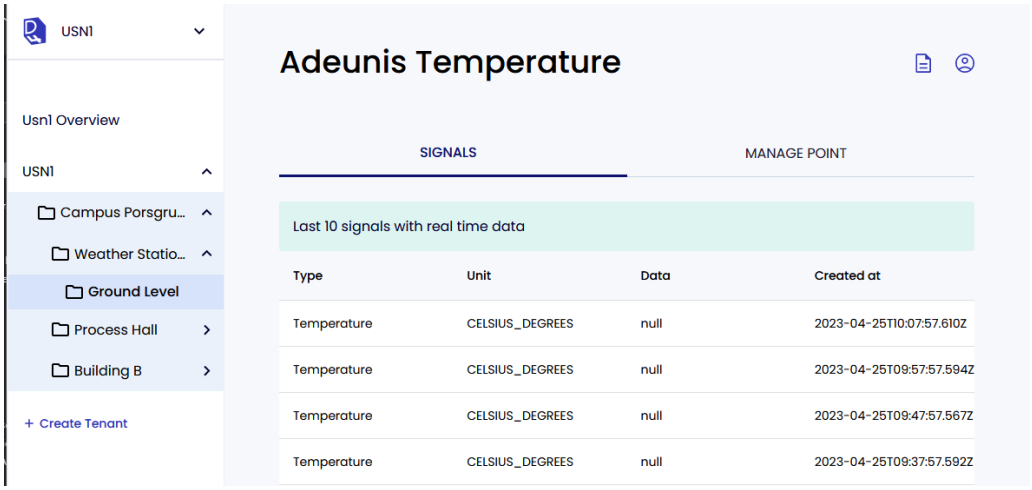


Figure 7.2: MQTT publishing on Dimension Four observed on the portal webpage.

7.2 Azure IoT Hub

Successful transmission to Azure cloud service maybe examined in the “Metrics” tab on the Azure home page (Figure 7.3).

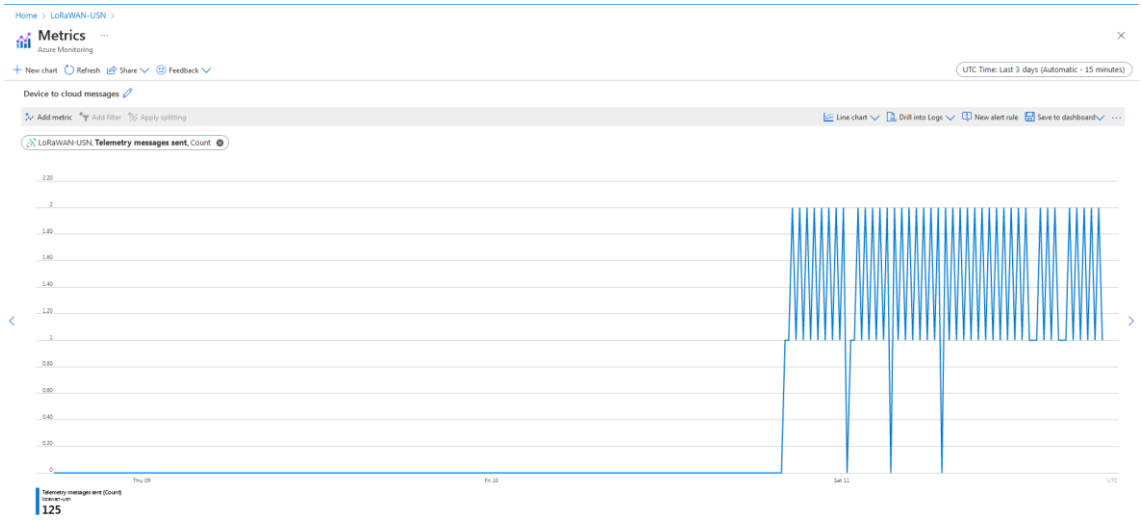


Figure 7.3: Azure IoT Hub metrics.

On a local machine, the messages have been monitored with Azure IoT Explorer (Figure 7.4). Content of the messages corresponds to the ones transmitted with MQTT connection.

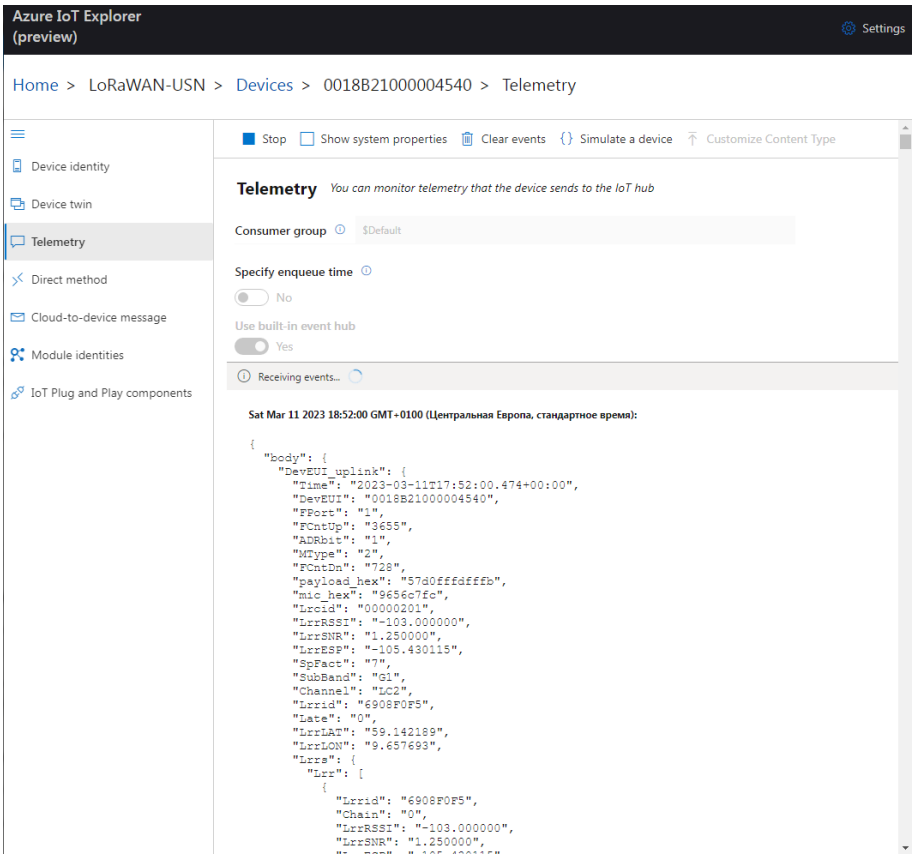


Figure 7.4: Azure IoT Explorer received message.

Error message from Temperature sensor that was shown as null in Dimension Four had also been transmitted to Azure and could be observed with more details (Figure 7.5).


```
        "DynamicClass": "A",
        "payloadDecodedError": {
          "code": "com-4006",
          "message": "driver internal error: Error: invalid uplink payload: unknown frame type",
          "driverId": "actility:adeunis-temp2:1"
        },
        "downlinkUrl": "https://altibox.thingpark.com/iot-flow/downlinkMessages/4aca9198-2cd2-4487-8a1f-9fd50e112054"
      }
    },
    "enqueuedTime": "Sat Mar 11 2023 18:52:00 GMT+0100 (Центральная Европа, стандартное время)"
}
```


Figure 7.5: Azure IoT Explorer error message.

The same could also be observed on ThingPark Wireless-Logger portal (Figure 7.6).

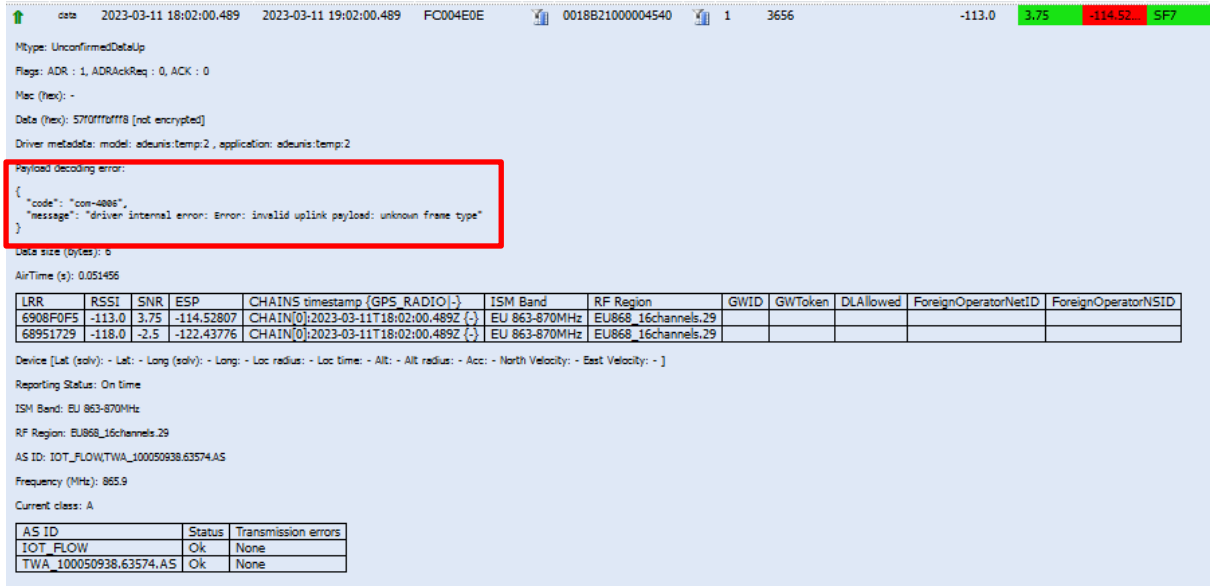


Figure 7.6: ThingPark Wireless-Logger error message.

7.3 Altibox application server

Structure of Digital Twin comprises several nested twins identifying locations in the area of the campus. It has been separated into areas that are supposed to be used outside of the buildings and inside the buildings. Therefore, the campus has been divided into Outdoor and USN Building Digital Twins (Figure 7.7). Each of the nested Digital Twins has been marked on the map for easier identification.

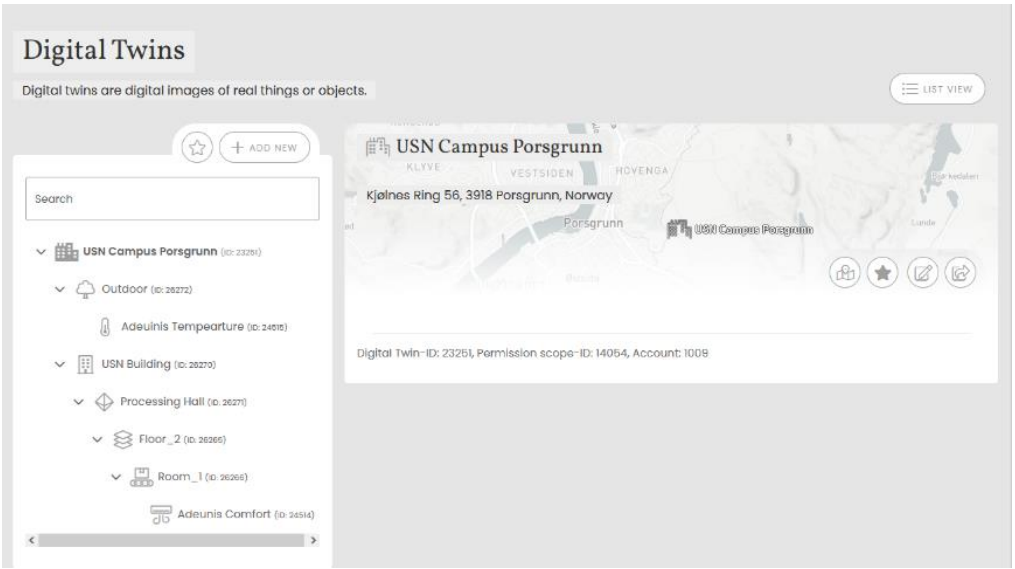


Figure 7.7: niotix: Digital Twin structure.

Using the niotix tools for specifying polygon, USN building has been marked on the map. This polygon belongs to the root Digital Twin “USN Campus Porgrunn” and is supposed to include all buildings and areas on the campus (Figure 7.8).

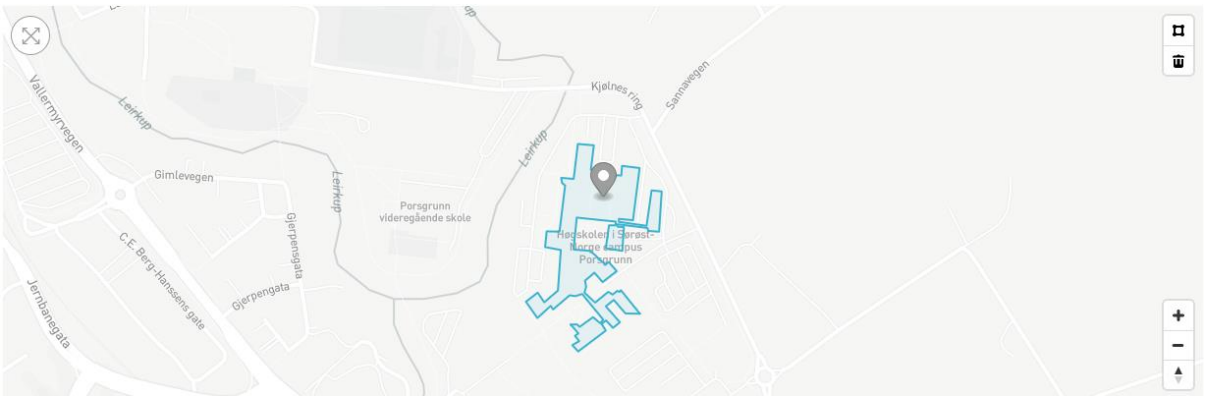


Figure 7.8: niotix: USN Campus Porsgrunn polygon.

Building where one of the sensors is assumed to be placed is Processing hall. Its location is marked as a polygon on the same map around corresponding the university building. It can be easily distinguished from other buildings when user choose this Digital Twin (Figure 7.9).

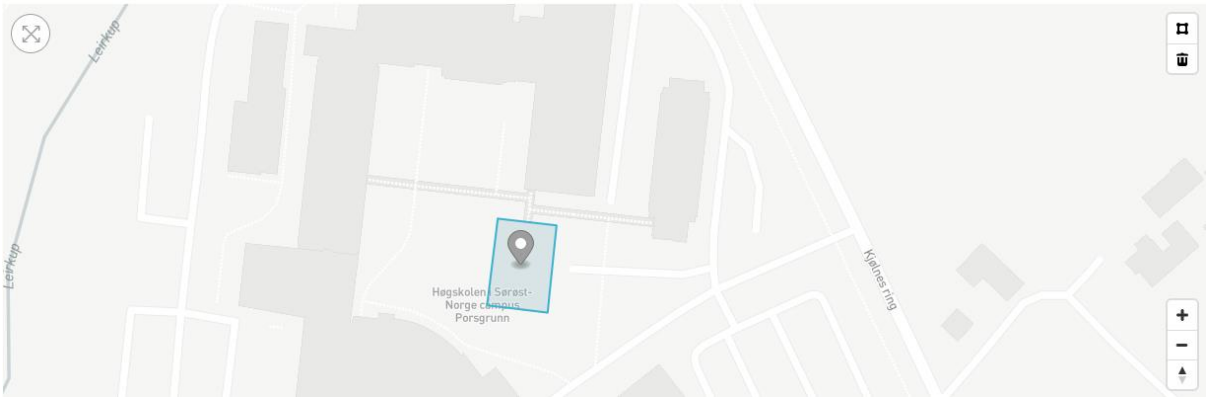


Figure 7.9: niotix: Processing hall polygon.

Digital Twin Floor_2 is of type “Floor/Level”, it includes an image of the floorplan and a polygon specifying the actual size of the floor (Figure 7.10). Floor type is chosen for keeping logical structure of the construction. The image is not meant to represent the actual floorplan.

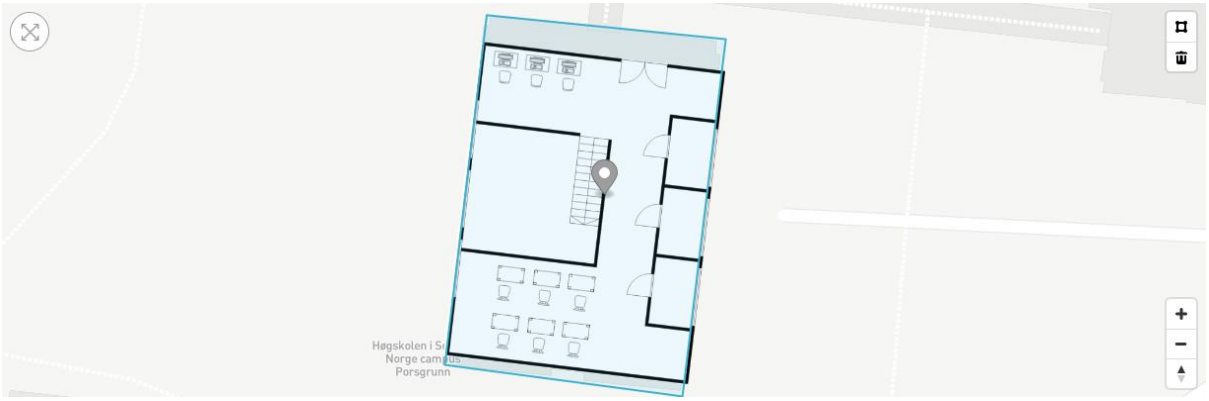


Figure 7.10: niotix: Floor 2 polygon with floorplan.

Room_1 is of type “Zone” (Figure 7.11) and is marked as a part of the floor, which corresponds to the specific room.

Figure 7.11: niotix: Room polygon.

Under the Floor_2 Digital Twin the sensor Adeunis Comfort is defined (Figure 7.12). The sensor has its area of activity, that may be used to alarming when devices transmitting their location are in the range of measurements. The marked area does not represent the actual measurement range, but similar approach can be used in real implementation.

Figure 7.12: niotix: Sensor polygon.

Overall design may also be shown as a 3D map highlighting all defined elements (Figure 7.13). Outdoor and USN Building Digital Twins are in the same location. However, Outdoor area has not been projected to the real area and only shows a hexagon polygon. The sensor, Adeunis Temperature, nested under the Outdoor, is located outside of the Processing Hall building.

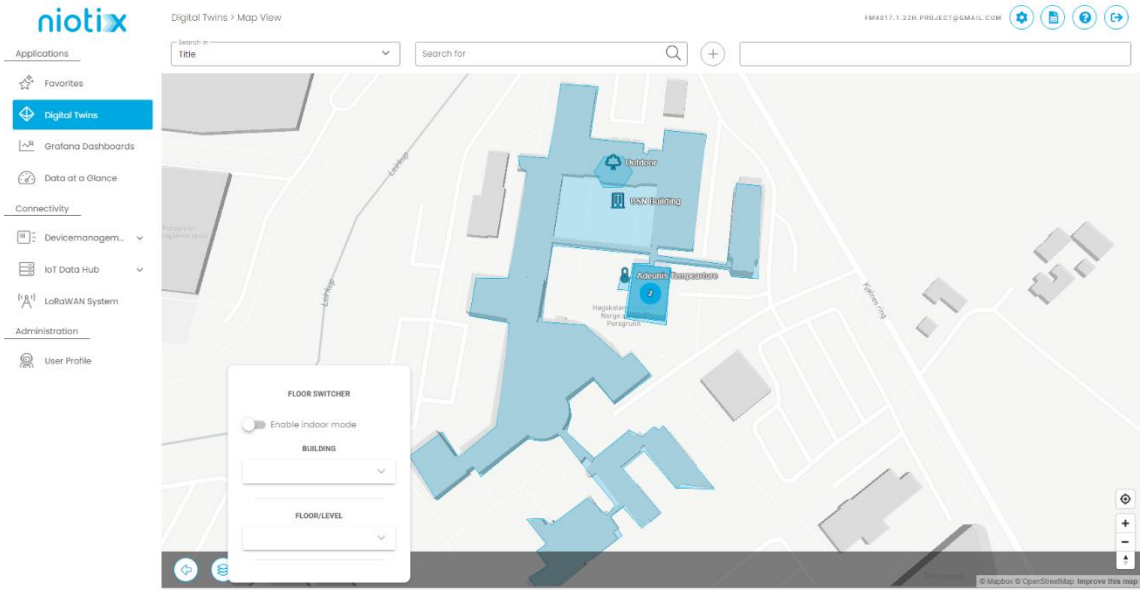


Figure 7.13: niotix: USN campus map 3D, outdoor mode.

To display the indoor elements on the general map, the user must activate the “Enable indoor mode” (Figure 7.14). The map changes immediately.

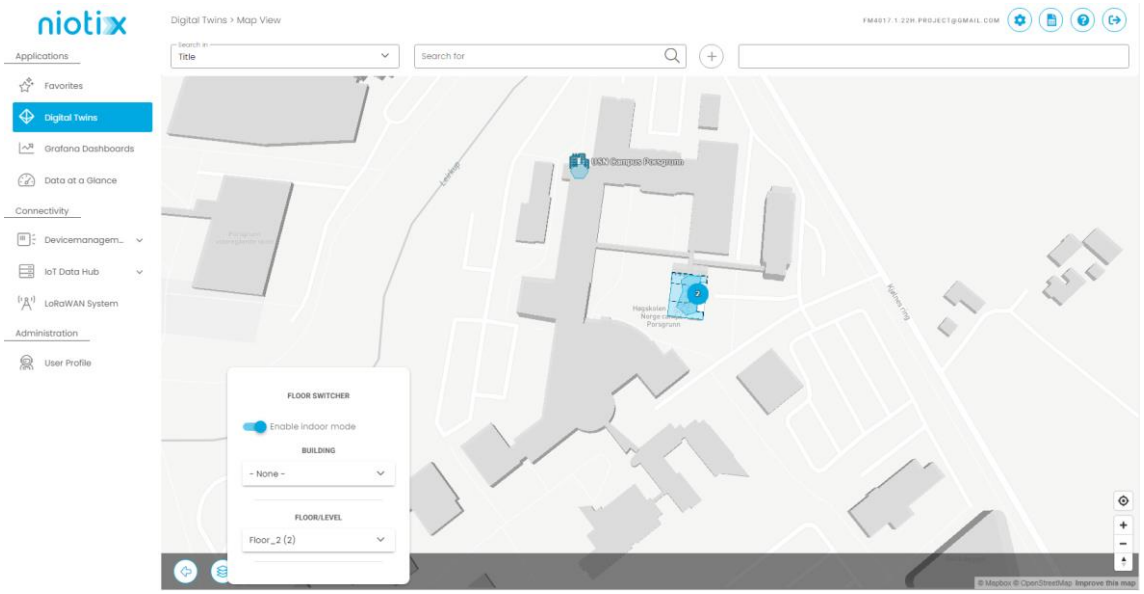


Figure 7.14: niotix: USN campus map 3D, indoor mode.

There has been registered only one application, USN_Porsgrunn_Sensor_Network, for the root Digital Twin, that includes all available devices (Figure 7.15).

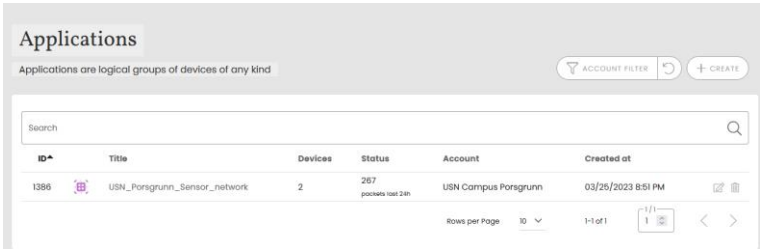


Figure 7.15: niotix: application.

With that application, it has been registered two devices representing Adeunis Comfort and Adeunis Temperature sensors (Figure 7.16). Both devices are identified by their DevEUI.

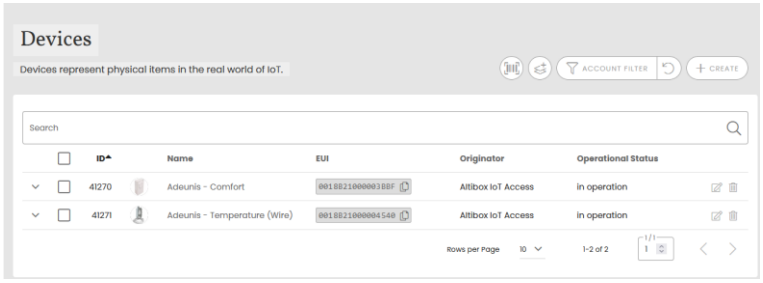


Figure 7.16: niotix: devices.

Measurements from Adeunis Comfort have been successfully transferred to the niotix and are demonstrated in user interface (Figure 7.17). The measurements are saved for two separate Data states: Temperature and Humidity.

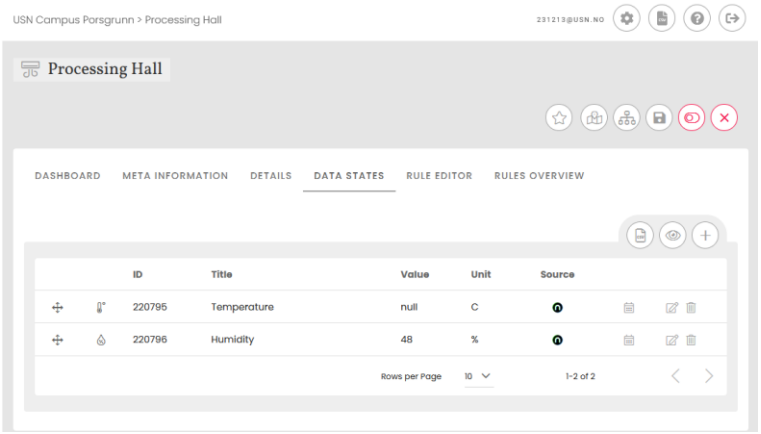


Figure 7.17: niotix: Adeunis Comfort data states.

These states are also visualised on two separate charts (Figure 7.18).

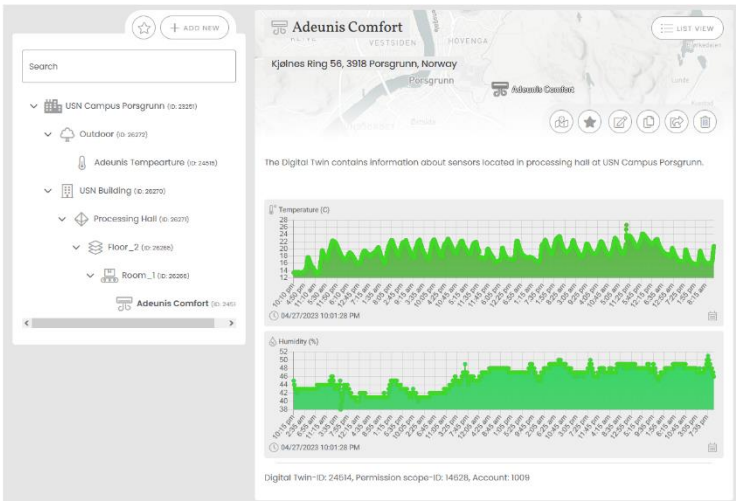


Figure 7.18: niotix: Adeunis Comfort plot.

Adeunis Temperature sensor is registered as a Data state under Outdoor Digital Twin. The data is also available on the overview screen as a plot (Figure 7.19).

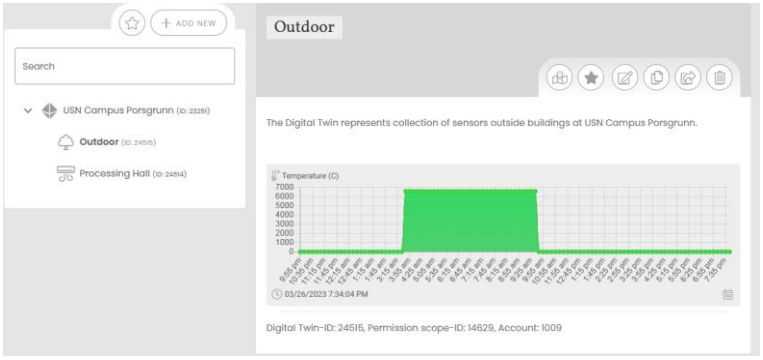


Figure 7.19: niotix: Adeunis Temperature plot.

It is possible to save data as a csv file from a particular data state for desired period of time (Figure 7.20).

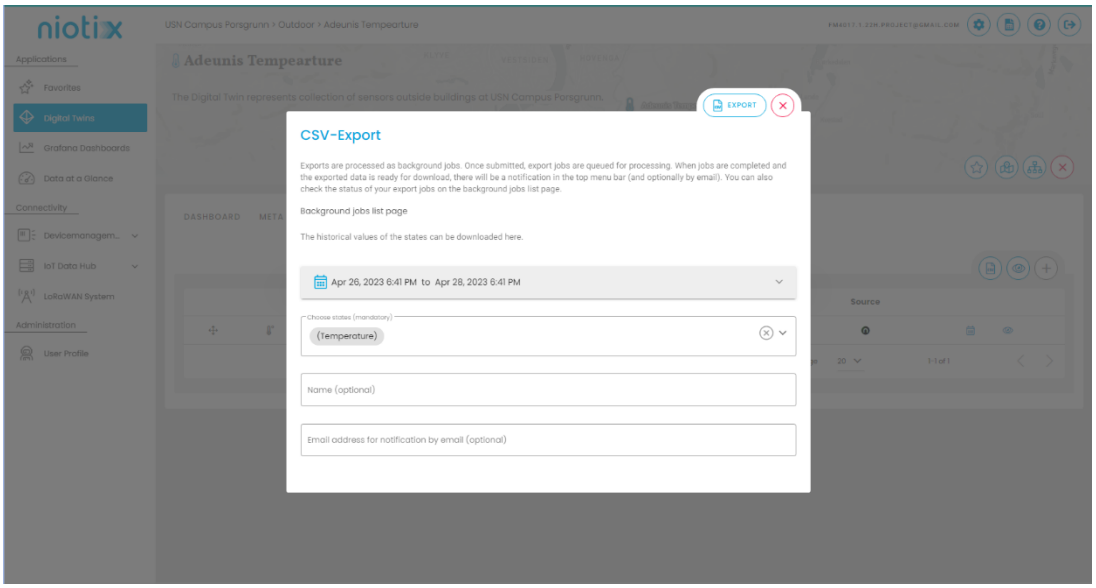


Figure 7.20: niotix: Adeunis Temperature data export.

7.4 Grafana dashboard

Overall dashboard is available for the end-user similar to niotix dashboard. Grafana service retrieves data from niotix and visualises it as defined. In this case, two sensors produce in total three Digital States. All these states are presented as separate charts and tables (Figure 7.21).

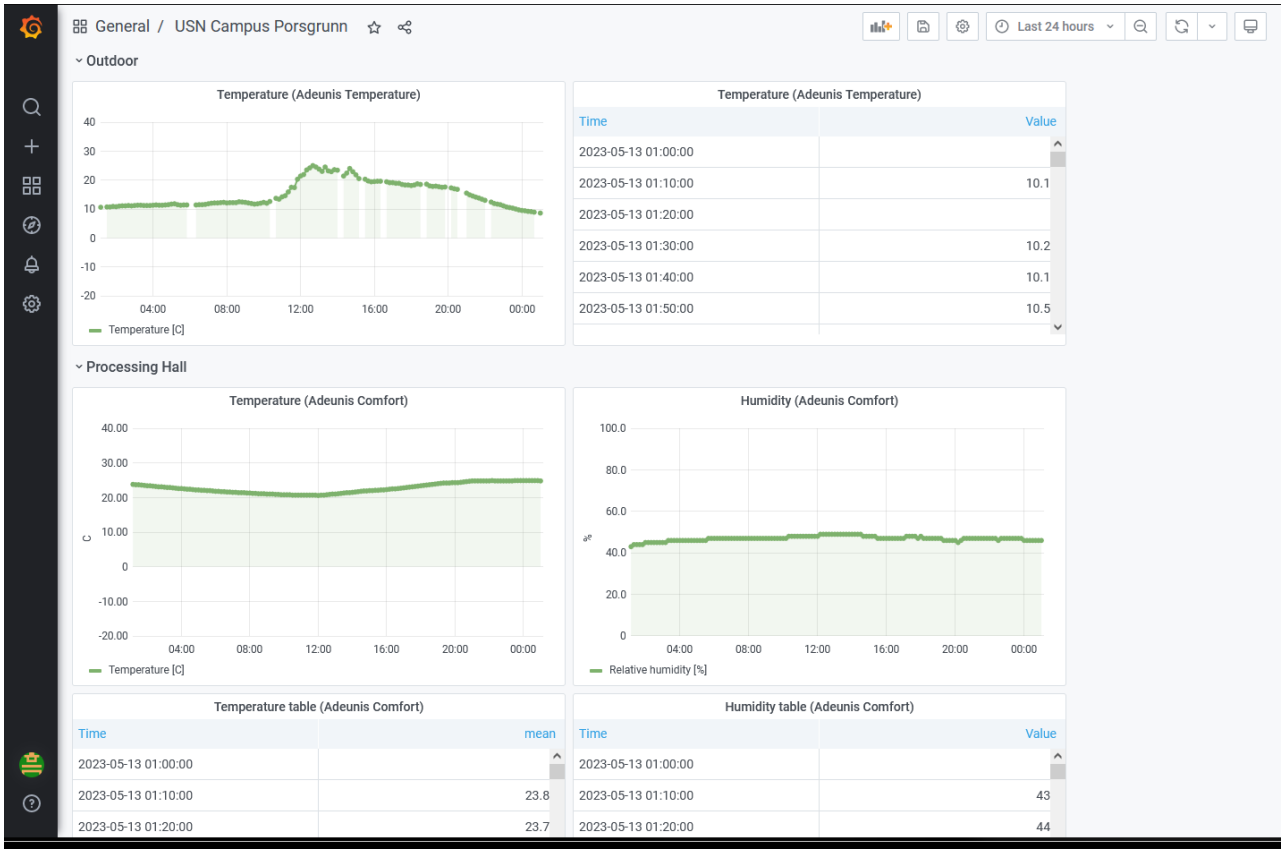
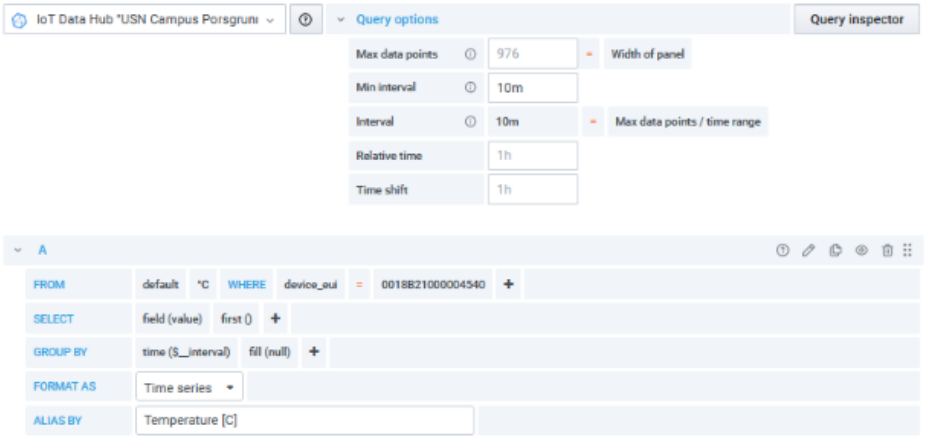
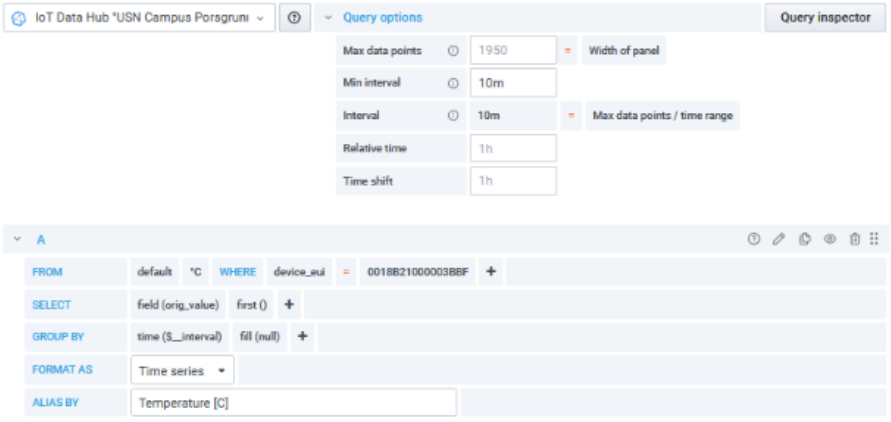


Figure 7.21: Grafana: user interface.

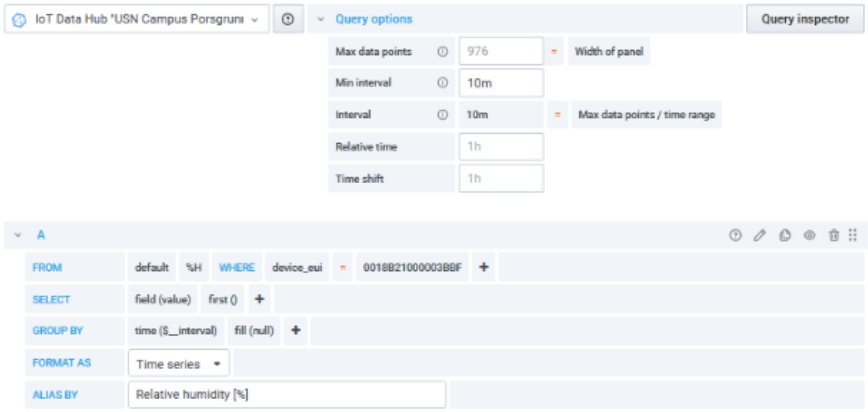
Queries for each of the devices identify the sources by DevEUI and the type of measurement (Figure 7.22). Visualisations are pushed to display points with 10 minutes time interval between adjacent points, what corresponds the actual sampling rate. The number of points displayed, however, depends on the requested time range for the whole dataset, 24 hours in this case. Therefore, larger or shorter time ranges will result in displaying of data points as separate lines, if the range is too short, or averaging of values, if the range is too large.



a)



b)



c)

Figure 7.22: Grafana: queries: a) Adeunis Temperature, c) Adeunis Comfort temperature, a) Adeunis Comfort humidity.

Data can be exported from Grafana for each measured parameter. By following steps as shown in Figure 7.23, user enables the menu that gives options to save values for the chosen time period.

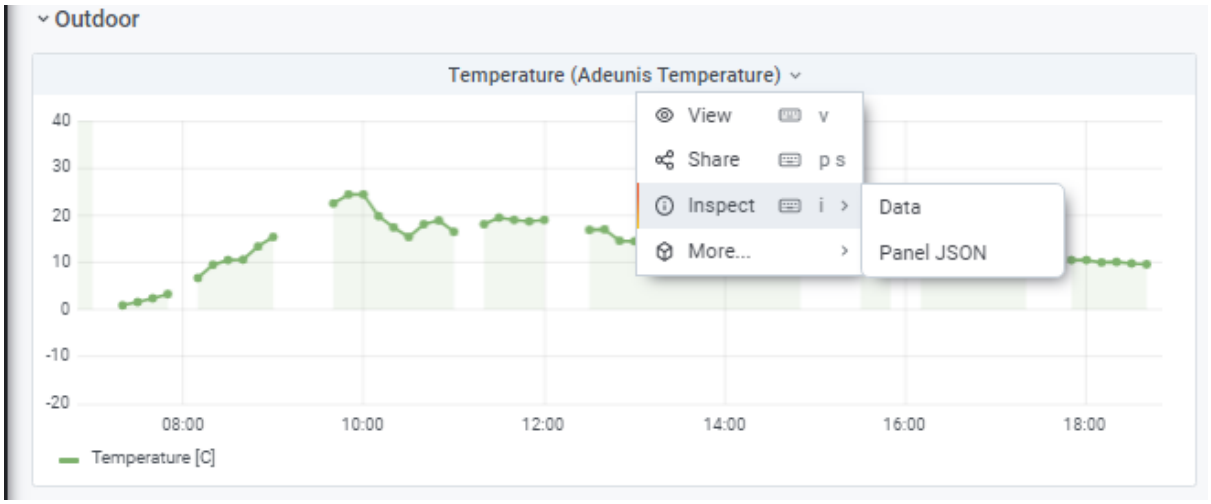


Figure 7.23: Grafana: data export.

7.5 Node-RED monitoring application

In this sub-chapter, user interfaces developed with Node-RED programming tool are demonstrated. Each of the interfaces represents a separate dashboard with buttons, date picker, charts and text fields, which user may interact with.

7.5.1 MQTT interface

This version of dashboard (Figure 7.24) has a MQTT subscriber running continuously in the background. The new value transmitted by device is added to the chart. In case of Adeunis Temperature sensor, if the value is null, caused by mentioned previously error with driver, the measurement is converted from “payload_hex” field in the metadata. User is then informed that special treatment was required. This process is, however, implemented only for one device. “Reset chart” button allows to clear the chart area.

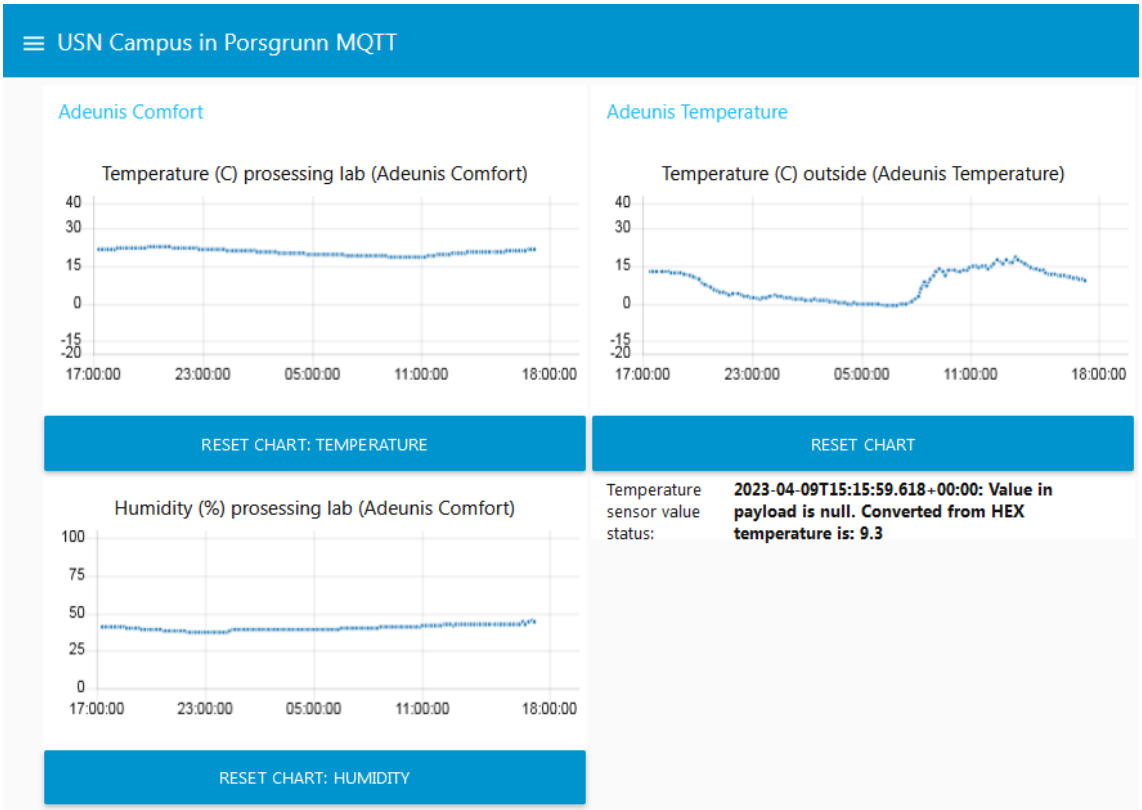


Figure 7.24: Node-RED: MQTT dashboard.

7.5.2 Dimension Four interface

The dashboard gives a user an opportunity to retrieve a certain number of stored data points from the platform. The number is limited from 0 to 100, if the number exceeds 100, it is replaced with 100 (Figure 7.25). Each of the parameters is read separately.

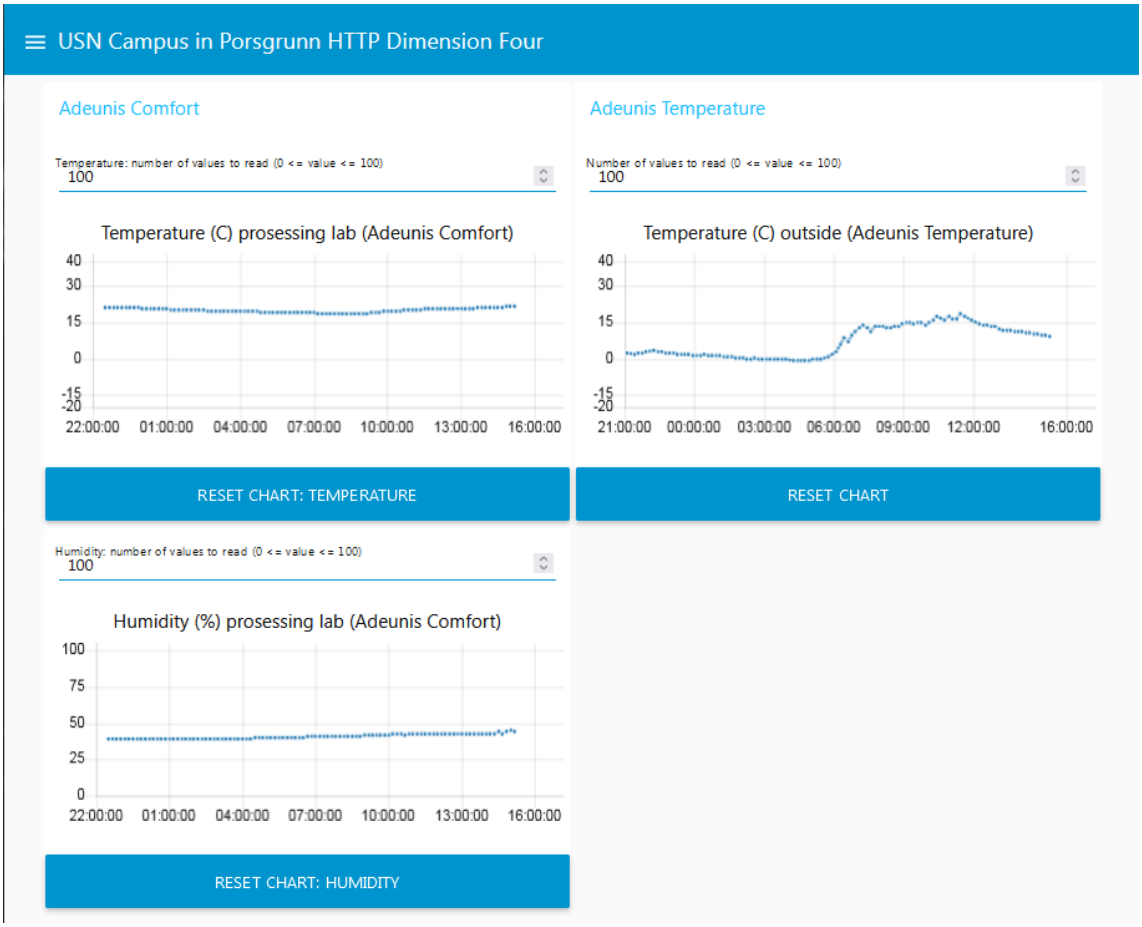


Figure 7.25: Node-RED: Dimension Four dashboard.

7.5.3 Niotix interface

The interface allows a user to request the last value stored for a Data state in the niotix IoT hub (Figure 7.26). The value is then visualised with a gauge widget and is plotted in the chart. Reading of data is initialised by the user for each individual parameter.

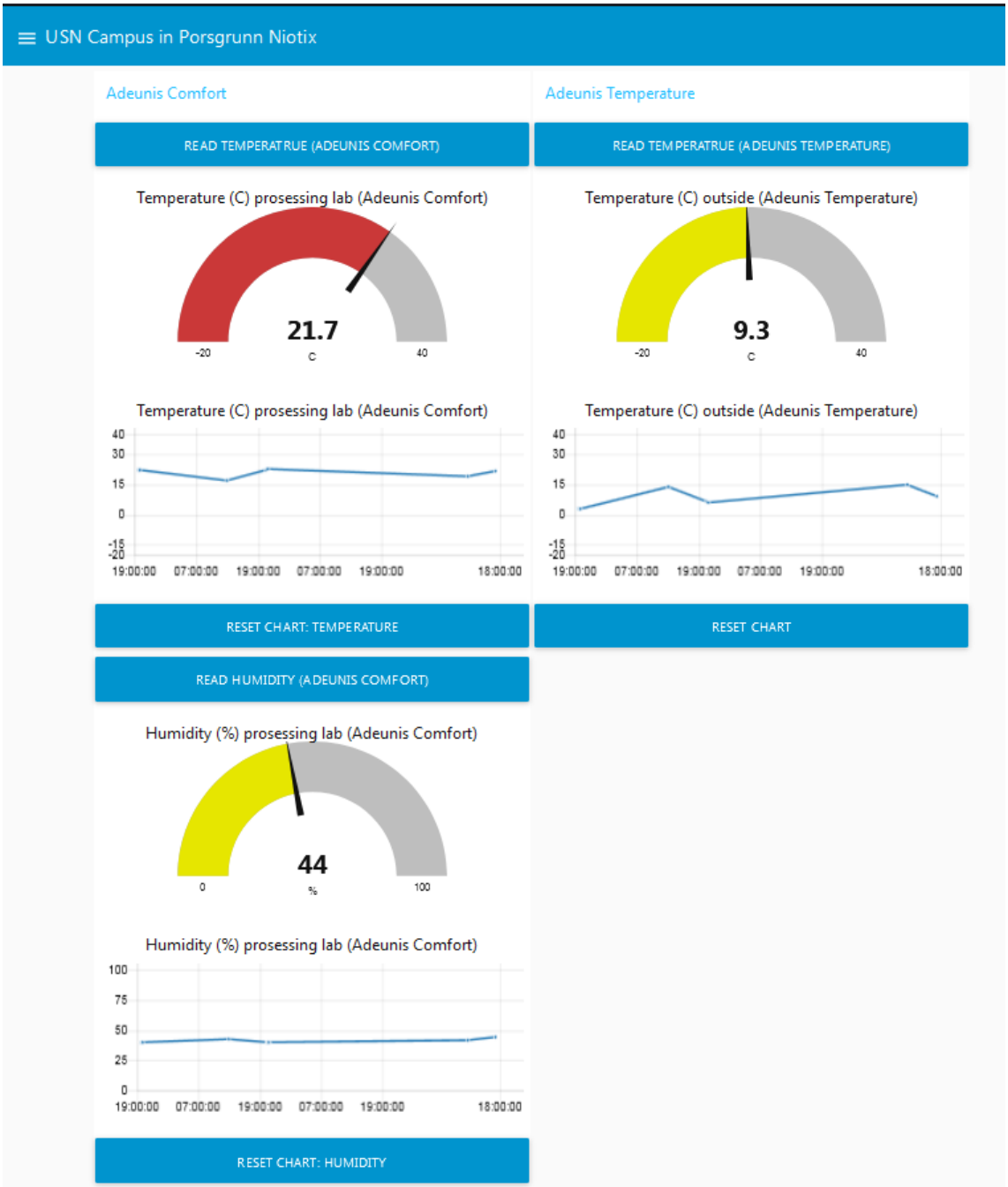


Figure 7.26: Node-RED: niotix dashboard.

7.5.4 InfluxDB interface

InfluxDB dashboard provides a convenient way to request available measurements for a desired period, display them and store as csv file (Figure 7.27)

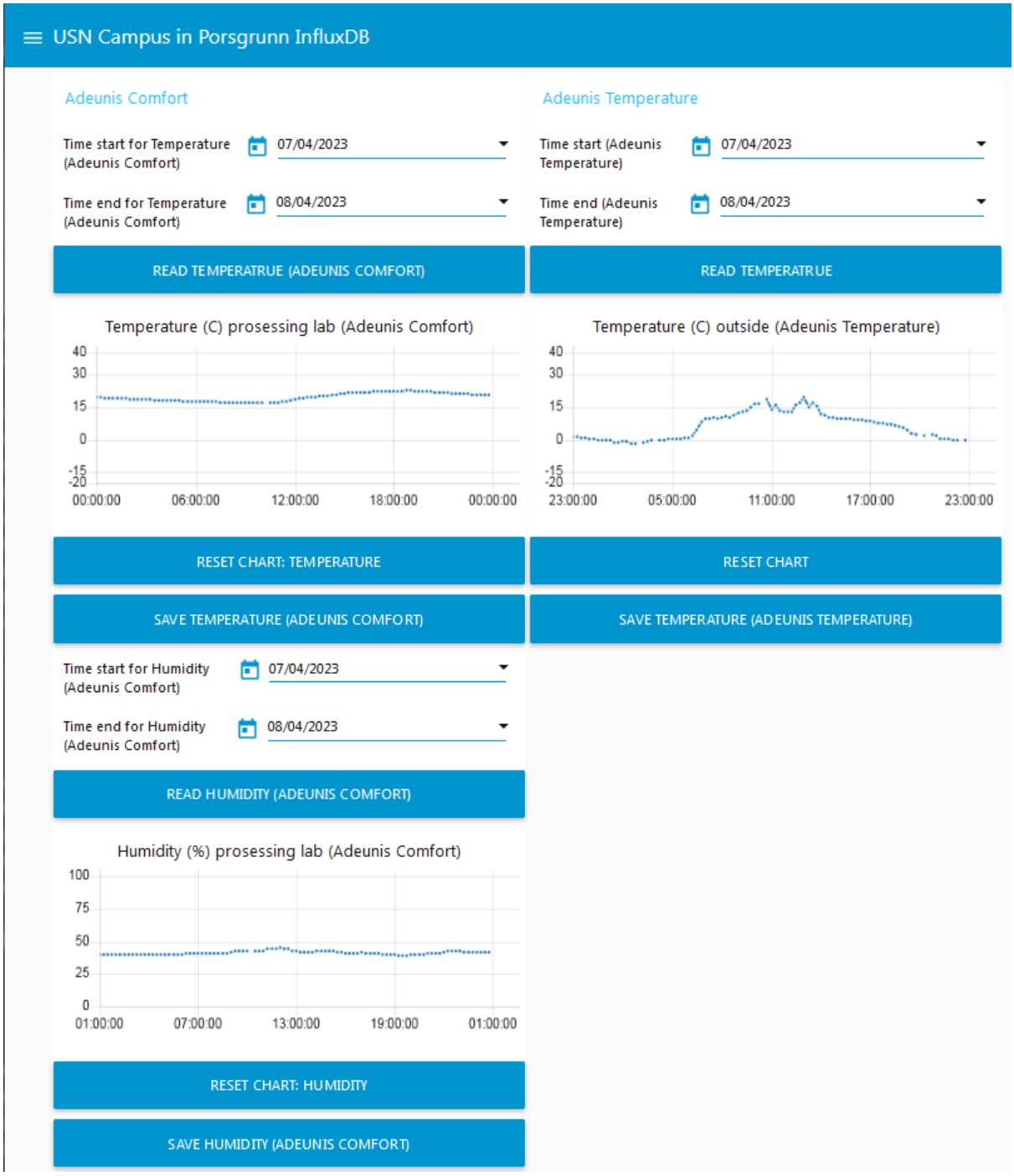


Figure 7.27: Node-RED: InfluxDB dashboard.

The values are first stored in the flow-variable and then using “Save to file” node is converted to text and stored to the specified path (Figure 7.28).

a) b)

c)

Figure 7.28: Node-RED: InfluxDB dashboard data: a) output from the dashboard; b) content of the saved file; c) list of saved files.

8 Discussion

Development of the monitoring system has resulted in several paths for storing, retrieving and presenting measurements. The general structure of the resulted system is shown in Figure 8.1.

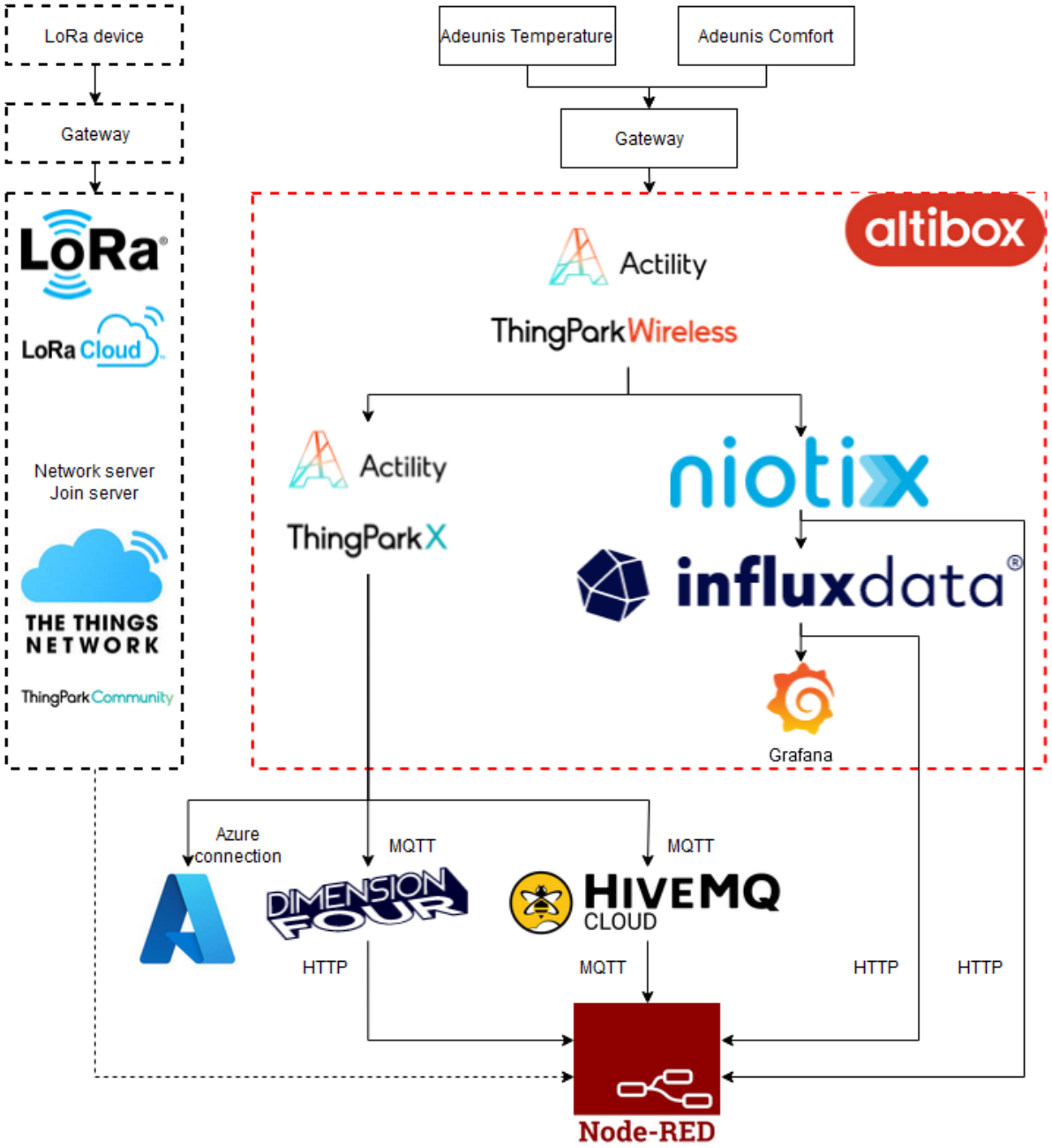


Figure 8.1: System overview.

All the currently used data flow control is kept under Altibox umbrella and is essentially managed by the incorporated services (red dashed box in the Figure 8.1). The data outgoing from the Altibox domain can be, however, manipulated and presented to a user by other services locally or web based. In this chapter, the findings and performance of the designed solutions are discussed.

Combination of LoRaWAN servers, such as Network Server, Join Server, Application Server, with ThingPark platform, developed by Actility, has shown to be a very convenient way of managing and observing the network at lower level. Users with proper subscription may add, remove and edit devices. With that service one can identify potential errors during transmissions and evaluate quality of signal based on information provided by gateways. The platform also allows addressing messages to a custom server.

At the same time, connecting of new devices must be approved by a system administrator. Extension of the network with additional gateways cannot happen without permission, as the administrator registers new gateways. Therefore, it is important to note that any manipulations with devices and the network must be coordinated with the administrator in the first place. While this process provides better governing of deployed network, such hierarchy takes a part of flexibility from user and may be time consuming.

Another Actility platform, ThingParkX, provides a significant diversity to types of services that users may integrate in their application. Possibility to connect LoRaWAN devices to different protocols and end points allows to combine several non-related LoRaWAN platforms to present data in one monitoring portal.

One of the established connections has been also forwarding packets to Microsoft Azure IoT hub service. While the service provides sophisticated security level, it has quite a restricted functionality in terms of access to it and monitoring measurements. IoT Hub is not free of charge, and user must have a license from Microsoft Corporation.

On this server, there are several versions of quotas that user can use with registered devices. The free one allows to receive up to 8000 messages from all devices. If one Adeunis Temperatrue sensor transmits 1 UL every 10 minutes, then total number of UL should not exceed 144 a day. This does not take in account additional UL with only MAC or Join-requests, those are not transmitted often. That means that IoT Hub should allow to have about 55 devices connected to it. Nevertheless, UL counting is not straightforward in the service, so it is possible to have over 400 UL from a device with 10 minutes sampling time. That does not correspond to reality.

Moreover, it is not possible to monitor content of transmitted UL on the Azure IoT Hub web portal. There are nodes in Node-RED for connecting to, that are designed by Azure team. Nonetheless, during this work the nodes could not be installed because of new version of Node-RED that Azure nodes are not compatible with. Alternatively, retrieving of UL is possible with application developed in C#, but that was not an objective of the work. Therefore, there has not been created a dashboard for this connection.

niotix, made available by Altibox, has shown to be the most versatile of the solutions presented here. The server allows to register available devices, receive UL and send DL, examine the payload, plot measurements and observe quality of signal and transmissions. In the end, the values may be logged in a physical file for a desired period.

The service has a functional API allowing connection to two related data storages IoT Hub and InfluxDB. At the same time, interconnection between the storages is not very clear, API responses from the hub and the database are different. Therefore, they are not interchangeable, if any of them goes down.

There is also very useful feature, from user perspective, that connects areas of interest and devices to the real map. Additional option that the user can include own floorplan drawing of a part of a building or outside area makes monitoring more related to the real-life objects and conditions.

Usability of niotix takes an additional advantage from implementing Digital Twin of type "Trackable". These devices change their position and can trigger actions if enter a polygon defined to be a zone of activity of a stationary device. Such approach makes monitoring system more dynamic than with only fixed sensors. However, that implies additional power consumption for the trackable devices, as together with moving they will also have to request and transmit own position and measurements very often to be properly referred to the map. In the case when a trackable device updates its position too seldom, it may never be registered as entered an area if the area is too narrow and the device moves too fast. But the trackable devices have not been implemented in the current work, and this functionality requires additional investigation.

While niotix implements the most important functions of a monitoring and logging application, there are parts of the process that administrator must be aware of.

During the work, data from the provided temperature sensor had never been properly decoded as was indicated by the error message shown in ThingPark. Nevertheless, niotix could show the correct value during state setup. It is more likely that the server relies not only on the decoded value but can also decode it on its own from the hex value in the payload.

The messages sent by a device can be analysed on the page of the device. By clicking on payload, it is possible to see the available underlying information for a specific message (Figure 8.2). In this figure, "RAW LORAWAN PAYLOAD" contains "573000100012", where two bytes "0010", equal to 16, correspond to the first channel of the device, and "0012", equal to 18, to the second one. These values are then properly converted as specified for the device to 1.6 ℃ and 1.8 ℃, respectively.

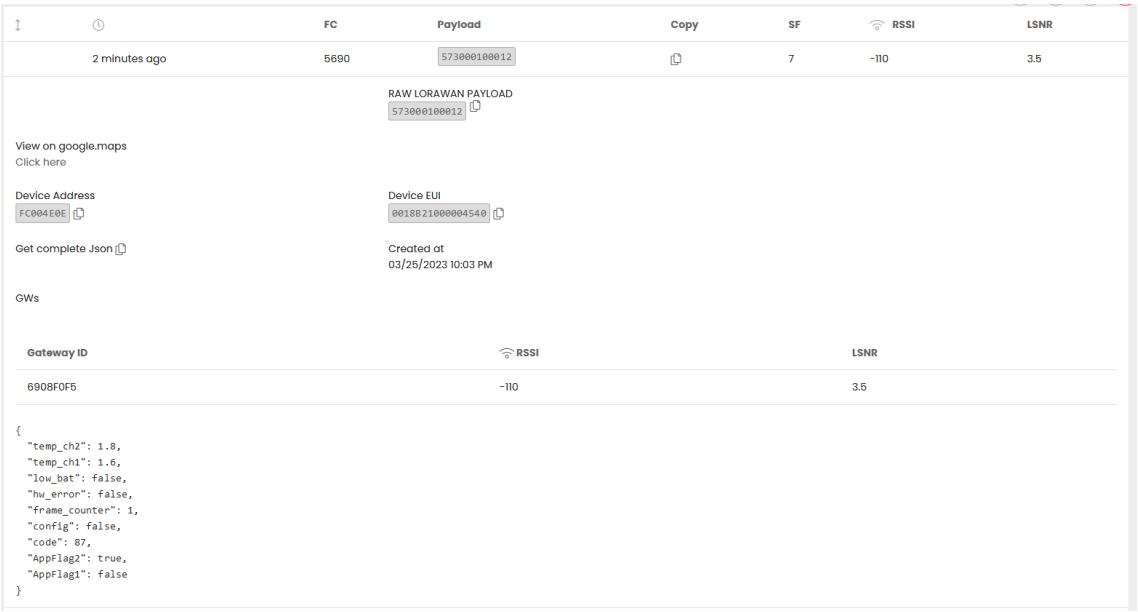


Figure 8.2: niotix: details for a packet with positive temperature.

However, when negative measurement is transmitted, the payload will contain data encoded in a different way, e.g. “fff4 and “fff6” (Figure 8.3). One can see that temperature is then 6552.4, for the first channel, which was directly derived from the HEX value of “fff4”. Therefore, additional manipulation during transformation is needed. This shows a rather significant flaw of the provided server.

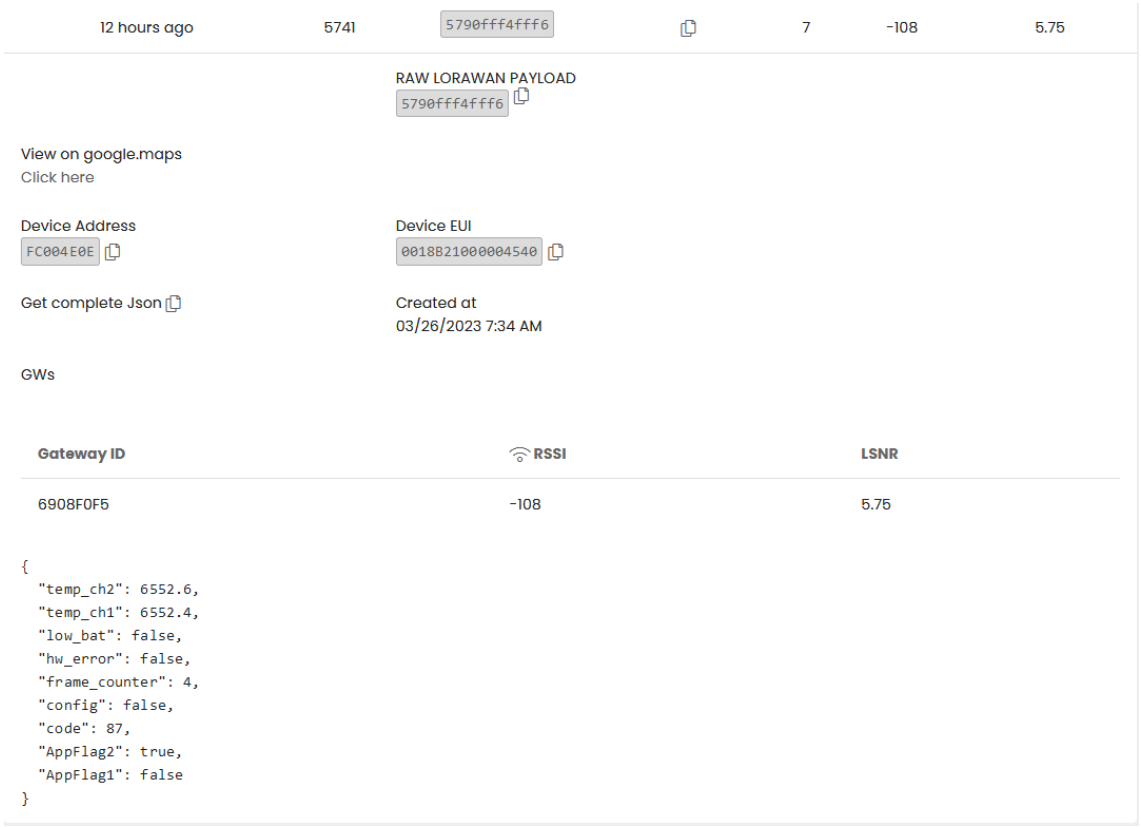


Figure 8.3: niotix: details for a packet with negative temperature.

To cope with that issue several approaches can be implemented. On the server side, the type of value could be specified as signed or not signed number, not just number. Other way is to use existing “Javascript Transformation” function in niotix (Figure 8.3).

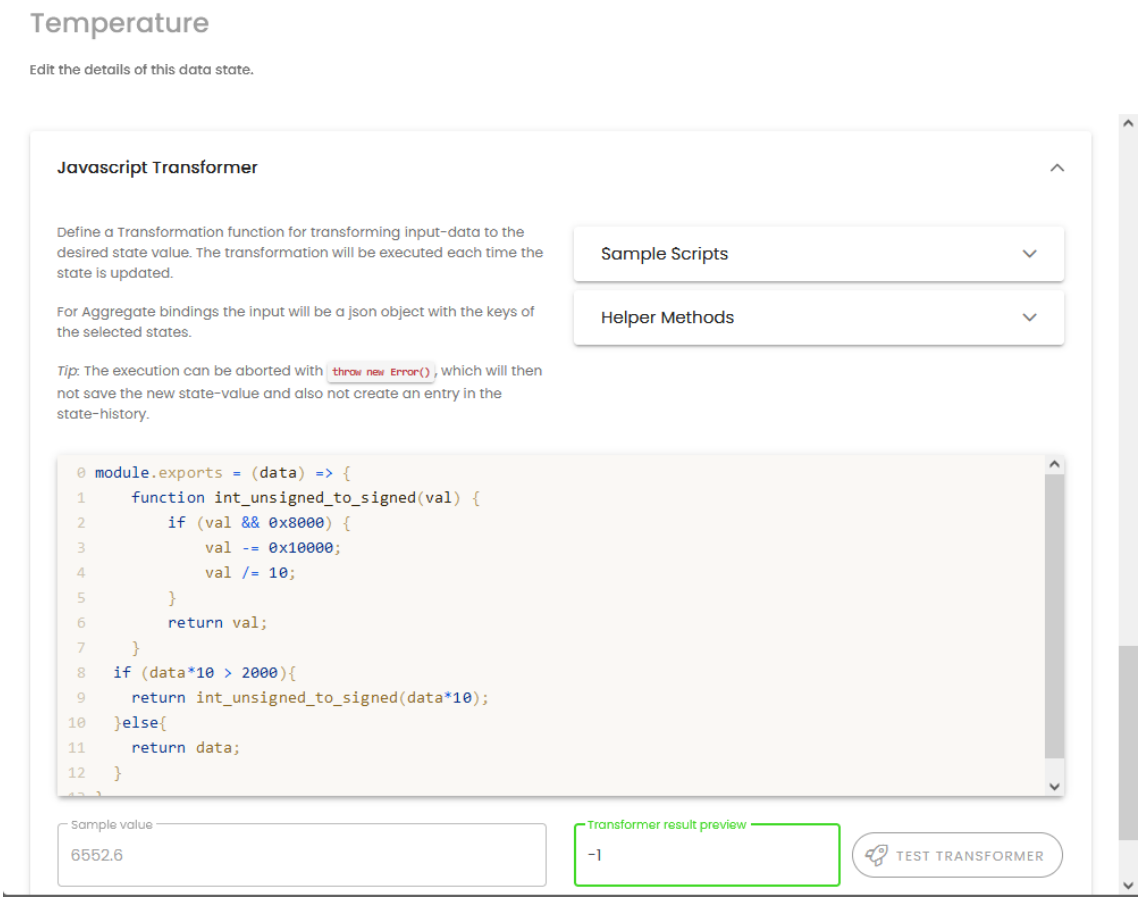


Figure 8.4: niotix: value conversion in Javascript Transformer.

One may compare values plotted before the transformation had been implemented (Figure 7.19) and after (Figure 8.5).

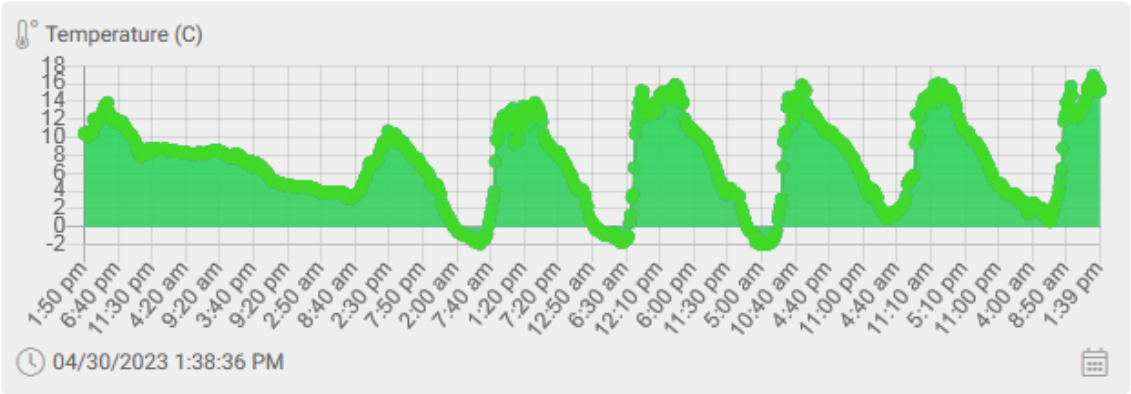


Figure 8.5: niotix: negative temperature is properly displayed.

Considering the stated above, it is a user, or administrator, who first should check what type of data is going to be transmitted and if it can be properly interpreted by the server.

Another precaution should be taken with respect to testing values and storing the values during the test. Under development of the function for transformation of data, several values with intentionally wrong magnitude have been supplied through the transformer interface that resulted in storing of the values as correct ones (Figure 8.6). Because there was no direct access to the data, they were displayed on the main dashboard and could not be removed from the storage. Eventually, after a week it was not possible to find the values again. Therefore, the process of manipulation with data remains unclear. Besides, plotted values are aggregated, minimum, maximum and average, and may not represent conditions in the area of measurements as precise as desired.

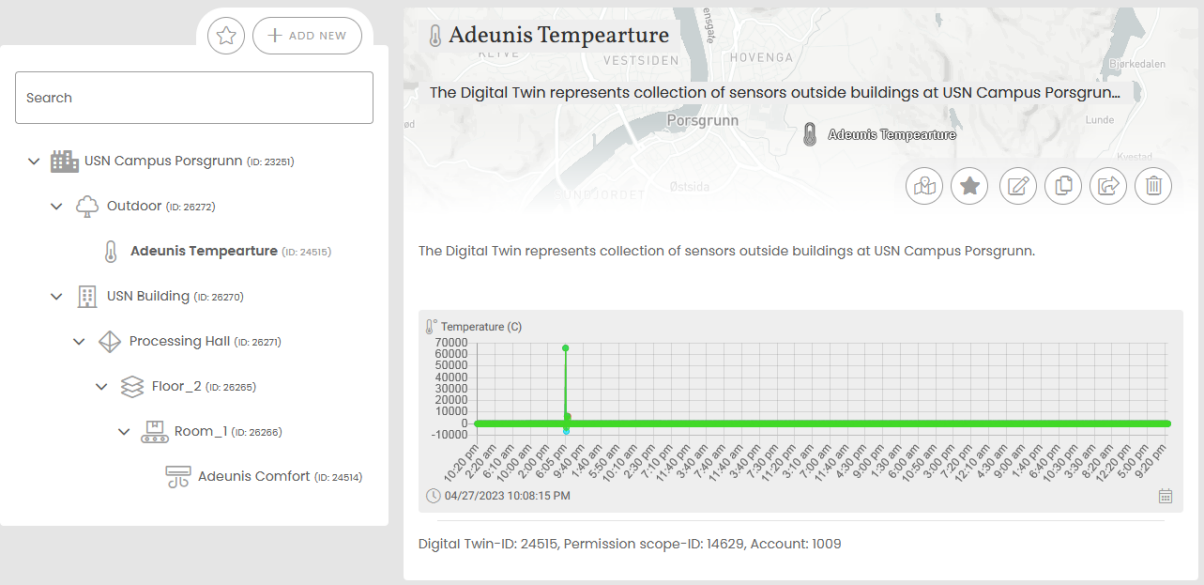


Figure 8.6: niotix: historical data contains test values.

Forwarding data further to application server hosted by Grafana Labs provides various ways to visualise data. Administrator can create and arrange multiple dashboards and panels that suite measurements best. From the user perspective, it will be possible to observe data and save it in a .csv file.

Values requested by Grafana from IoT Hub does not seem to be processed separately or modified. Therefore, any error in the source will propagate further. In this case, the negative temperature measured by Adeunis Temperature sensor was erroneously processed by niotix from “payload_hex”, as described above, and plotted as a value over 6500 degrees Celsius in Grafana (Figure 8.7).

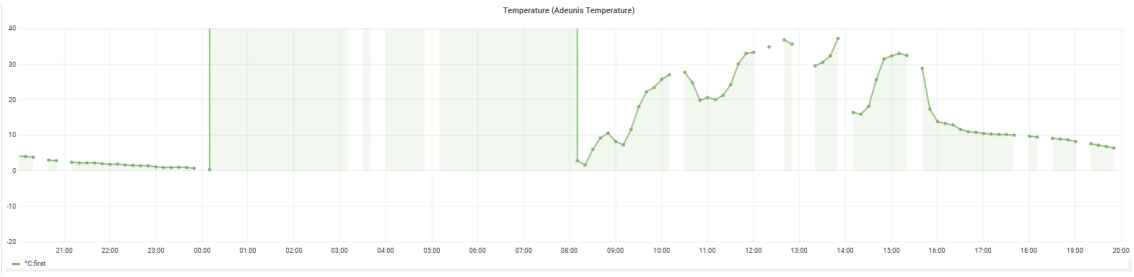


Figure 8.7: Grafana: plot with extreme values.

Transformation of values in Grafana is an experimental tool, there are however options for mathematical and statistical operation, but no conditional tests or user defined functions could be defined.

Another issue regarding Grafana is representation of values as a single time series. Administrator combining Grafana tools on the interface may define number of points that should comprise a graph, or time interval between the points. Each plot when displayed, occupies a specific area assigned to it by administrator. The plot then adjusts the number of points to be displayed out of the space it occupies.

This procedure, while being rational, may function misleading during operation of the interface, because user can change the overall time range for the data sets to display.

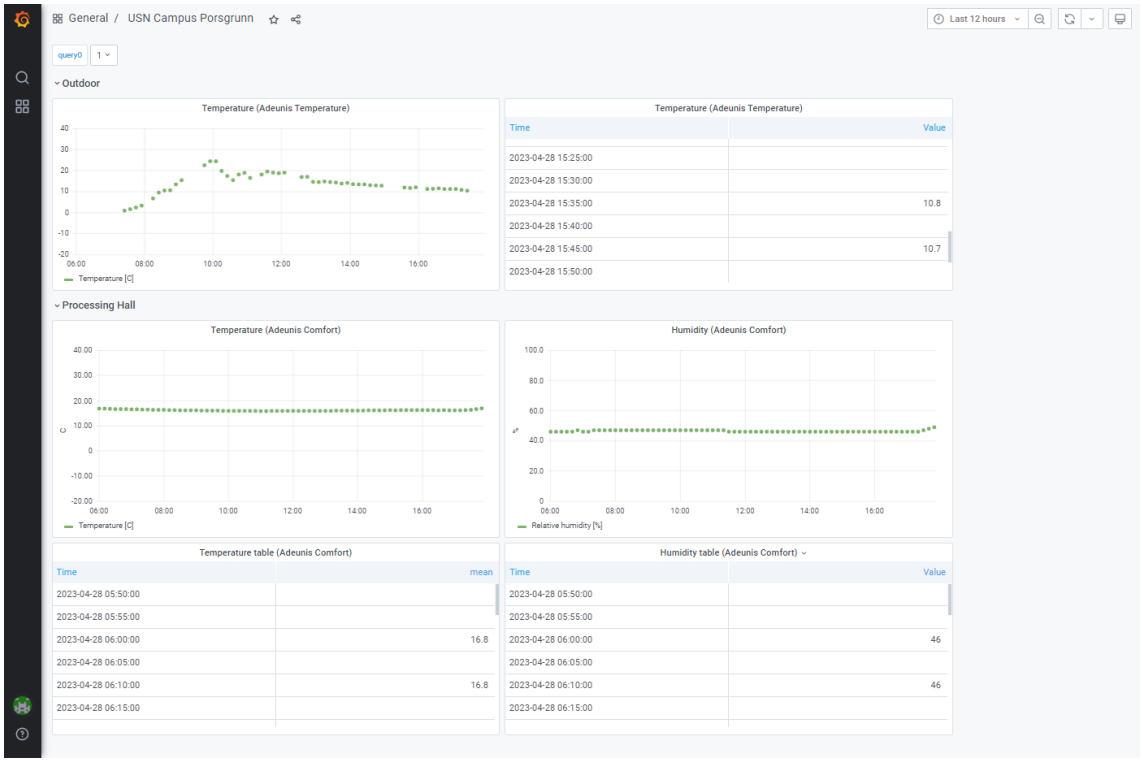


Figure 8.8: Grafana: short time range.

Changing of the time range affects the number of data points available for a plot display. When defining the number of points or time interval administrator may consider just a specific time range, such as 24 hours. But making the range shorter (Figure 8.8) may result in plotting of the points as separate independent lines. At the same time, when longer range is used (Figure 8.9), the values are averaged to fit the plot size, what may not represent the actual state of measurements.

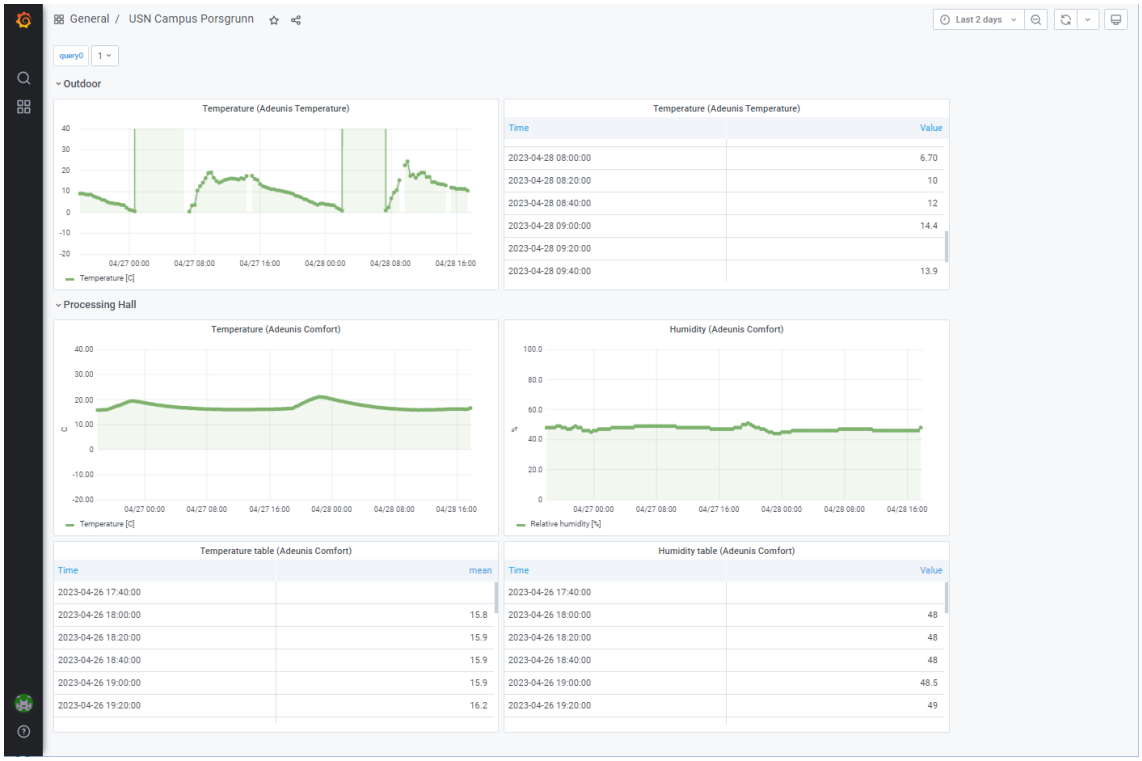


Figure 8.9: Grafana: long time range.

Data source is also an important detail in the Grafana service. Data from IoT Hub (Figure 8.10) and InfluxDB (Figure 8.11) may be actually represented differently. As one can see, the negative value from InfluxDB is shown as over 6000 value from IoT Hub. This fact indicates that the storages used in niotix are different, and the transformation of data does not relate changes to both destinations. It is also clear that the plots are slightly different in general, data obtained from the database seem to be averaged. This also should make administrator and user cautious about what data is plotted in the graphs.

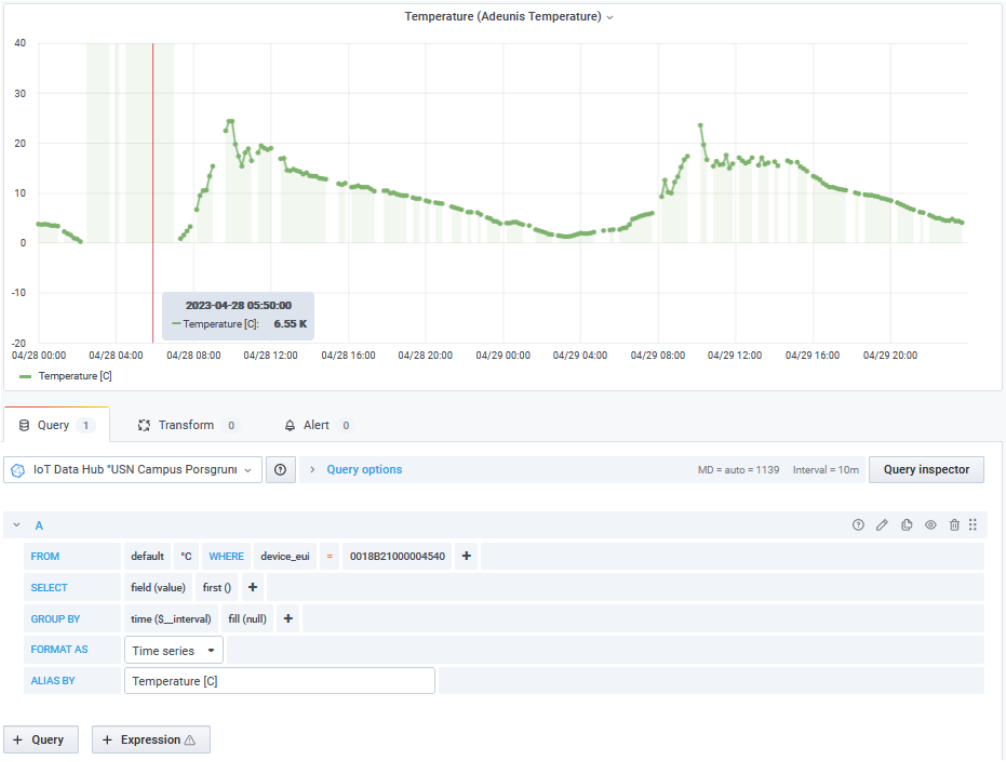


Figure 8.10: Grafana: request from IoT Hub.

Figure 8.11: Grafana: request from InfluxDB.

There are still some experimental or not relevant for this work features in Grafana that have not been highlighted in the work.

Node-RED dashboards represent alternative to niotix and Grafana servers that could be used for monitoring of data.

Direct connection between Node-RED and MQTT allows to display real-time data. That could also be possible to send data to devices using dedicated topics for DL. The significant disadvantage of the solution is that arrived data has never been stored anywhere and, therefore, cannot be requested. Of that reason, logging of data cannot be initiated for user defined time range. There could be implemented strategies for how to store the data automatically, such as logging data continuously or daily. But that does not solve the issue with requesting data for a specific period of time.

The dashboard connected to Dimension Four platform could solve the issue with storing data dynamically and retrieving the last or historical values. However, the queries are limited to return only 100 values at once. That is possible to implement a more sophisticated requests by examining the returned values and updating queries to return more values if not all are present in the response. But implementing this approach is limited by the number of calls to the Dimension Four API due to free subscription. This type of subscription limits also the number of sensors that can be connected to the service to 10. These limitations must be taken into account before establishing a network. If the number of sensors is enough, then frequency of data submission and retrieving must be evaluated.

The niotix dashboard developed in Node-RED retrieves data directly from Data states defined in niotix IoT Hub using the provided API. The clear disadvantage of the approach is that it is impossible to request historical data, only the last measurement is provided (shown with the gauge plot in the Results). In that sense, the result would be similar to the MQTT dashboard, if the requesting data would be initiated by a timer and not a human action. Also, the issue related to storing the plotted data arises, just as with MQTT. Besides, the implementation is a duplication of the existing niotix server and is relevant if an alternative application server is required.

Interface based on use of InfluxDB is a solution that provides a good functionality in terms of time range for requesting data and possibility to properly save it. This dashboard could be used instead of Grafana service as their interfaces are visually close to each other from the user point of view.

There is also a similarity between Grafana and Node-RED visualisation. The graph area is limited in both cases and user cannot observe more data points than fit the area of a chart. If there are too many points, Grafana averages them to display the general picture. Chart in Node-RED follows settings for the plot: it is possible to display 1000 points or values for 24 hours only. Therefore, if the requested range is 3 days, Node-RED displays only values for the first day.

9 Conclusion

During the current work, several technologies involved in development of IoT systems that use wireless communication have been analysed. The choice of any of the technologies must be based on users' needs and preferences. For the case when long range data transmission is required and devices have limited access to power storage, while useful data payload does not exceed 250 bytes, LoRaWAN protocol has shown to be a reliable solution.

The protocol has a certain level of security, which is supported by a sophisticated AES CMAC algorithm and special way of sharing security keys within the network. When a message is being transmitted, each node in its path can decode only the part addressed to that node. The implemented modulation for the signal, Chirp, has also shown to be a reliable method of data transmission.

As there is no perfectly safe system, some of the nodes, such as end-device and join server, may be compromised and the secret keys could become available to attackers. Also, the radio signal itself may suffer from jamming attacks. However, while the implemented security measures may not withstand distortion of transmissions, they will not allow stealing of data or hijacking of devices.

The structure of LoRaWAN requires a proper governing of the network. This results in organisation of public or private networks that are managed by communities or companies. In the current project the network was controlled by Altbox AS. The company has provided access to service, Actility ThingPark Wireless, that gives an opportunity to efficiently manage connected device and examine their performance. Functionality of the service strongly depends on the type of user logged in. Subscriber, as in this case, may add, remove, and edit devices and forward packets to a desired application server for presenting to an end-user.

At the same time, there are technical specifications that will not allow to perform these operations at ones will. If there is a desire to add a new device or a gateway, to extend the network range, this must be agreed with the system administrator and is out of control of subscriber. ThingPark Wireless is an in interface that hides operations and settings for gateway, join server and network server and, therefore, some steps in communication in the established network may not be clear. For instance, application server must be assigned an application session key from a join server to be able to exchange data with end devices. But when the server is specified in ThingPark Wireless using a URL, there is obviously no key that will be assigned to the server. That means that the custom application server is not the application server as defined by LoRaWAN, but ThingPark comprises functions of both join, network and application servers.

Another functional part of Actility services, ThingParkX, has demonstrated to be a useful tool in combining several types of destinations for measurements. The service allows to direct packets to multiple types of servers and specify which connection should be used by a particular device. A significant advantage of ThingParkX is that it is possible to transform the outgoing messages and implement JavaScript functionality for it. This opens a way to transmit data to servers that require a special format for incoming data. In this work, there have been established successful connection with Azure IoT Hub, HiveMQ MQTT broker and Dimension Four MQTT broker.

Further, several interfaces for monitoring and logging measurements have been established

One is niotix. The server is defined as destination in ThingPark Wireless and, therefore, does not require ThingParkX service. niotix provides all required function for a monitoring application. It allows to acquire data from sensors, modify it with JavaScript Transformer, store, and visualise it. Finally, user can save the data as a .csv file. Additionally, the server provides a very convenient way to represent the devices themselves by allowing to locate them geographically and even indoors with specification of floorplans. Such tool can be utilised with a special type of devices, trackable, to follow moving device. The service can be accessed by users with permission that allows only to monitor data and not to modify it. This is clearly beneficial, as the same application server may be securely accessed by public.

From niotix, user may also get access to Grafana visualisation service. User with administrator permission can organise interface in the service and give access for monitoring of measurements. The service has multiple visualisation tools, however, may not be as relevant for the task as niotix.

It has also been shown that erroneous measurements may be transmitted by the underlying service responsible for decoding of data. In that case, the data will be interpreted wrongly by the described application servers as well. To cope with that, the administrator may use transformation tools in forwarding service, Actility ThingParkX, if it is available, or take measures on the application server side. In case of niotix and Grafana, several storage methods have been implemented, but transformed data was saved only in one of them. The way the data is managed in background remains unclear and requires further investigation. At the moment, niotix application server should be preferred, but administrator must use the server with precaution, and perform analysis of incoming data for possible errors.

As an alternative the application server provided by Altibox, four dashboards have been developed in Node-RED programming tool. In contrast to niotix, the dashboards must be run on a local machine of a user or on a server managed separately from Altibox services. This limits functionality and extensibility of the dashboards because introduction of new devices and maintenance of a server may require additional development time.

Nevertheless, communication with LoRa devices may be organised using other providers than Altibox, as shown on the left side of Figure 8.1. In that case, integration of various source of data into one interface provided by one company may be problematic. Options for alternative LoRaWAN setup include open-source and community managed solutions. A source independent application server, such as one developed with Node-RED would be a reasonable solution for such system. It is also important to note that administrator can use a full spectra of JavaScript functionality in Node-RED. That will make it much easier to format the incoming data and not rely on the solutions from the network provider.

In future work, it would also be useful to consider development of end-devices and gateways with use of available LoRa modules for Arduino and Raspberry Pi. These devices could be then used in Altibox network, if the provider agrees to include them, connected to the other mentioned in the Figure 8.1 providers, or integrated in a private network developed at USN.

10 Bibliography

[1] Semtech Corporation, “The Challenges of IoT Connectivity,” Semtech Corporation, 29 08 2022. [Online]. Available: https://learn.semtech.com/mod/page/view.php?id=131&forceview=1. [Accessed 13 01 2023].

[2] J. P. Queralta, T. N. Gia, Z. Zou, H. Tenhunen and T. Westerlund, “Comparative Study of LPWAN Technologies on Unlicensed Bands for M2M Communication in the IoT: beyond LoRa and LoRaWAN,” in *The 14th International Conference on Future Networks and Communications (FNC)*, Halifax, 2019.

[3] K. Mekki, E. Bajic, F. Chaxel and F. Meyer, “A comparative study of LPWAN technologies for large-scale IoT deployment,” *Information & Communications Technology Express,* vol. 5, no. 1, pp. 1-7, 2017.

[4] SIGFOX, “Sigfox Technical Overview,” SIGFOX, Labrège, 2017.

[5] Sigfox , “0G NETWORK COVERAGE,” Sigfox , [Online]. Available: https://www.sigfox.com/coverage/. [Accessed 15 04 2023].

[6] LoRa Alliance Technical Committee, “RP002-1.0.3 LoRaWAN® Regional Parameters,” LoRa Alliance, Inc., Fremont, 2021.

[7] S. MONTAGNY, LoRa - LoRaWAN and Internet of Things for beginners, Chambéry: Université Savoie Mont Blanc, 2022.

[8] Semtech Corporation, “LoRa® and LoRaWAN®: A Technical Overview,” Semtech Corporation, Camarillo, 2019.

[9] ETSI, “Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz; Part 2: Harmonised Standard for access to radio spectrum for non specific radio equipment,” ETSI, Sophia Antipolis Cedex, 2018.

[10] LoRaWAN Alliance, “LoRaWAN® Coverage,” LoRaWAN Alliance, [Online]. Available: https://lora-alliance.org/lorawan-coverage/. [Accessed 20 04 2023].

[11] LoRa Alliance Technical Committee, “LoRaWAN® Backend Interfaces Technical Specification,” LoRa Alliance, Inc., Fremont, 2020.

[12] Semtech Corporation, “An In-depth Look at LoRaWAN® Class A Devices,” Semtech Corporation, Camarillo, 2019.

[13] LoRa Alliance Technical Committee, “LoRaWAN™ 1.1 Specification,” LoRa Alliance, Inc., Beaverton, 2017.

[14] LoRa Alliance Technical Committee, “LoRaWAN® L2 1.0.4 Specification (TS001-1.0.4),” LoRa Alliance, Inc., Fremont, 2020.

[15] Semtech Corporation, “An In-depth Look at LoRaWAN® Class B Devices,” Semtech Corporation, Camarillo, 2019.

[16] Semtech Corporation, “An In-depth Look at LoRaWAN® Class C Devices,” Semtech Corporation, Camarillo, 2019.

[17] Semtech Corporation, “Understanding LoRaWAN Gateways,” Semtech Corporation, 27 09 2022. [Online]. Available: https://learn.semtech.com/mod/page/view.php?id=47. [Accessed 05 03 2023].

[18] Semtech Corporation, “Understanding the LoRaWAN Network Server,” Semtech Corporation, 01 02 2023. [Online]. Available: https://learn.semtech.com/mod/page/view.php?id=35&forceview=1. [Accessed 05 03 2023].

[19] B. Reynders and S. Pollin, “Chirp Spread Spectrum as a Modulation Technique for Long Range Communication,” in *2016 Symposium on Communications and Vehicular Technologies (SCVT)*, Mons, 2016.

[20] ETSI, “Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz; Part 1: Technical characteristics and methods of measurement,” ETSI, Sophia Antipolis Cedex, 2017.

[21] E. Aras, G. S. Ramachandran, P. Lawrence and D. Hughes, “Exploring the Security Vulnerabilities of LoRa,” in *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, Exeter, 2017.

[22] J. Song, J. Lee, R. Poovendran and T. Iwata, “The AES-CMAC Algorithm,” 01 06 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4493. [Accessed 16 04 2023].

[23] Semtech Corporation, “Using LoRa for Geolocation,” Semtech Corporation, 12 09 2022. [Online]. Available: https://learn.semtech.com/mod/page/view.php?id=113&forceview=1. [Accessed 15 04 2023].

[24] Adeunis, “User Guide - FIELD TEST DEVICE - LoRaWAN,” Adeunis, Crolles, 2021.

[25] Adeunis, “LoRaWAN Smart Building COMFORT. Transceiver Temperature & Humidity. User Guide Version 2.1.1,” Adeunis, Crolles, 2018.

[26] Adeuins, “User Guide - TEMP - LoRaWAN,” Adeuins, Crolles, 2017.

[27] Microsoft, “IoT concepts and Azure IoT Hub,” 19 11 2022. [Online]. Available: https://learn.microsoft.com/en-us/azure/iot-hub/iot-concepts-and-iot-hub?WT.mc_id=Portal-HubsExtension.

[28] niotix, “niotix Manual,” niotix, [Online]. Available: https://docs.niotix.io/. [Accessed 31 03 2023].

[29] Grafana Labs, “Query and transform data,” Grafana Labs, [Online]. Available: https://grafana.com/docs/grafana/latest/panels-visualizations/query-transform-data/. [Accessed 31 03 2023].

11 Appendices

11.1 Appendix A. Project description.

FMH606 Master's Thesis

Title: Development and Testing of LoRaWan Sensor Network for Internet of Things Applications

USN supervisor: Hans-Petter Halvorsen

External partner: Altibox

Task background:

LoRaWan is short for Long Range Wide Area Network. LoRaWan is a low power wide area network that uses open-source technology. LoRaWan is designed for the Internet of Things (IoT). LoRaWan provides a far longer range than, e.g., WiFi and Bluetooth. It is also very suitable for outdoor measurements.

Task description:

In this project the LoRaWan infrastructure from Altibox and ThingPark will be used for development and testing of a LoRaWan sensor network. Different LoRaWan sensors will be available as part of the project.

The following tasks should be done in this project:

- Explore LoRaWan in depth and in general explore LoRaWan Applications and User Cases. Explore specifically the Altibox LoRaWan service, ThingPark and LoRaWAN Relays. Explore other providers and other alternatives.
- Create/Setup System for Logging and Viewing Sensor data from LoRaWan Sensors using Altibox LoRaWan service
- Application Server: Explore ThingPark X Connections and select 2-3 different types of connections for further implementation and use, e.g., HTTP, MQTT, Azure IoT Hub, Microsoft Teams
- Explore possibilities to connect to Dimension Four directly from ThingPark
- Explore and if possible, create a personal LoRaWan system with own Gateway, etc.
- Explore Data Security in context of LoRaWan and this project
- Testing, Installation and Deployment of System on Production Server.
- Include necessary scientific aspects, methods, and analysis
- The system needs to be properly documented so it possible for others to maintain and use the system after the project is finished

Student category: IIA, both campus and online, but also for industry master students that want to take a project outside their own company.

The task is suitable for online students (not present at the campus): Yes. Most of the work can be done online.

Practical arrangements: None

Supervision:

As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature):

Student (write clearly in all capitalized letters):

Student (date and signature):

11.2 Appendix B. Device raw packet content.

Example of a packet transmitted from a device.


```
{
  "DevEUI_uplink" : {
    "Time" : "2023-03-14T20:37:43.271+00:00",
    "DevEUI" : "0018B21000003BBF",
    "FPort" : "1",
    "FCntUp" : "2783",
    "ADRbit" : "1",
    "MType" : "2",
    "FCntDn" : "597",
    "payload_hex" : "4ce000ab30",
    "mic_hex" : "d3614b69",
    "Lrcid" : "00000201",
    "LrrRSSI" : "-46.000000",
    "LrrSNR" : "10.750000",
    "LrrESP" : "-46.350853",
    "SpFact" : "7",
    "SubBand" : "G1",
    "Channel" : "LC1",
    "Lrrid" : "10004D88",
    "Late" : "0",
    "LrrLAT" : "0.000000",
    "LrrLON" : "0.000000",
    "Lrrs" : {
      "Lrr" : [ {
        "Lrrid" : "10004D88",
        "Chain" : "0",
        "LrrRSSI" : "-46.000000",
        "LrrSNR" : "10.750000",
        "LrrESP" : "-46.350853"
      } ]
    },
    "DevLrrCnt" : "1",
    "CustomerID" : "100050938",
    "CustomerData" : {
      "loc" : null,
      "alr" : {
        "pro" : "ADRF/COMFORT",
        "ver" : "1"
      },
      "tags" : [ ],
      "doms" : [ ],
      "name" : "Adeunis - Comfort"
    },
    "BaseStationData" : {
      "doms" : [ ],
      "name" : "aib-entgw5-ufi-test"
    },
    "ModelCfg" : null,
    "DriverCfg" : {
      "mod" : {
        "pId" : "adeunis",
        "mId" : "comfort",
        "ver" : "1"
      },
      "app" : {
        "pId" : "adeunis",
        "mId" : "comfort",
        "ver" : "1"
      },
      "id" : "actility:adeunis-comfort:1"
    },
    "InstantPER" : "0.000000",
    "MeanPER" : "0.019608",
    "DevAddr" : "FC004EA5",
    "TxPower" : "2.000000",
    "NbTrans" : "1",
    "Frequency" : "868.1",
    "DynamicClass" : "A",
    "payload" : {
      "frameCounter" : 7,
      "configurationInconsistency" : false,
      "hardwareError" : false,
      "lowBattery" : false,
      "configurationSuccessful" : false,
      "type" : "TEMPERATURE_AND_HUMIDITY_READINGS",
      "register" : {
        "temperatureAndHumidityReadings" : [ {
          "offset" : 0,
          "temperature" : 17.1,
          "humidity" : 48
        } ]
      }
    },
    "downlinkUrl" : "https://altibox.thingpark.com/iot-flow/downlinkMessages/f5d4580e-5ee6-4c08-a4fa-dd0ecfb3d446"
  }
}
```


Figure 11.1: Comfort sensor raw packet sample.

11.3 Appendix C. MQTT setup with HiveMQ

On the HiveMQ portal after creating a user account one can create a cluster that can perform as a broker or subscriber. There are two options available at the time of making this project: free and payed (Figure 11.2). The free one has been chosen.

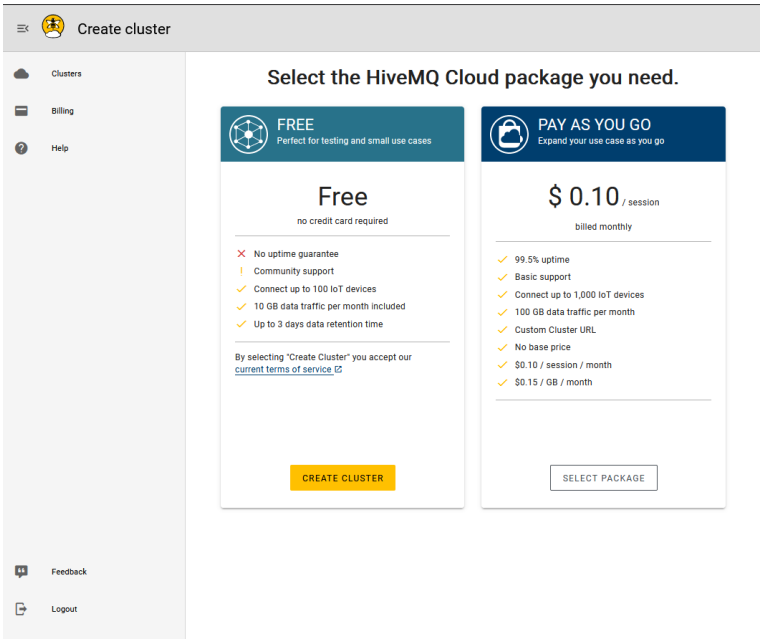


Figure 11.2: HiveMQ: cluster creation.

During cluster creation one has also an option to choose a hosting for that, it may be cloud service provided by Microsoft Azure or Amazon (Figure 11.3).

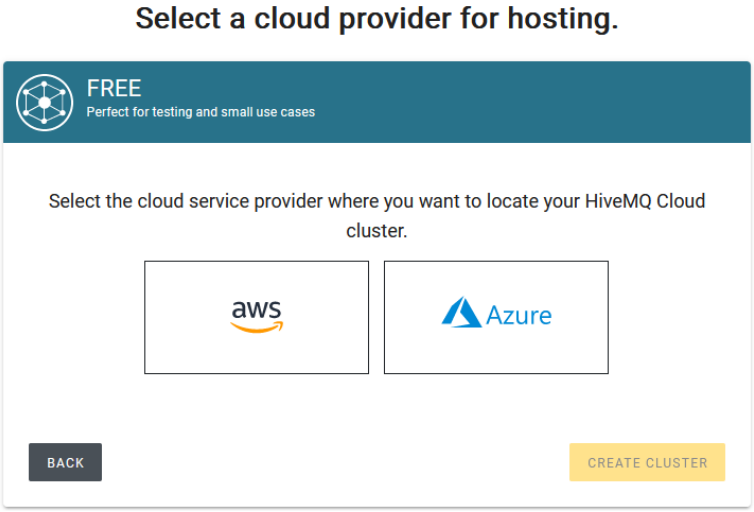


Figure 11.3: HiveMQ: available hostings.

In the current project Azure service has been chosen. After creation a cluster is assigned to a publicly available unique URL and has a defined port for secure communicating with MQTT server over Transport Layer Security (TLS) protocol (Figure 11.4).

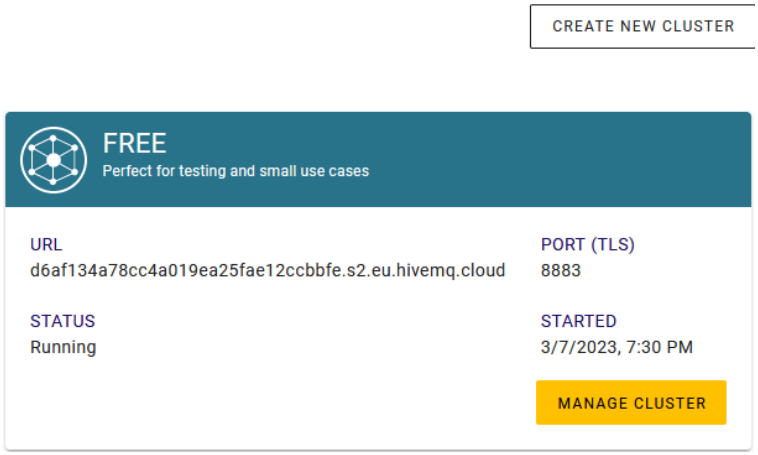


Figure 11.4: HiveMQ: hosting description.

In the "Manage cluster" menu section (Figure 11.5) user can view the current settings for the cluster, including alternative not secured port, option to change URL, number of transmitted messages and data traffic size.

From that menu one can manage access to the broker and use web client for publishing messages or subscribe to topics.

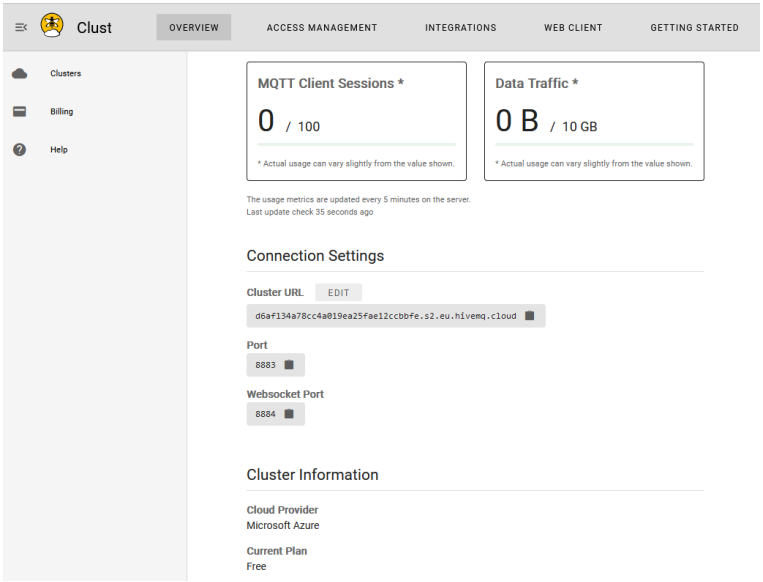


Figure 11.5: HiveMQ: cluster description.

In "Access management" (Figure 11.6) user creates an account that clients can use to communicate with the broker. It requires to specify username and a password. User may

create several credentials with different names and different permissions, such as only subscribe, only publish or both.

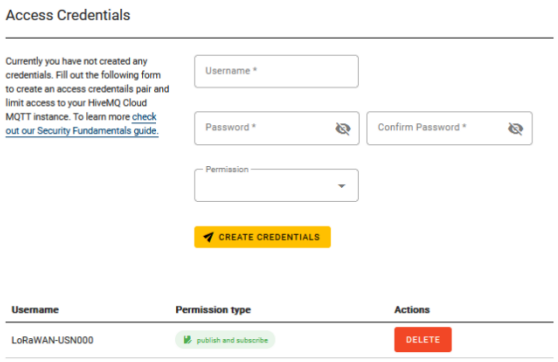


Figure 11.6: HiveMQ: cluster credentials setup.

After setting up a cluster and credentials one can start running broker using web-client provided by the HiveMQ service (Figure 11.7). It is possible to subscribe to any topic and listen to them and publish messages to that topics as well.

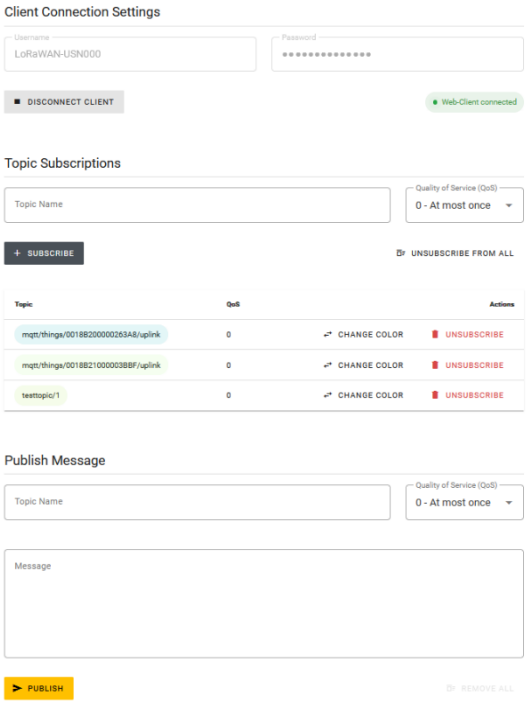


Figure 11.7: HiveMQ: running web-client.

Local application MQTTX (Figure 11.8) allows to create a client and publish and subscribe to topics on a MQTT broker.

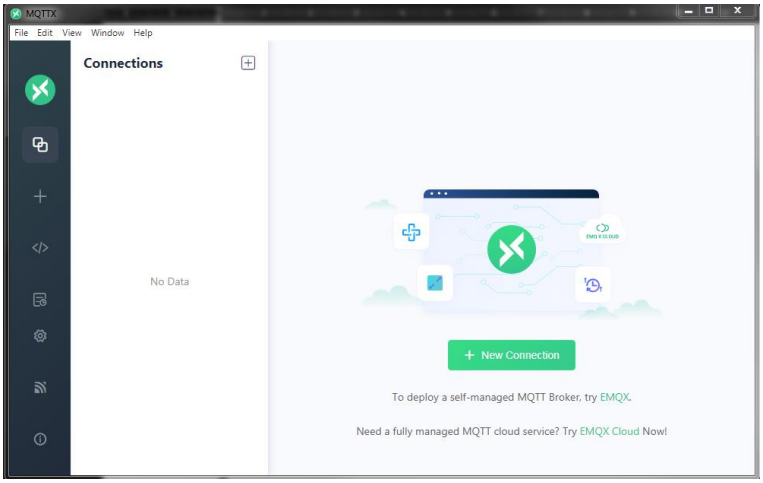


Figure 11.8: MQTTX: appclication home window.

To setup a connection in MQTTX one need to know the credentials of the broker (Figure 11.9). Client ID field can be specified by the user on local machine and is not related to the broker credentials. But it is important to give proper options for communication. If connection is supposed to use TLS, then Host URL must be given with mqtts:// scheme and "SSL/TLS" and "SSL Secure" options enabled.

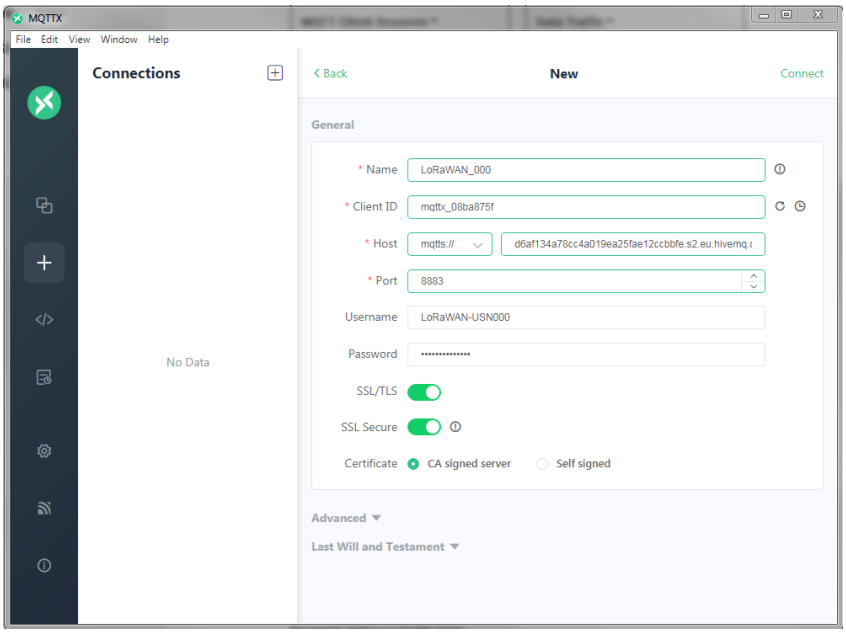


Figure 11.9: MQTTX: connection setup.

11.4 Appendix D. MQTT setup with Dimension Four

Dimension Four credentials have been used to establish a connection from ThingParkX to the storage (Figure 11.10).

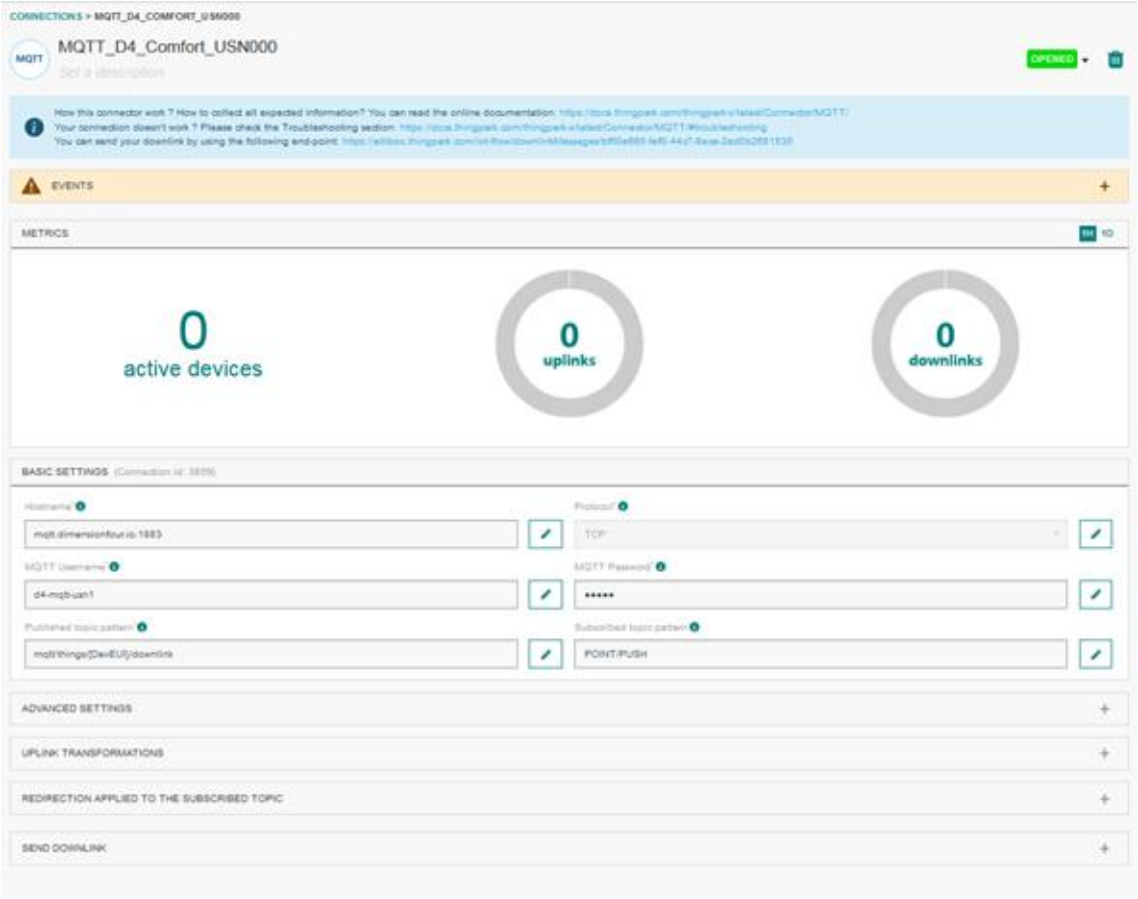


Figure 11.10: ThingParkX: MQTT connection to Dimension Four.

It is important to notice that by default ThingPark X uses secured connection with SSL, however, it is not implemented in Dimension four. That is why user need to change "Protocol" field in the Basic settings from SSL to TCP. One cannot subscribe to Dimension Four point, therefore, the subscribe topic left unchanged. At the same time, to store date there is only one topic available, "POINT/PUSH".

To correctly save data into a point, messages must have a special structure (Figure 11.11, Figure 11.12). It is important to specify correct pointId, tenantId and tenantKey. Further the data must be given as an array named "signals".

```
{
  "pointId":"6362ccc1fc18de50dedd2c78",
  "tenantId":"usn1",
  "tenantKey":<secret key>,
  "signals":[
    {
      "value": string(.DevEUI_uplink.payload.register.temperatureAndHumidityRead-
ings[0].temperature),
      "unit":"CELSIUS_DEGREES",
      "type":"Temperature",
      "timestamp": .DevEUI_uplink.Time,
      "metadata": {
            "DevEUI" : .DevEUI_uplink.DevEUI,
            "DevAddr" : .DevEUI_uplink.DevAddr,
            "payload_hex" : .DevEUI_uplink.payload_hex,
            "frequency" : .DevEUI_uplink.Frequency,
            "RSSI" : .DevEUI_uplink.LrrRSSI,
            "SNR" : .DevEUI_uplink.LrrSNR,
            "SF" : .DevEUI_uplink.SpFact,
            "latitude" : .DevEUI_uplink.LrrLAT,
            "longitude" : .DevEUI_uplink.LrrLON,
            "battery_low" : .DevEUI_uplink.lowBattery,
            "battery_voltage" : (.DevEUI_uplink.BatteryLevel - 1)/253*100
        }
    },
    {
      "value": string(.DevEUI_uplink.payload.register.temperatureAndHumidityReadings[0].hu-
midity),
      "unit":"PERCENTS",
      "type":"Humidity",
      "timestamp": .DevEUI_uplink.Time,
      "metadata": {
            "DevEUI" : .DevEUI_uplink.DevEUI,
            "DevAddr" : .DevEUI_uplink.DevAddr,
            "payload_hex" : .DevEUI_uplink.payload_hex,
            "frequency" : .DevEUI_uplink.Frequency,
            "RSSI" : .DevEUI_uplink.LrrRSSI,
            "SNR" : .DevEUI_uplink.LrrSNR,
            "SF" : .DevEUI_uplink.SpFact,
            "latitude" : .DevEUI_uplink.LrrLAT,
            "longitude" : .DevEUI_uplink.LrrLON,
            "battery_low" : .DevEUI_uplink.lowBattery,
            "battery_voltage" : (.DevEUI_uplink.BatteryLevel - 1)/253*100
        }
    }
  ]
}
```

Figure 11.11: ThingParkX: Comfort sensor to Dimension Four transformation.

```
{
    "pointId": "640ce2ef9d00b33bc24bce15",
    "tenantId": "usn1",
    "tenantKey": <secret key>,
    "signals":[
        {
            "value": string(.DevEUI_uplink.payload.temperature),
            "unit": "CELSIUS_DEGREES",
            "type":"Temperature",
            "timestamp": .DevEUI_uplink.Time,
            "metadata": {
                "DevEUI" : .DevEUI_uplink.DevEUI,
                "DevAddr" : .DevEUI_uplink.DevAddr,
                "payload_hex" : .DevEUI_uplink.payload_hex,
                "frequency" : .DevEUI_uplink.Frequency,
                "RSSI" : .DevEUI_uplink.LrrRSSI,
                "SNR" : .DevEUI_uplink.LrrSNR,
                "SF" : .DevEUI_uplink.SpFact,
                "latitude" : .DevEUI_uplink.LrrLAT,
                "longitude" : .DevEUI_uplink.LrrLON,
                "battery_low" : .DevEUI_uplink.lowBattery,
                "battery_voltage" : (.DevEUI_uplink.BatteryLevel - 1)/253*100
                }
        }
    ]
}
```

Figure 11.12: ThingParkX: Temperature sensor to Dimension Four transformation.

An important specification in transformation is that all numerical values must be converted to strings. Therefore, “values” fields include type casting with JSLT built-in function “string()”. “Units”-field have defined values, specified in documentation to the Dimension Four API, and “type”-field is a free text field. If there is additional information that should be supplied with the signal, it may be added to a “metadata” that does not have predefined elements.

11.5 Appendix E. Connection to Azure IoT Hub

To create IoT Hub one must first visit available Microsoft Azure service (Figure 11.13).

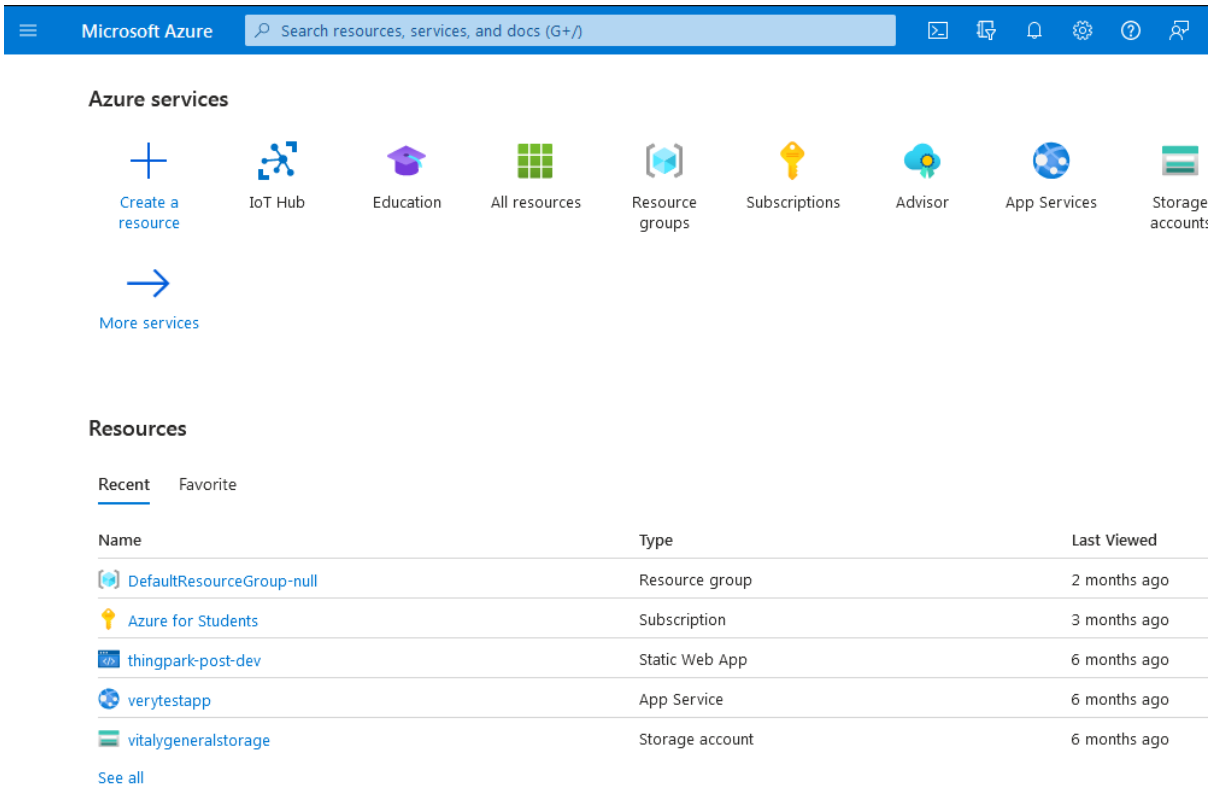


Figure 11.13: Azure IoT Hub: home window.

From the list one choses the desired service and enters setup section (Figure 11.14).

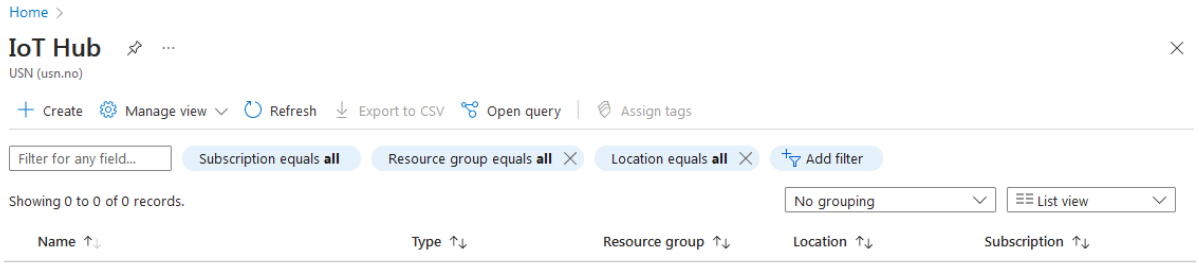


Figure 11.14: Azure IoT Hub: creation.

“Create”-button will then initiate create process, where user must go through several pages of settings, starting with basics (Figure 11.15).

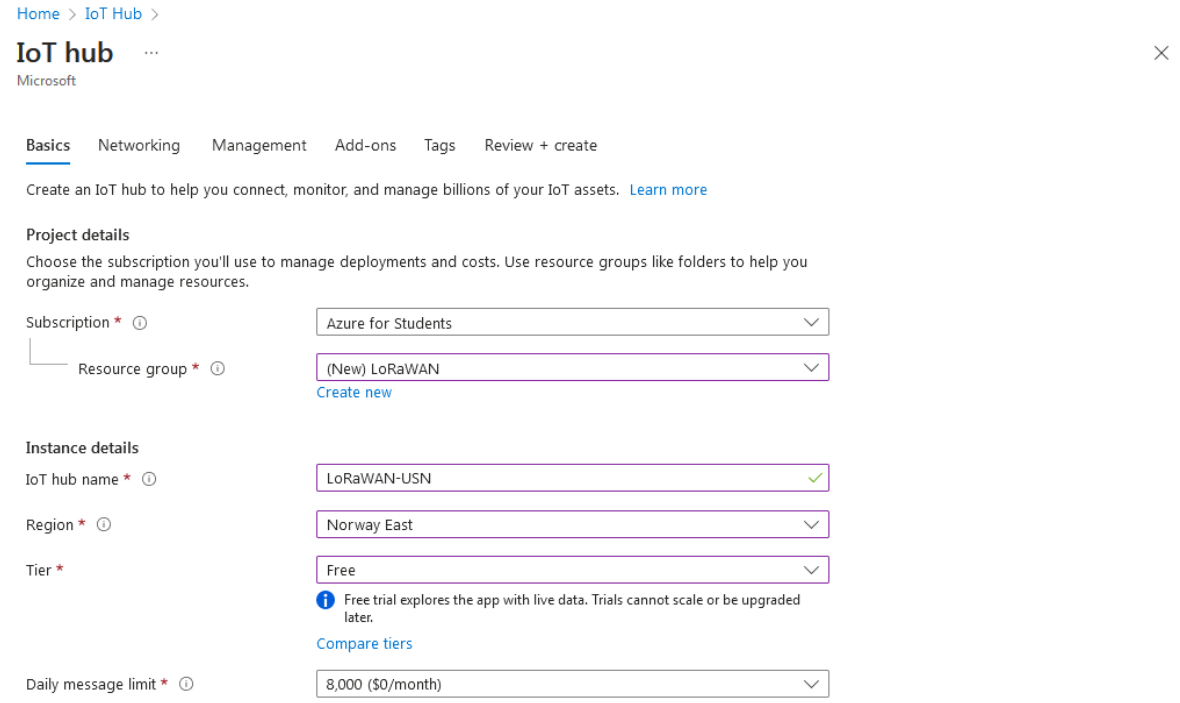


Figure 11.15: Azure IoT Hub: basics tab.

It is important that user has an active account. Then user may create a new “Resource group” that will be related to the hub. With free student license number of daily message transmissions is limited to 8000.

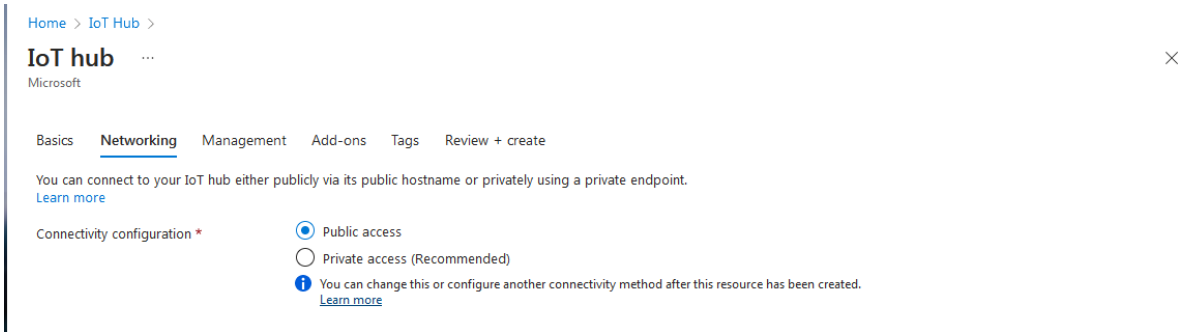


Figure 11.16: Azure IoT Hub: networking tab.

For the current project it was desired to have a publicly available data storage, therefore, in Network setting (Figure 11.16) “Public access” has been chosen. In case of private access, a more detailed settings required.

There were no specific management considerations done in regarding the project, and less strict “Shared access policy + RBAC” has been used (Figure 11.17).

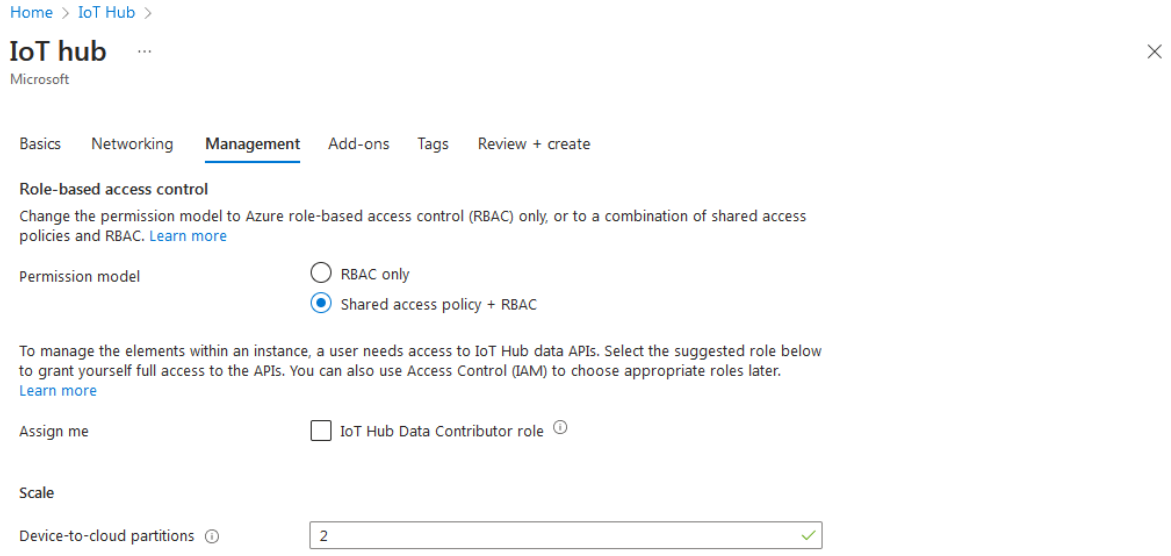


Figure 11.17: Azure IoT Hub: management tab.

Figure 11.18 and Figure 11.19 demonstrate additional settings for the hub, such as add-ons and a tag, but none of these has been used.

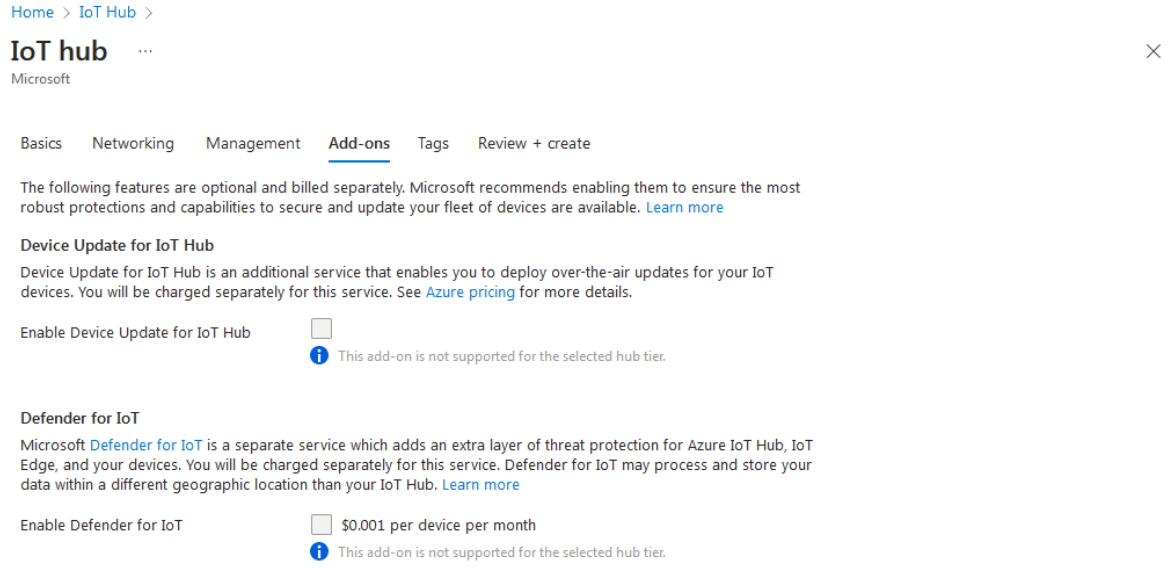


Figure 11.18: Azure IoT Hub: add-ons tab.

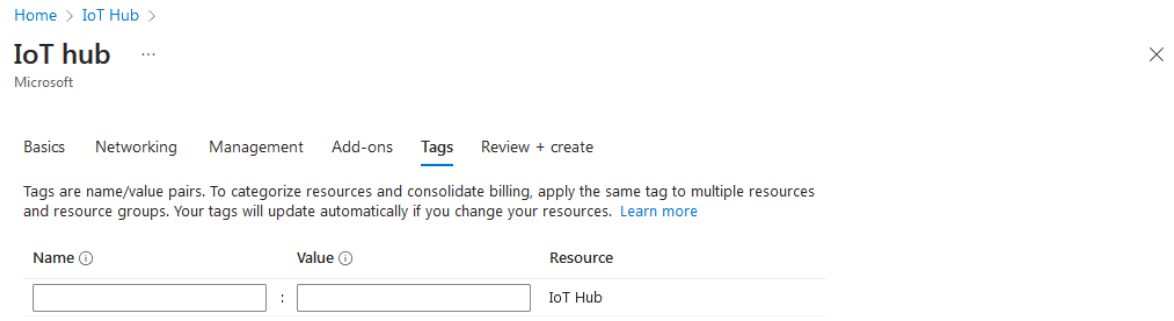


Figure 11.19: Azure IoT Hub: tags tab.

After the hub is been created, user has a list of predefined shared access policies, that may be used for communication with the hub. However, it is possible to create a new one with a specific name, such as "ThingParkX_policy" (Figure 11.20).

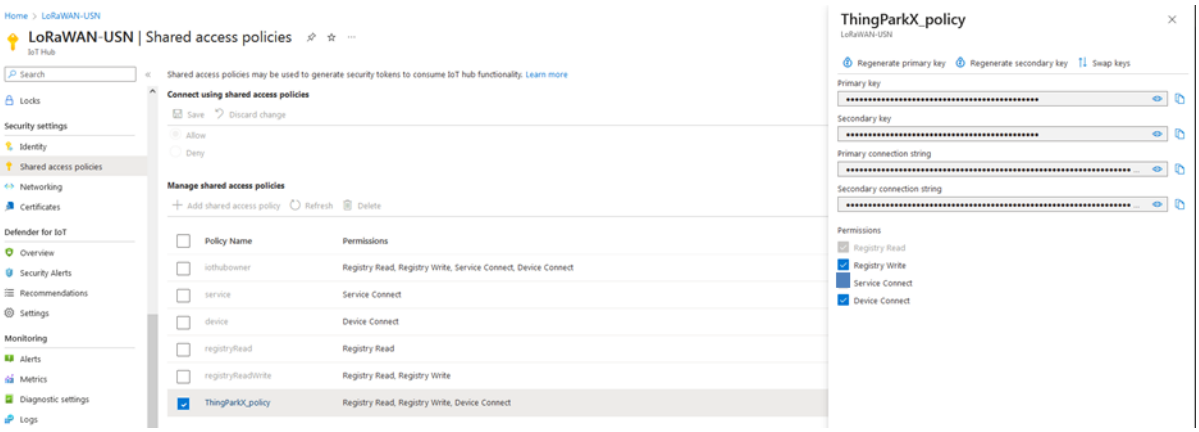


Figure 11.20: Azure IoT Hub: shared access policies list.

The keys related to the policy are automatically generated. The marked permissions by default are "Registry Write" and "Device Connect". Therefore, to allow the hub to receive telemetry from a device it is also important to mark "Service Connect" permission.

Overview of the created hub (Figure 11.21) contains the status and identity of the service after specifications.

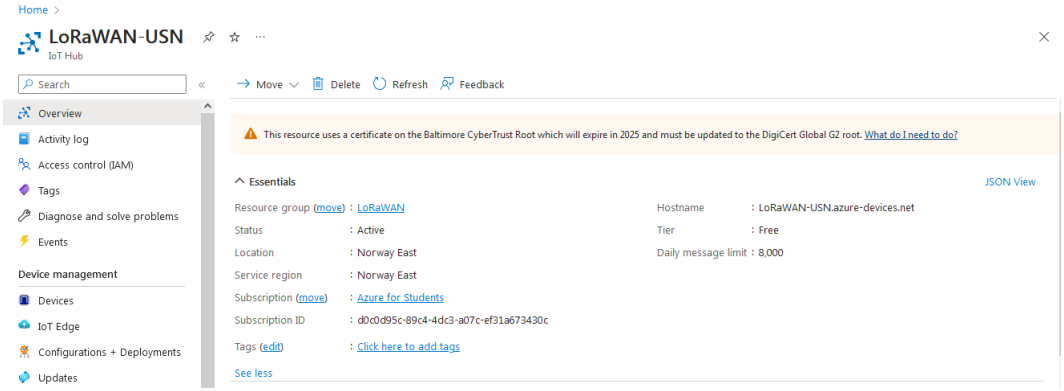


Figure 11.21: Azure IoT Hub: overview.

The next step to establish data transmission from LoRa devices to Azure is to setup ThingParkX connection (Figure 11.22). For that purpose user needs hostname from IoT hub overview page (Figure 11.21), name of the Shared access policy and one of the corresponding keys (Figure 11.20). IoT Hub Tier “F1” corresponds to the free subscription, such as student one.

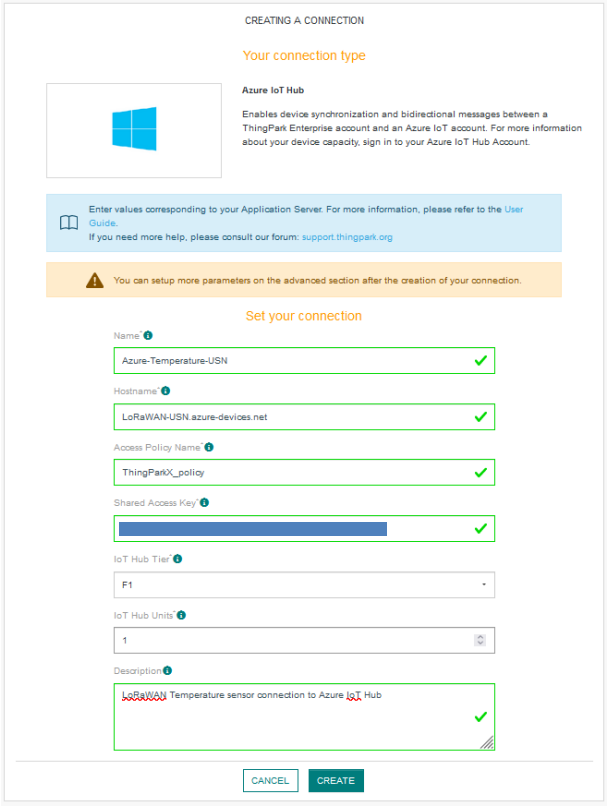


Figure 11.22: ThingParkX: Azure connection.

To configure Azure IoT Explorer one must follow the steps described below.

User starts with adding a new connection and specifying credentials in a similar way as for ThingParkX (Figure 11.23). The values are copied from the window describing Shared access policy named ThingParkX_policy.

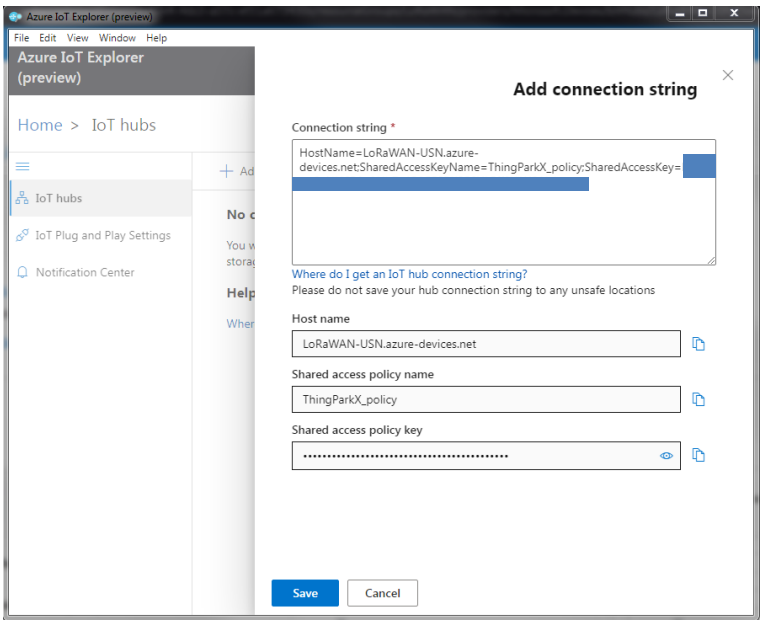


Figure 11.23: Azure IoT Explorer: create connection.

However, this specification can be filled in two different ways. On can either enter values for "Host name", "Shared access policy name" and "Shared access policy key" or enter the connection string as shown in the figure above. After adding connection string, other fields are filled automatically.

After connection creation one can observe a list of registered devices, that is empty at first (Figure 11.24).

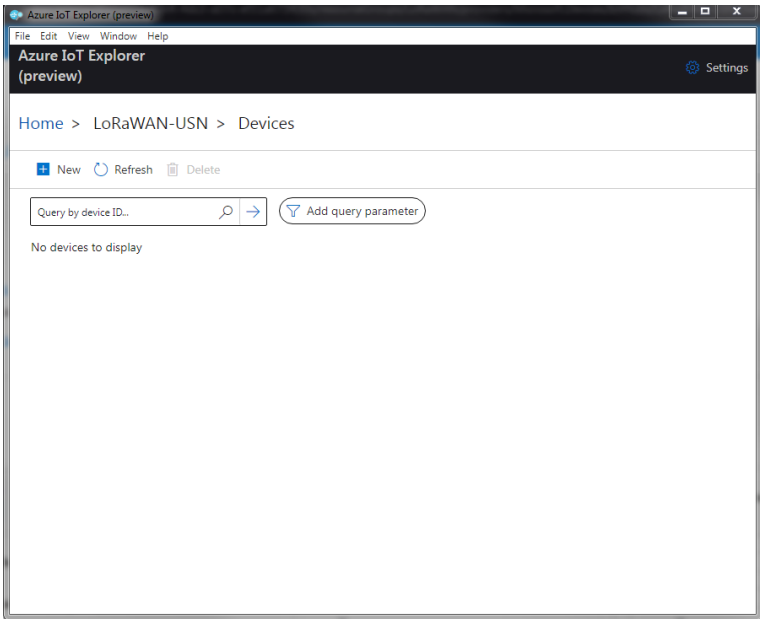


Figure 11.24: Azure IoT Explorer: device list.

Activation of “New”-button opens a menu for specifying LoRa device DevEUI and choosing method for authentication (Figure 11.25). Automatically generated symmetric key is satisfactory for that communication. One should also enable “Connect this device to IoT Hub”.

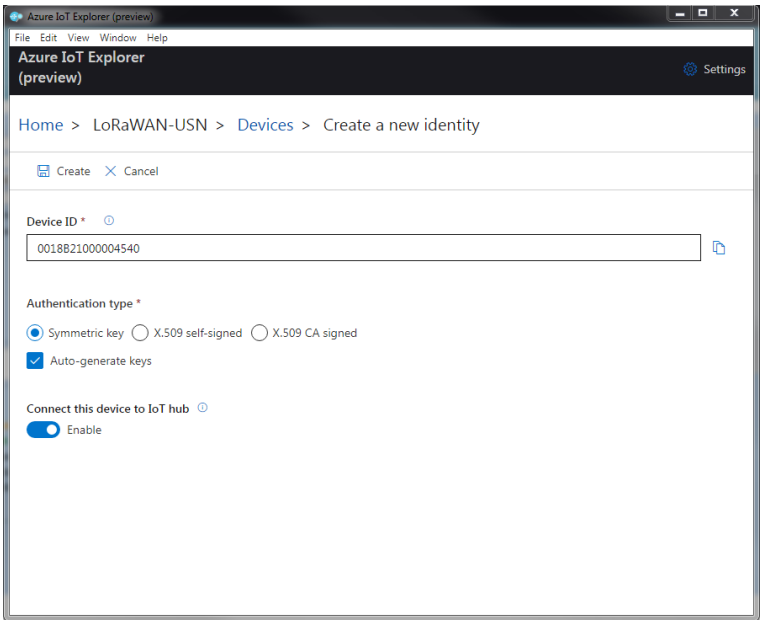


Figure 11.25: Azure IoT Explorer: device creation.

The newly created device credentials are then automatically filled in the corresponding fields (Figure 11.26).

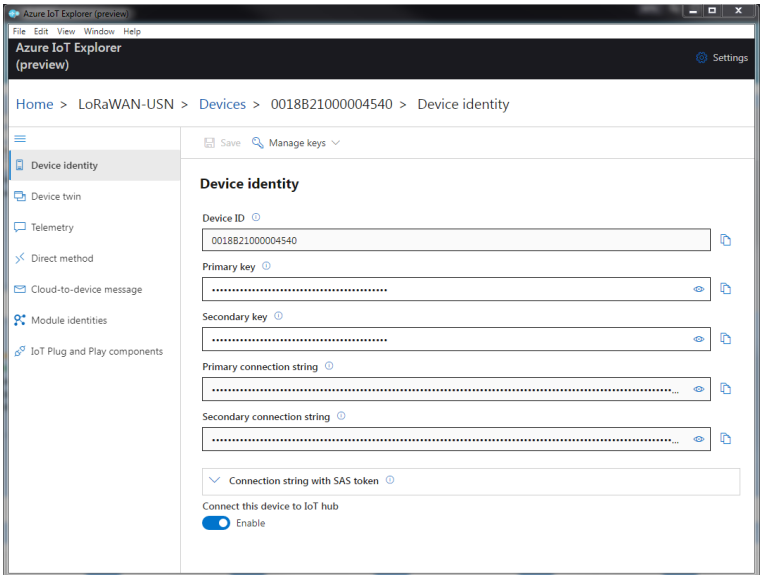


Figure 11.26: Azure IoT Explorer: device credentials.

After creating of the device in Azure IoT Explorer, it is also added to the IoT Hub automatically (Figure 11.27).

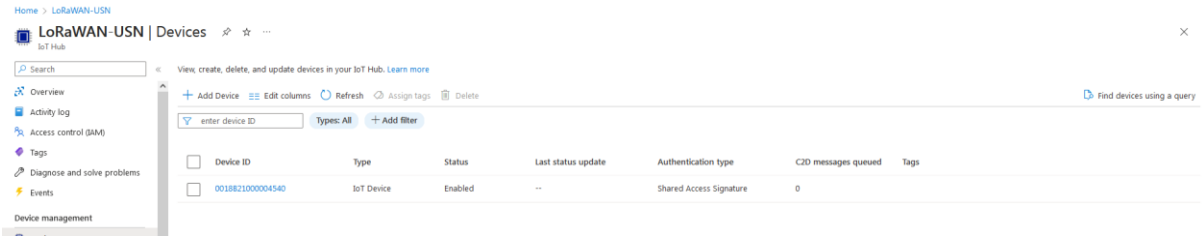


Figure 11.27: Azure IoT Hub: device list.

11.6 Appendix F. niotix

User creates a new Digital Twin by activating "+ Add new" – button and providing name.

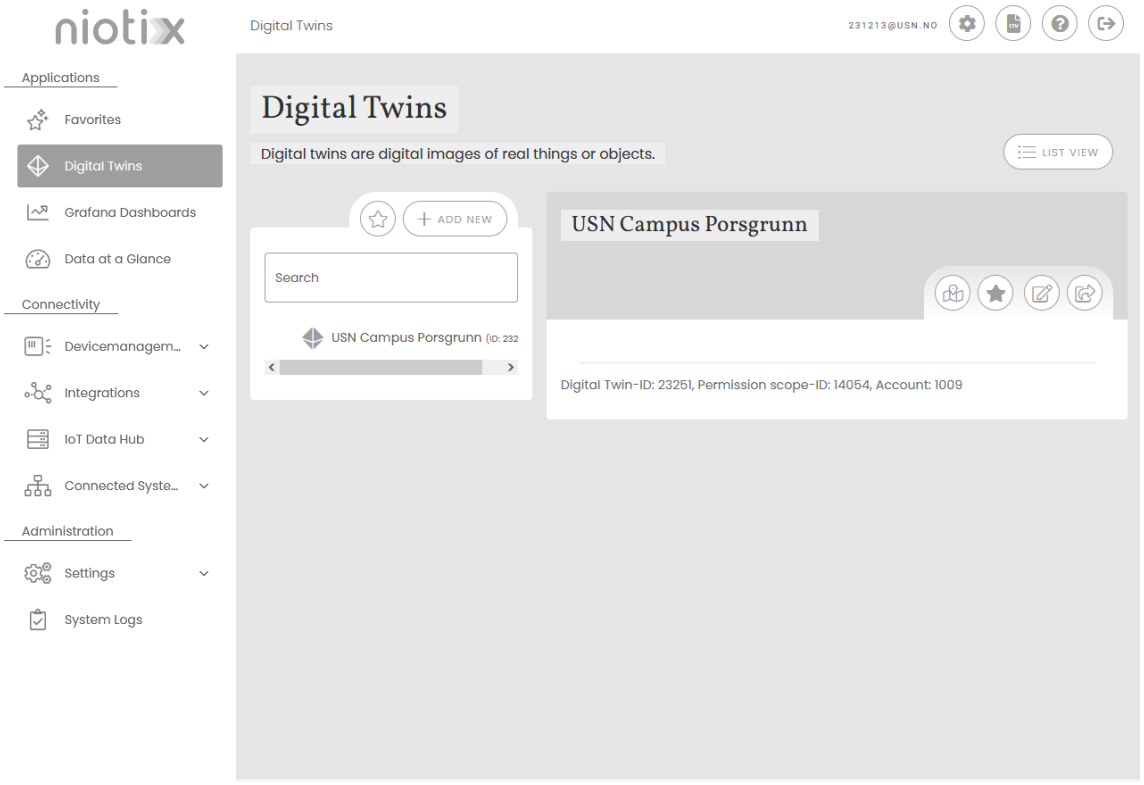


Figure 11.28: niotix: home window.

To be able to receive messages from devices a Digital Twin must be assigned an application. It is done by choosing "Application" sub-section in "Connectivity"-tab to the left of home window under "IoT Data Hub" section and then choosing "+ Create" in the appearing window (Figure 11.29).

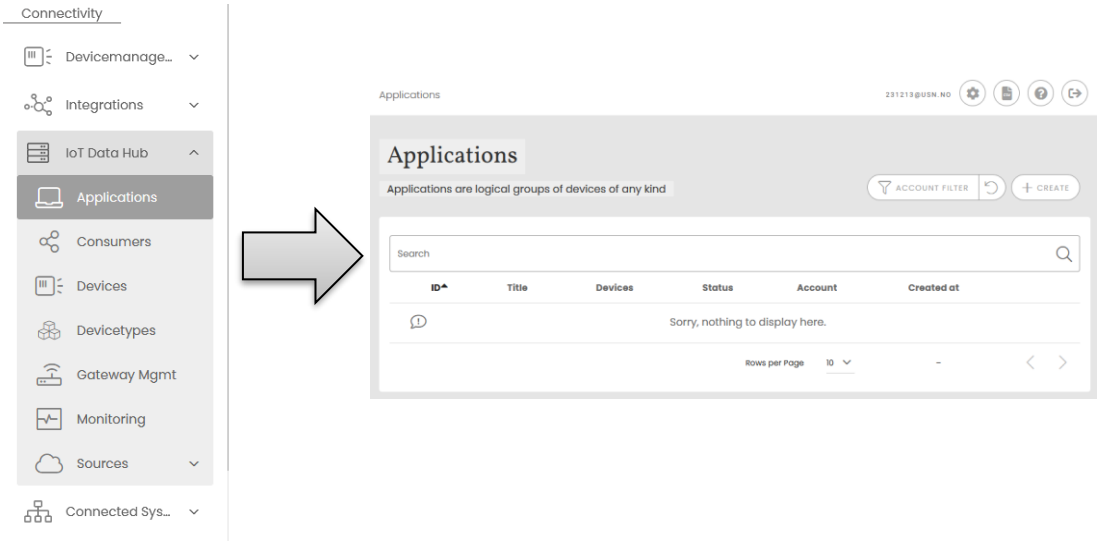
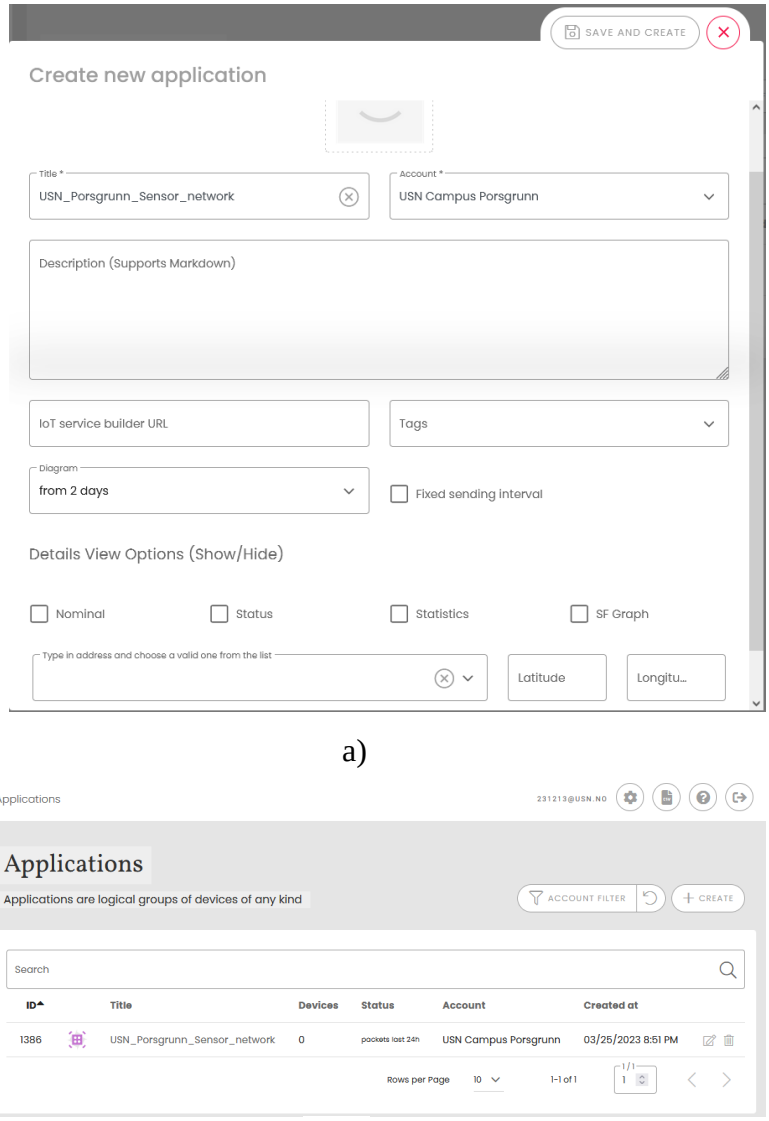


Figure 11.29: niotix: application creation.

The minimal setting for an application is to provide the title (Figure 11.30, a)). Account value is the existing Digital Twin that will be hosting the application. If there are several existing Digital Twins already registered for the user, one of them must be chosen.

After application has been created, it appears in the list of applications (Figure 11.30, b)).



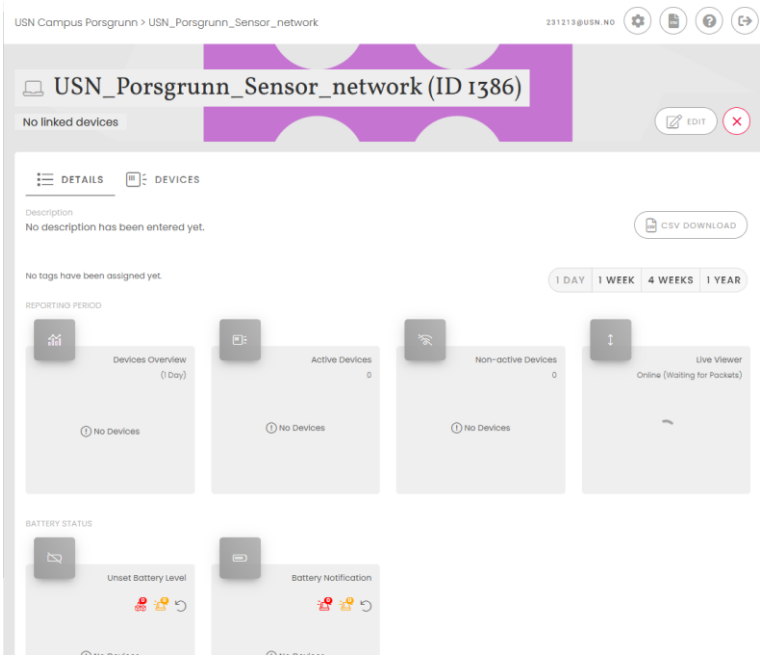
a)

b)

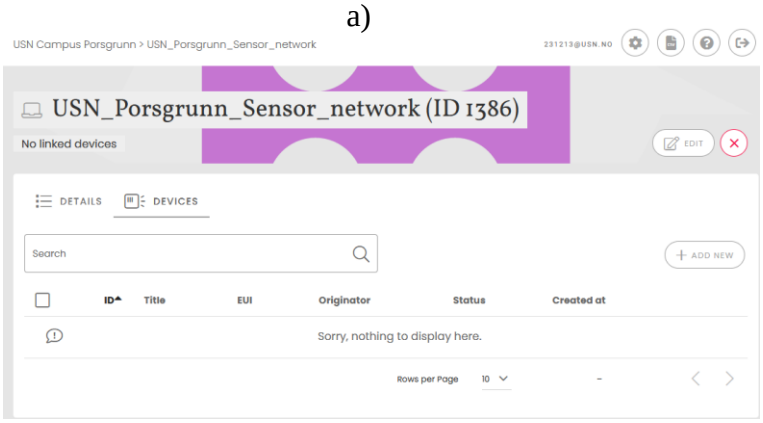
Figure 11.30: niotix: a) application properties, b) applications list.

After an application has appeared in the list, it may be set up. By opening the application, user finds first overview of events, number of devices, transmitted packets, and other statistic (Figure 11.31, a)). Opening “Devices”-tab allows to add, remove, and modify devices assigned to the application (Figure 11.31, b))

To create a new device one can activate “+ Add new”-button in the “Devices”-tab.



a)

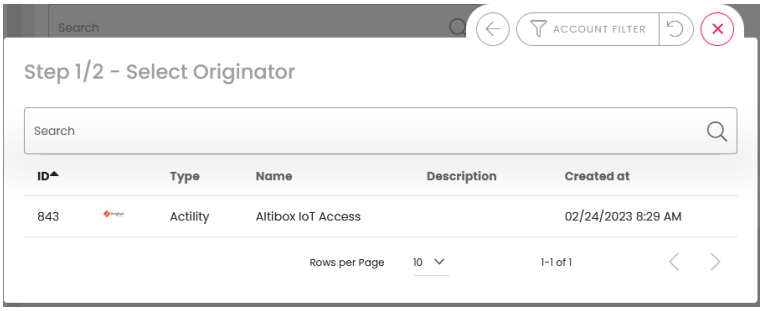


b)

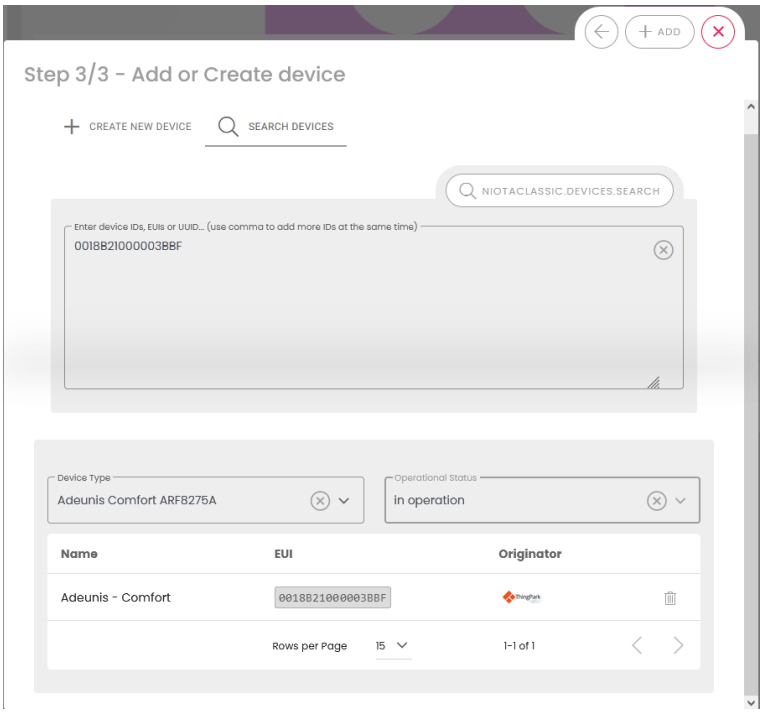
Figure 11.31: niotix: a) application details, b) devices connected to application.

The process of creating a device will depend on the origin of the device. As the access to niotix has been provided in cooperation with Altibox, other services related to the operator have also been enabled for niotix. Of that reason, creation of a device took fewer steps than it normally would take (Figure 11.32).

In step 1 (Figure 11.32, a)), only option for “Originator” is available, Actility, where niotix obtains the required for step 2 information and goes directly to step 3 (Figure 11.32, b)).



a)



b)

Figure 11.32: niotix: a) device creation step 1, b) device creation step 3.

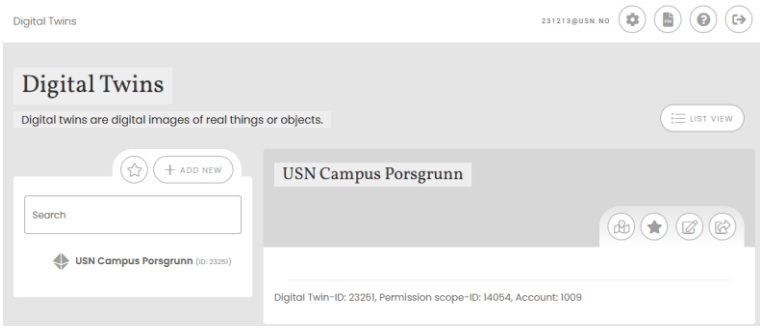
To activate already existing device, user can find it by choosing “Search devices”, inserting devices DevEUI and click “NIOTACLASSIC.DEVICES.SEARCH”-button. List of available devices with this ID appears under the insertion field. That must be only one device, as DevEUI is unique. Activating “Add”-button adds the device to the list.

It is also important to note that, while “Device Type” is not marked as mandatory field, it should be filled as well. One must choose a proper device from the appearing drop-down list. If that field is not chosen, the device will not appear in IoT Hub in the next step. Properties of the device may also be modified afterwards by finding it in the “Device” item in the menu.

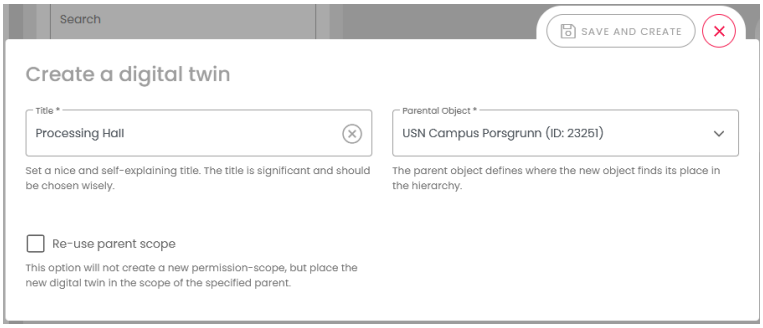
Application details will then be updated and graphs depicting quality of the signal and number of transmitted packages will be displayed.

Under the existing Digital Twin, it is possible to create sub-twins connected to separate devices or having specific properties. To do that, user must select existing Digital Twin and activate “+

Add new”-button (Figure 11.33, a)). Then a new window will appear providing field for name of the new Digital Twin and showing the selected one as “Parental Object” (Figure 11.33). If there are already existing several Digital Twins and the correct “Parental Object” is not shown, user must choose one manually from the drop-down list.



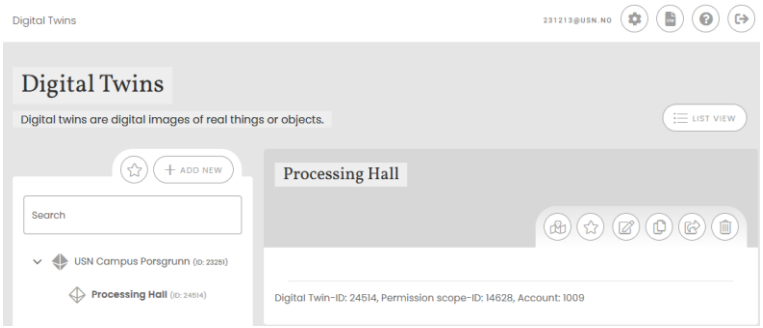
a)



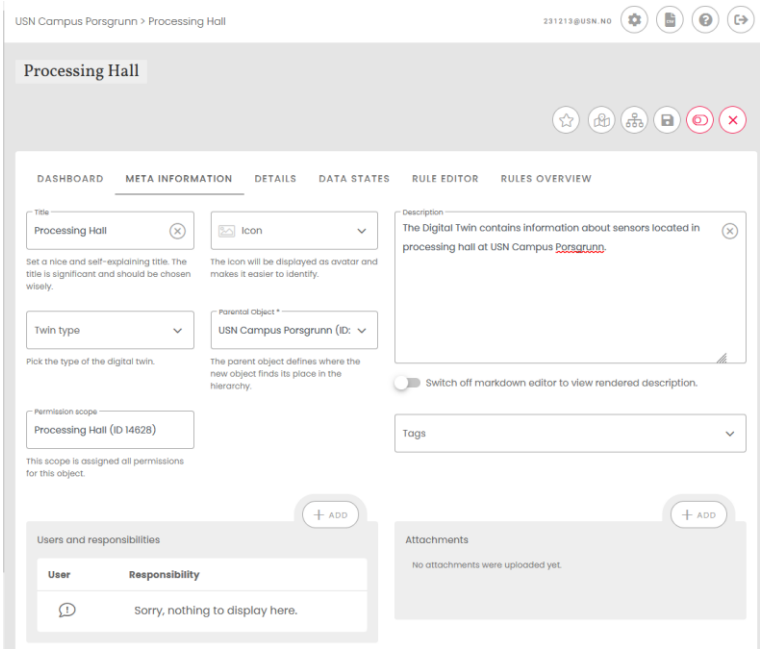
b)

Figure 11.33: niotix: a) parent Digital Twin, b) nested Digital Twin.

Further the new Digital Twin may be added a metadata and type (Figure 11.34).



a)



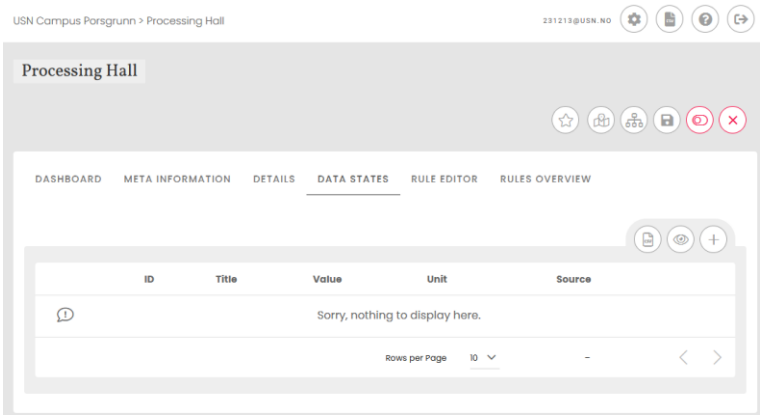
b)

Figure 11.34: niotix: a) Digital Twin structure, b) nested Digital Twin properties.

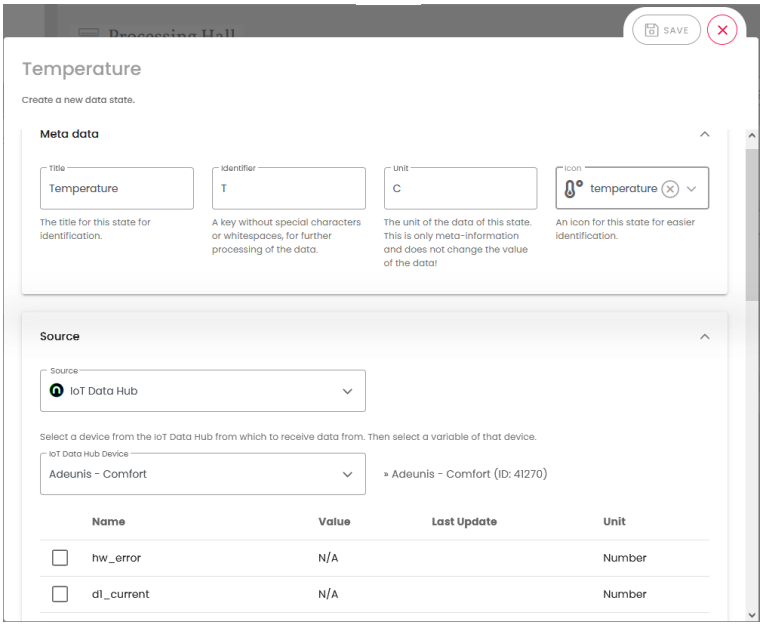
There may be multiple levels of nested Digital Twins. Each twin can be of different type, such as “Default”, “Building”, “Sensor”, “Room”, “Door”, “Zone”, “Floor” or “Trackable” [28]. These types can be created and related logically to each other, e.g. “Building” nests one or several “Floor”, which nests “Room” and so on. All of them may include information about location, however, in addition to static geographical information “Building”, “Floor”, “Zone” and “Trackable” types may be used to follow the change of location of objects and their presence in one zone or another [28]. On the shown webpage (Figure 11.34), these properties are set in “Details”-tab.

Data stream to a Digital Twin is represented as “Data State” in niotix. It can be added by activating on “+” button (Figure 11.35, a)), after that a specification menu appears (Figure 11.35, b)).

It is possible to specify a desired “Title”, “Identifier”, and “Unit”. “Icon” can be chosen from the list of provided images. It is important that the device type has been properly specified during its creation. In the current project “IoT Data Hub” is the source of data and “Adeunis - Comfort” is the type of the device that must appear in the drop-down list.



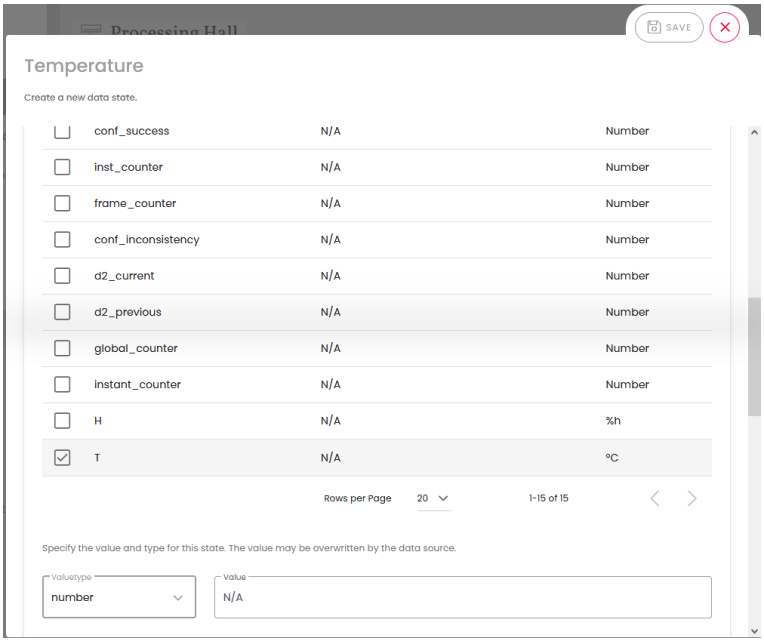
a)



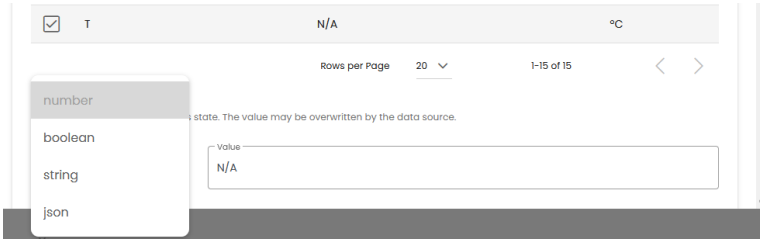
b)

Figure 11.35: niotix: a) Data state list, b) Data state basic properties.

In the section “Source” under the device type, a list of available values appears. It is possible to choose only one specific value. In this case, it is “T” for temperature. Other names may not be intuitive, and user must consult documentation first if the source of interest is not present.



a)



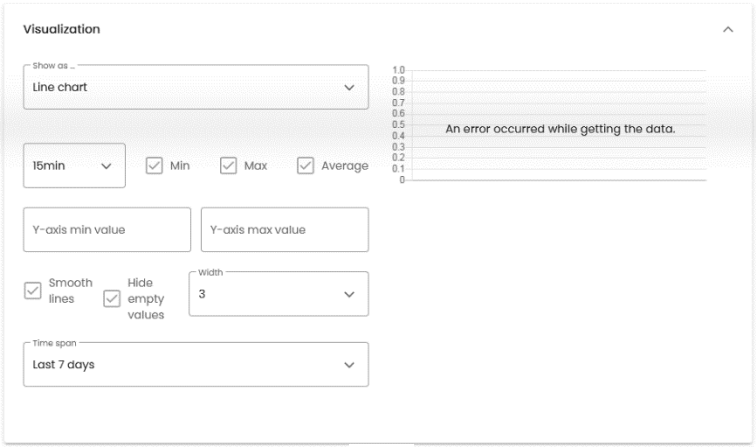
b)

Figure 11.36: niotix: a) Data state sources, b) Data state source type.

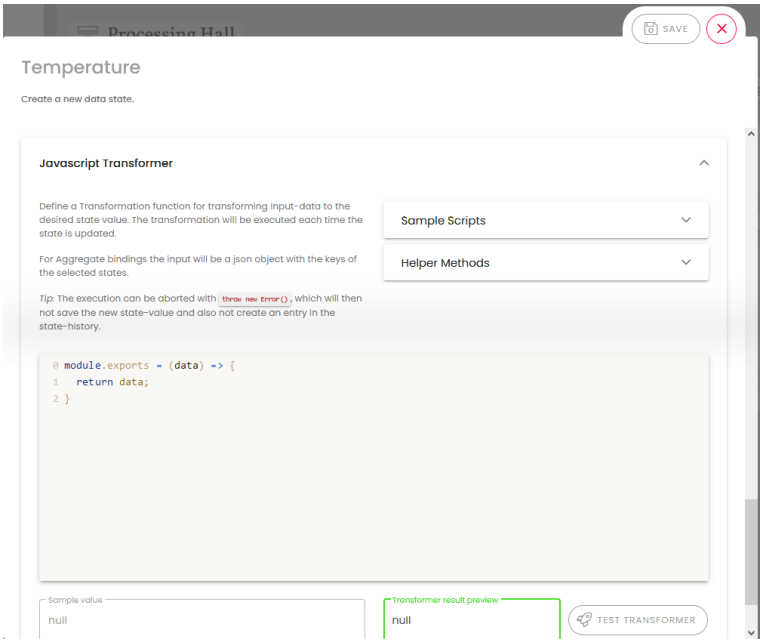
In addition, user may specify the type of data that will arrive (Figure 11.36, b)). There are four available options: number, Boolean, string and JSON for more complex data. In this project temperature is considered a numeric value.

In the next sub-section of the menu, user may choose a specify type of visualisation, which suits incoming data best, aggregate vales for the specific data stream, and time range. These properties may be modified afterwards.

When data is complex, i.e. JSON object, or requires preconditioning before plotting or storing, user may implement JavaScript algorithm to modify it in “JavaScript Transformer”-sub-section (Figure 11.37, b)).



a)



b)

Figure 11.37: niotix: a) Data state visualisation settings, b) Data state source transformation.

There is also alternative way to initialise procedure for creation of a device. A device may be created directly from the menu (Figure 11.38) and then used with applications. The procedure of creation remains the same.

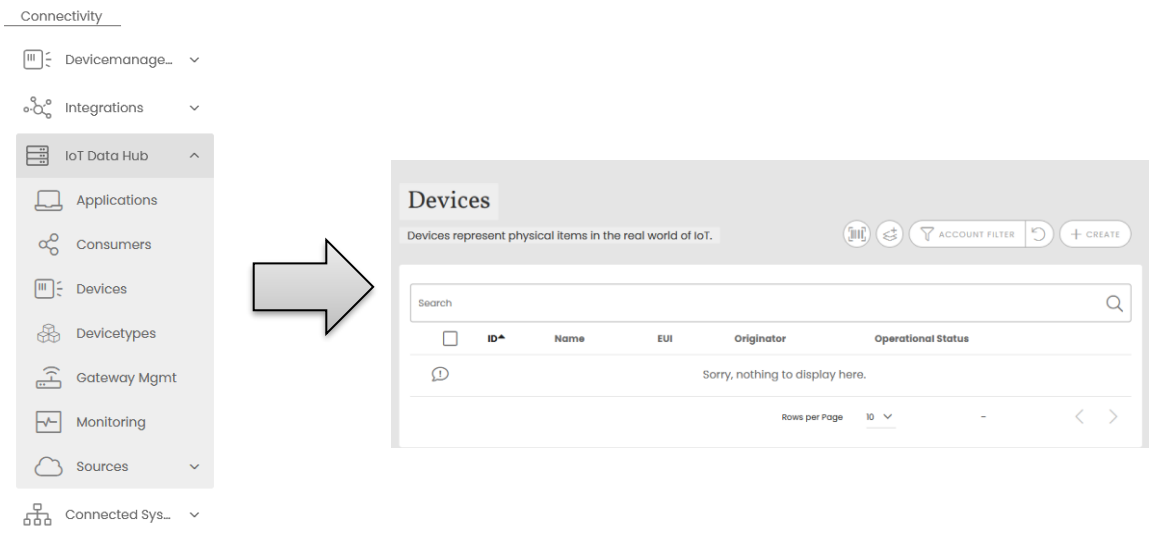


Figure 11.38: niotix: create device from menu.

11.7 Appendix G. Grafana

"Graph" visualisation has been chosen for data in the current work. It may be specified to use two y-axes (Figure 11.39). Minimum and maximum values have been set according to expected data variation, and x-axis is chosen to represent time.

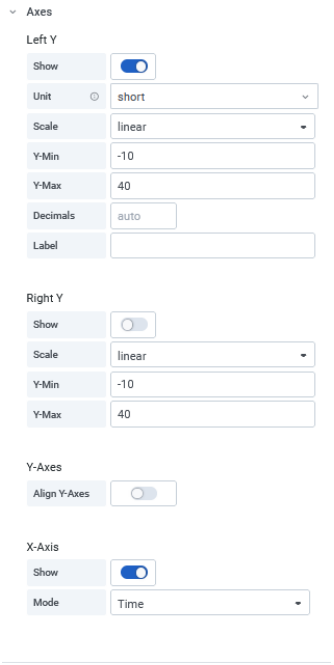


Figure 11.39: Grafana: graph settings.

To retrieve data, one must set "Query options" and specify the query parameters (Figure 11.40).

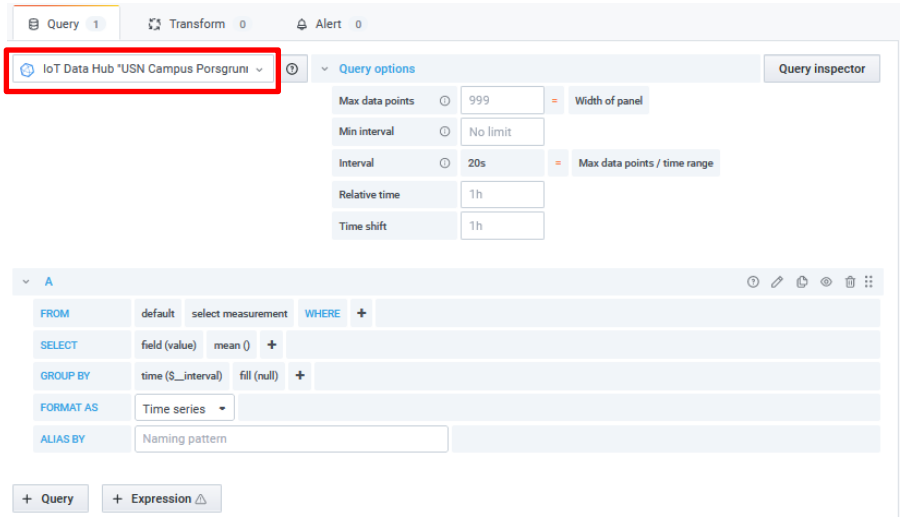
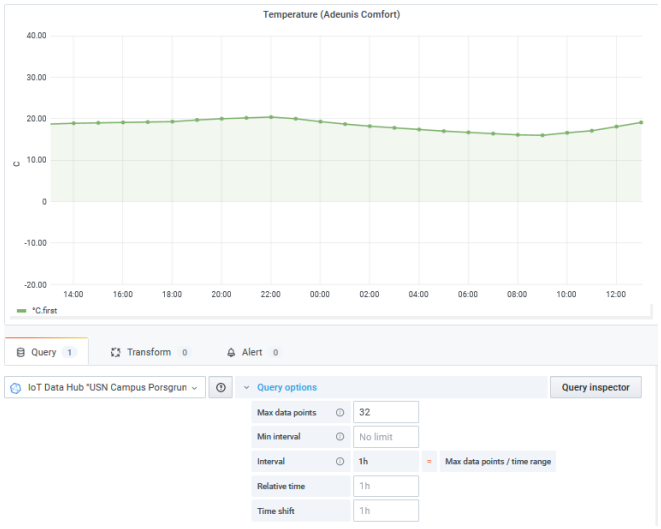


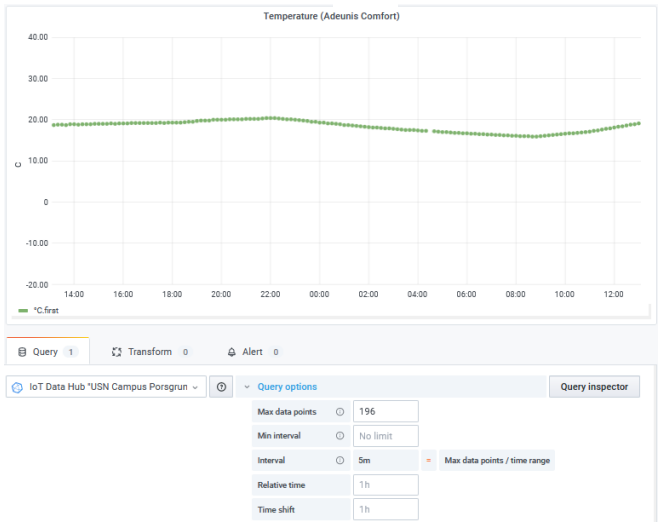
Figure 11.40: Grafana: query composition.

Query components are specific for each data source, marked red in Figure 11.40. For the chosen IoT Data Hub, one selects “default” and “select measurement” in FROM field. It is also possible to specify a device by providing its DevEUI in WHERE field.

The chosen “Max data points” will influence visualisation of the data (Figure 11.41). If there are more data points than specified in the settings, sampling time for measurement is more frequent than for visualisation, the values in the graph will be aggregated by the data source using a min, max or other function [29].



a)



b)

Figure 11.41: Grafana: query options controlled by the number of points: a) 1 hour interval, b) 5 minutes interval.

If the sampling time chosen for visualisation is larger than for the measurements, each data point is represented as a distinct line and is not connected to adjacent points. This property indicates that a user must take into account the real sampling time of a device and the data

range used to display data for. Otherwise, the graph may be not intuitive to understand. Data range is controlled by the menu in the top of the window Figure 5.25.

If the range is 24 hours, then “Max data points” 144 will allow to display measurements with 10 minutes interval. However, it is possible to set a minimum interval between adjacent points. In this case the interval will not shrink, even if the number of data points increases.

When composing a query, the available options depend on the chosen data source. If the source is «IoT Data Hub», then data can be requested using device EUI. For «States InfluxDB» one must specify digital twins and states ID instead of devices (Figure 11.42). These values are available in niotix.

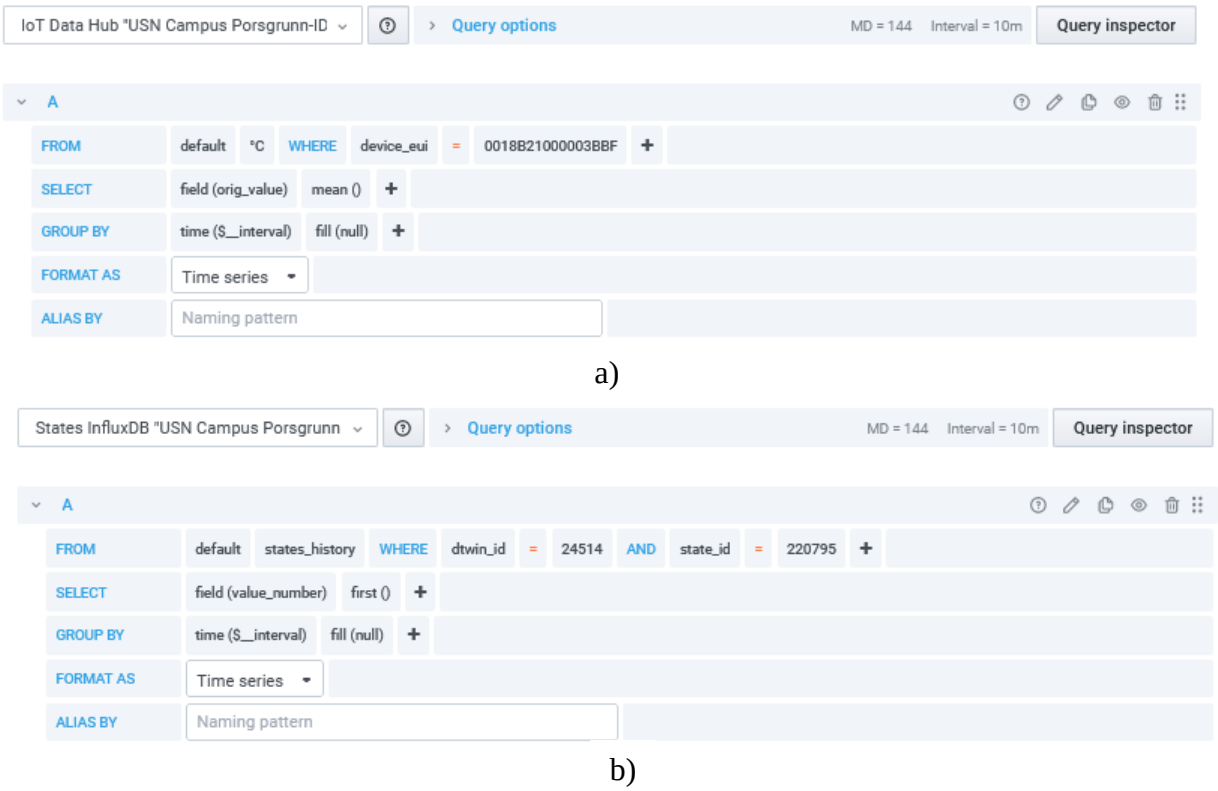


Figure 11.42: Grafana: query variation depending on the data source: a) IoT Data Hub, b) States InfluxDB.

«Query inspector» function allows to see the actual query that will be sent to the server and number of returned rows in the tab «Query» (Figure 11.44). The returned data may be also observed and saved into a .csv file (Figure 11.43).

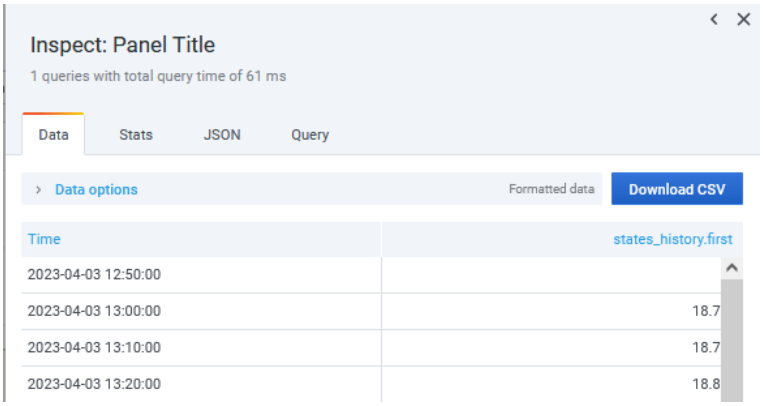


Figure 11.43: Grafana: query inspector, data section.

```
SELECT first("value_number") FROM "states_history" WHERE ("dtwin_id" = '24514' AND "state_id"
= '220795') AND time >= now() - 24h GROUP BY time(10m) fill(null)
```

Figure 11.44: Grafana: query for execution.

11.8 Appendix H. Node-RED dashboard “USN Campus in Porsgrunn MQTT”.

Figure 11.45 and Figure 11.46 represent a .json file content, they can be combined in one file and imported into Node-RED flow.


```
[{"id":"22a467f61dd2b679","type":"tab","label":"LoRaWAN dashboard MQTT","disab-
led":false,"info":"","env":[]},{"id":"b624b1292c5f6240","type":"mqtt
in","z":"22a467f61dd2b679","name":"Adeunis Comfort","to-
pic":"mqtt/things/0018B21000003BBF/uplink","qos":"0","datatype":"auto-detect","bro-
ker":"d76e3107831a9632","nl":false,"rap":true,"rh":0,"inputs":0,"x":160,"y":300,"wi-
res":[["4d33418482d73356","4bfde041f5fee7cf"]]},{"id":"8dcd0520f9b6fc0c","type":"mqtt
in","z":"22a467f61dd2b679","name":"Adeunis Temperature","to-
pic":"mqtt/things/0018B21000004540/uplink","qos":"0","datatype":"auto-detect","bro-
ker":"d76e3107831a9632","nl":false,"rap":true,"rh":0,"inputs":0,"x":180,"y":420,"wi-
res":[["8573c849c3cbd577"]]},{"id":"e21993f06e95a14b","type":"ui_chart","z":"22a467f61dd2b679","nam
e":"","group":"05906767cbdfc9f7","order":1,"width":0,"height":0,"label":"Temperature (C) prosessing
lab (Adeunis Comfort)","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"li-
near","nodata":"","dot":true,"ymin":"-20","ymax":"40","removeOlder":"1","removeOlderPoints":"","re-
moveOlderUnit":"86400","cutout":0,"useOneColor":false,"useUTC":false,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":990,"y":240,"wi-
res":[[]]},{"id":"4d33418482d73356","type":"function","z":"22a467f61dd2b679","name":"Adeunis Com-
fort: extract temperature value","func":"let temp = Number(msg.payload.signals[0].value);\nmsg.to-
pic = \"Temperature\";\nmsg.payload = temp;\nreturn msg;","outputs":1,"noerr":0,"ini-
tialize":"","finalize":"","libs":[],"x":490,"y":240,"wi-
res":[["e21993f06e95a14b"]]},{"id":"4bfde041f5fee7cf","type":"func-
tion","z":"22a467f61dd2b679","name":"Adeunis Comfort: extract humidity value","func":"let rh = Num-
ber(msg.payload.signals[1].value);\nmsg.topic = \"RH\";\nmsg.payload = rh;\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":480,"y":320,"wi-
res":[["a4f03a9e315103b2"]]},{"id":"8573c849c3cbd577","type":"func-
tion","z":"22a467f61dd2b679","name":"Convert HEX to signed numeric value","func":"let temp =
0;\n\nif (msg.payload.signals[0].value == \"null\"){\n    //Parse hex and convert to signed int.\n
temp = parseInt(msg.payload.signals[0].metadata.payload_hex.slice(4, 8), 16);\n    if ((temp &
0x8000) > 0) {\n        temp = temp - 0x10000;\n    }\n    temp = temp / 10;\n    msg.comment =
String(msg.payload.signals[0].timestamp) + \": Value in payload is null. Converted from HEX tempe-
rature is: \" + String(temp);\n}else{\n    temp = parseFloat(msg.payload.signals[0].value);\n
msg.comment = String(msg.payload.signals[0].timestamp) + \": Value is: \" +
String(temp);\n}\n\n\nmsg.topic = \"Temperature\";\nmsg.payload = temp;\n\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":470,"y":420,"wi-
res":[["ac4aaa75dcad1e33","92be71928a499312"]]},{"id":"ac4aaa75dcad1e33","type":"ui_chart","z":"22a
467f61dd2b679","name":"","group":"17741b92f7b1b3b8","order":2,"width":0,"height":0,"label":"Tempe-
rature (C) outside (Adeunis Temperature)","chartType":"line","legend":"false","xfor-
mat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"-20","ymax":"40","removeOl-
der":1,"removeOlderPoints":"","removeOlderUnit":"86400","cutout":0,"useOneCo-
lor":false,"useUTC":false,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":880,"y":420,"wi-
res":[[]]},{"id":"a4f03a9e315103b2","type":"ui_chart","z":"22a467f61dd2b679","name":"","group":"059
06767cbdfc9f7","order":3,"width":0,"height":0,"label":"Humidity (%) prosessing lab (Adeunis Com-
fort)","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","no-
data":"","dot":true,"ymin":"0","ymax":"100","removeOlder":1,"removeOlderPoints":"","removeOlder-
Unit":"86400","cutout":0,"useOneColor":false,"useUTC":false,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":920,"y":320,"wires":[[]]},
```


Figure 11.45: Node-RED: importable specification for MQTT dashboard, part 1.

```
{"id":"e0560cfa14416ac4","type":"ui_button","z":"22a467f61dd2b679","name":"","group":"17741b92f7b1b
3b8","order":2,"width":0,"height":0,"passthru":false,"label":"Reset chart","tooltip":"","co-
lor":"","bgcolor":"","className":"","icon":"","payload":"","payloadType":"str","topic":"topic","to-
picType":"msg","x":150,"y":480,"wi-
res":[["51b978b7dbd8c535"]]},{"id":"51b978b7dbd8c535","type":"func-
tion","z":"22a467f61dd2b679","name":"Reset: Temperature","func":"msg.topic = \"Tempera-
ture\";\nmsg.payload = [];\nreturn msg;","outputs":1,"noerr":0,"initialize":"","fi-
nalize":"","libs":[],"x":420,"y":480,"wi-
res":[["ac4aaa75dcad1e33"]]},{"id":"1d6c7424c1aabbc4","type":"func-
tion","z":"22a467f61dd2b679","name":"Reset: Temperature","func":"msg.topic = \"Tempera-
ture\";\nmsg.payload = [];\nreturn msg;","outputs":1,"noerr":0,"initialize":"","fi-
nalize":"","libs":[],"x":660,"y":280,"wi-
res":[["e21993f06e95a14b"]]},{"id":"2903b70a7b93e8f6","type":"func-
tion","z":"22a467f61dd2b679","name":"Reset: RH","func":"msg.topic = \"RH\";\nmsg.payload = [];\nre-
turn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":650,"y":360,"wi-
res":[["a4f03a9e315103b2"]]},{"id":"c634a318fa38ec59","type":"ui_button","z":"22a467f61dd2b679","na
me":"","group":"05906767cbdfc9f7","order":2,"width":0,"height":0,"passthru":false,"label":"Reset
chart: Temperature","tooltip":"","color":"","bgcolor":"","className":"","icon":"","pay-
load":"","payloadType":"str","topic":"topic","topicType":"msg","x":430,"y":280,"wi-
res":[["1d6c7424c1aabbc4"]]},{"id":"92be71928a499312","type":"ui_text","z":"22a467f61dd2b679","grou
p":"17741b92f7b1b3b8","order":3,"width":0,"height":0,"name":"","label":"Temperature sensor value
status:","format":"{{msg.comment}}","layout":"row-spread","className":"","x":840,"y":460,"wi-
res":[]},{"id":"3ac3b601be4bbd0e","type":"ui_button","z":"22a467f61dd2b679","name":"","group":"0590
6767cbdfc9f7","order":4,"width":0,"height":0,"passthru":false,"label":"Reset chart: Humidity","to-
oltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","payloadType":"str","to-
pic":"topic","topicType":"msg","x":420,"y":360,"wi-
res":[["2903b70a7b93e8f6"]]},{"id":"d76e3107831a9632","type":"mqtt-broker","name":"MQTT-Hive-
USN000","broker":"d6af134a78cc4a019ea25fae12ccbbfe.s2.eu.hi-
vemq.cloud","port":"8883","tls":"cc11521750f5f168","clientid":"","autoCon-
nect":true,"usetls":true,"protocolVersion":"5","keepalive":"60","cleansession":true,"birthTo-
pic":"","birthQos":"0","birthPayload":"","birthMsg":{},"closeTopic":"","closeQos":"0","closePay-
load":"","closeMsg":{},"willTopic":"","willQos":"0","willPayload":"","willMsg":{},"user-
Props":"","sessionExpiry":""},{"id":"05906767cbdfc9f7","type":"ui_group","name":"Adeunis Com-
fort","tab":"4b04438b1909d61c","order":2,"disp":true,"width":"6","collapse":false,"class-
Name":""},{"id":"17741b92f7b1b3b8","type":"ui_group","name":"Adeunis Tempera-
ture","tab":"4b04438b1909d61c","order":2,"disp":true,"width":"6","collapse":false,"class-
Name":""},{"id":"cc11521750f5f168","type":"tls-config","name":"","cert":"","key":"","ca":"","cert-
name":"","keyname":"","caname":"","servername":"","verifyservercert":false,"alpnproto-
col":""},{"id":"4b04438b1909d61c","type":"ui_tab","name":"USN Campus in Porsgrunn
MQTT","icon":"dashboard","disabled":false,"hidden":false}]
```


Figure 11.46: Node-RED: importable specification for MQTT dashboard, part 2.

Each MQTT subscriber from the workspace is listening to the same broker, but to the topic that is dedicated to a specific device (Figure 11.47).

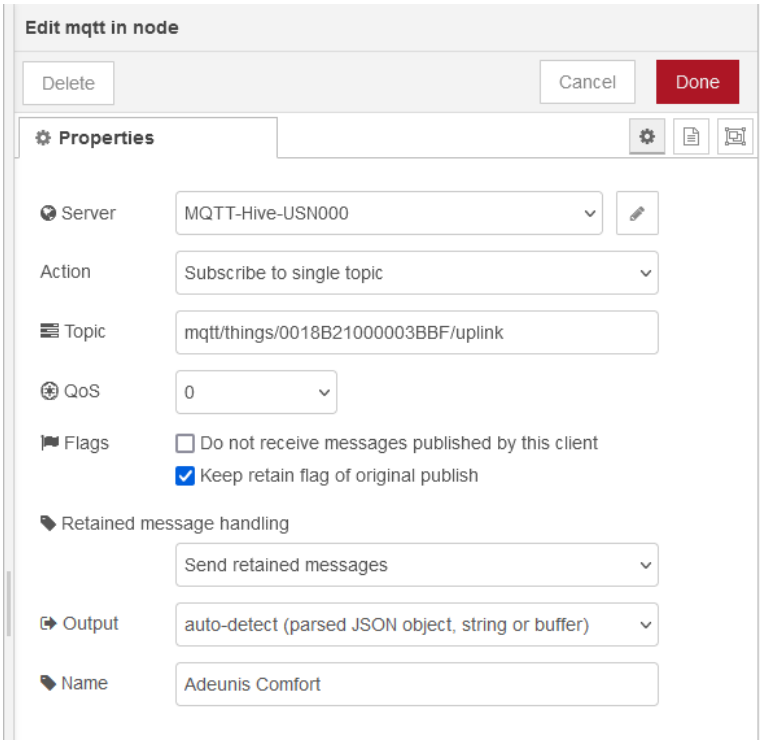


Figure 11.47: Node-RED: MQTT-in node.

The exact name of the topic is given in the settings for MQTT connection in ThingParkX. By default, it includes DevEUI, such as 0018B21000004540, of the publishing device.

Subscriber settings include URL of the broker, use of TLS specification, username and password (Figure 11.48).

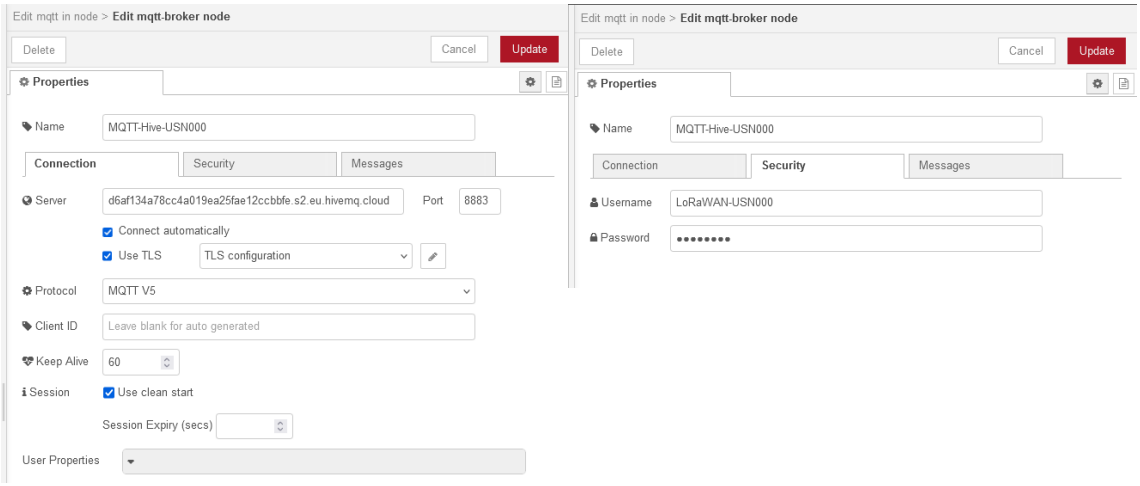


Figure 11.48: Node-RED: MQTT-in settings.

The message received by the subscriber is then forwarded in the flow as a JSON object and should be processed accordingly. Example of message received from Adeunis Comfort

sensor is shown inFigure 11.49. The part of the message that is transmitted is in payload member of the object. It may be extracted and processed as desired.

```
object
topic: "mqtt/things/0018B21000003BBF/uplink"
payload: object
    DevEUI: "0018B21000003BBF"
    signals: array[2]
        0: object
            value: "20.7"
            unit: "CELSIUS_DEGREES"
            type: "Temperature"
            timestamp: "2023-04-09T12:20:57.378+00:00"
            metadata: object
                DevEUI: "0018B21000003BBF"
                DevAddr: "FC004EA5"
                payload_hex: "4ca000cf2b"
                frequency: "868.3"
                RSSI: "-42.000000"
                SNR: "7.500000"
                SF: "7"
                latitude: "0.000000"
                longitude: "0.000000"
        1: object
            value: "43"
            unit: "PERCENTS"
            type: "Humidity"
            timestamp: "2023-04-09T12:20:57.378+00:00"
            metadata: object
                DevEUI: "0018B21000003BBF"
                DevAddr: "FC004EA5"
                payload_hex: "4ca000cf2b"
                frequency: "868.3"
                RSSI: "-42.000000"
                SNR: "7.500000"
                SF: "7"
                latitude: "0.000000"
                longitude: "0.000000"
qos: 0
retain: false
messageExpiryInterval: 259200
_msgid: "2d472249f0623844"
```

Figure 11.49: Node-RED: MQTT message sample.

The MQTT-in node is wired to two function nodes, one for extraction of temperature values and one for humidity. Function nodes contain JavaScript functions, example of extraction of temperature data as shown in Figure 11.50. Topic relates all sent values to the same time series in chart node.

```
let temp = Number(msg.payload.signals[0].value);
msg.topic = "Temperature";
msg.payload = temp;
return msg;
```

Figure 11.50: Node-RED: message processing function.

Functionality of JavaScript language allows to convert data in textual format representing numbers into numeric values. This way one may also parse “payload_hex” value from metadata from the incoming message and convert it from hex to a decimal number. Figure

11.51 shows example of parsing string “payload_hex”, finding the correct part of the string representing the measurement and converting the value to a number. This information is available in the documentation for a device. The function also considers the sign of the number, because representation of signed and unsigned integers in hex is different. If that step is not included all negative values would be represented as large positive integers. This conversion is activated if the value member of the object is null.

```
let temp = 0;

if (msg.payload.signals[0].value == "null"){
  //Parse hex and convert to signed int.
  temp = parseInt(msg.payload.signals[0].metadata.payload_hex.slice(4, 8), 16);
  if ((temp & 0x8000) > 0) {
    temp = temp - 0x10000;
  }
  temp = temp / 10;
  msg.comment = String(msg.payload.signals[0].timestamp) + ": Value in payload is null. Con-
verted from HEX temperature is: " + String(temp);
}else{
  temp = parseFloat(msg.payload.signals[0].value);
  msg.comment = String(msg.payload.signals[0].timestamp) + ": Value is: " + String(temp);
}

msg.topic = "Temperature";
msg.payload = temp;

return msg;
```

Figure 11.51: Node-RED: hex payload processing function.

If the chart must be cleared, this action may be triggered by activating button “Reset chart” wired to the chart of interest. The button activates function node that sends an empty array to the chart, for a specific topic.

```
msg.topic = "Temperature";
msg.payload = [];
return msg;
```

Figure 11.52: Node-RED: chart reset function.

Chart node settings can be left untouched. However, to make the chart representative, one must provide name, value range and time range (Figure 11.53). The node has also been assigned to a group of nodes, representing each sensor, so that the resulting dashboard is well organised and intuitive. The same group assignment is implemented to all nodes that should be displayed on a dashboard, such as buttons and text fields.

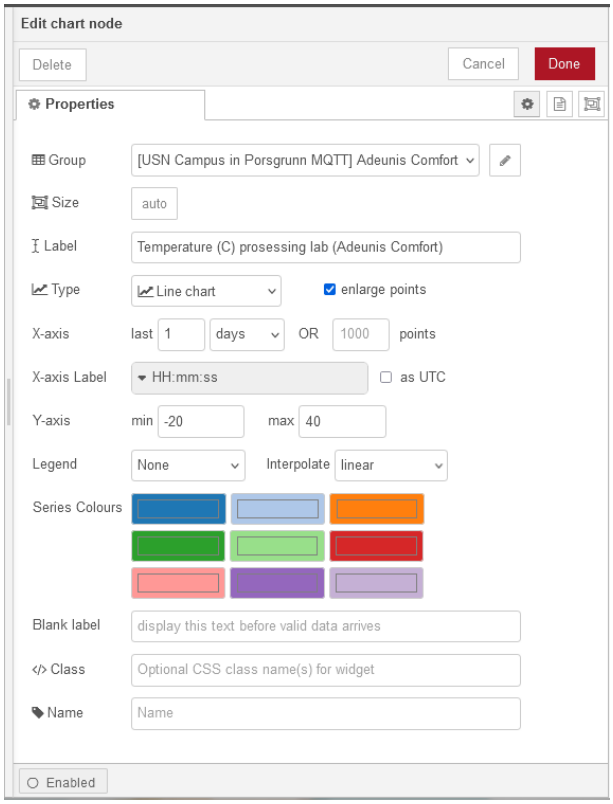


Figure 11.53: Node-RED: chart node.

Other function-nodes are shown below.

```
let rh = Number(msg.payload.signals[1].value);
msg.topic = "RH";
msg.payload = rh;
return msg;
```

Figure 11.54: Node-RED: extract humidity function.

```
msg.topic = "RH";
msg.payload = [];
return msg;
```

Figure 11.55: Node-RED: reset humidity chart function.

11.9 Appendix I. Node-RED dashboard “USN Campus in Porsgrunn HTTP Dimension Four”.

Figure 11.56, Figure 11.57 and Figure 11.58 represent a .json file content, they can be combined in one file and imported into Node-RED flow.


```
[{"id":"b660499a78c4e807","type":"tab","label":"LoRaWAN dashboard HTTP","disab-
led":false,"info":"","env":[]},{"id":"4911739177dc2f28","type":"http re-
quest","z":"b660499a78c4e807","name":"HTTP request for Adeunis Tempera-
ture","method":"POST","ret":"txt","paytoqs":"body","url":"https://iot.di-
mensionfour.io/graph","tls":"","persist":false,"proxy":"","insecureHTTPPar-
ser":false,"authType":"","senderr":false,"headers":[{"keyType":"Content-Type","keyValue":"","value-
Type":"application/json","valueValue":""},{"keyType":"Accept","keyValue":"","valueType":"applica-
tion/json","valueValue":""},{"keyType":"Accept-Encoding","keyValue":"","valueType":"gzip","valueVa-
lue":""},{"keyType":"other","keyValue":"x-tenant-id","valueType":"other","valueVa-
lue":"usn1"},{"keyType":"other","keyValue":"x-tenant-key","valueType":"other","valueVa-
lue":"5af033180135f1b5a669c714"}],"credentials":{},"x":760,"y":200,"wi-
res":[["53ed21f1fa9a82bd"]]},{"id":"53ed21f1fa9a82bd","type":"json","z":"b660499a78c4e807","name":"
","property":"payload","action":"","pretty":false,"x":870,"y":260,"wi-
res":[["1cfe2c33d3e44a45"]]},{"id":"1cfe2c33d3e44a45","type":"func-
tion","z":"b660499a78c4e807","name":"Process Adeunis Temperature","func":"function
hex_to_dec(string_in){\n    let temp = 0;\n    //Parse hex and convert to signed int.\n    temp =
parseInt(string_in.slice(4, 8), 16);\n    if ((temp & 0x8000) > 0) {\n        temp = temp -
0x10000;\n    }\n    return temp;\n}\nlet to_read = msg.to_read;\n\nlet data = [];\nlet
data_entries = [];\nfor (let o = 0; o < to_read; o++){\n    //data_entres.push({ \"x\": msg.pay-
load.data.signals.edges[o].node.timestamp, \"y\":Number(msg.payload.data.signals.edges[o].node.me-
tadata.SF)});\n    if (msg.payload.data.signals.edges[o].node.data.numericValue == null){\n
data_entries.push({ \"x\": msg.payload.data.signals.edges[o].node.timestamp, \"y\":
hex_to_dec(msg.payload.data.signals.edges[o].node.metadata.payload_hex) / 10 });\n    }else{\n
data_entries.push({ \"x\": msg.payload.data.signals.edges[o].node.timestamp, \"y\": Number(msg.pay-
load.data.signals.edges[o].node.data.numericValue) });\n
}\n}\n\ndata.push(data_entries);\n\nmsg.payload = [{\n    \"series\": [\"Temperature\"],\n
\"data\": data,\n    \"labels\": [\"\"]\n}];\nreturn msg;","outputs":1,"noerr":0,"ini-
tialize":"","finalize":"","libs":[],"x":1090,"y":260,"wi-
res":[["a5c547ddee1a0a2b"]]},{"id":"a5c547ddee1a0a2b","type":"ui_chart","z":"b660499a78c4e807","nam
e":"","group":"14c00f4d49d1f96b","order":4,"width":0,"height":0,"label":"Temperature (C) outside
(Adeunis Temperature)","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"li-
near","nodata":"","dot":true,"ymin":"-20","ymax":"40","removeOlder":1,"removeOlderPoints":"","re-
moveOlderUnit":"86400","cutout":0,"useOneColor":false,"useUTC":true,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":1060,"y":320,"wi-
res":[[]]},{"id":"ce441022384c4c10","type":"function","z":"b660499a78c4e807","name":"Reset func-
tion","func":"msg.payload = [];\nreturn msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":760,"y":320,"wires":[["a5c547ddee1a0a2b"]]},{"id":"1f65289c96c79368","type":"
ui_button","z":"b660499a78c4e807","name":"","group":"14c00f4d49d1f96b","or-
der":5,"width":0,"height":0,"passthru":false,"label":"Reset
chart","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","top-
icType":"msg","x":570,"y":320,"wires":[["ce441022384c4c10"]]},{"id":"d69fffb88ddb1c6f","type":"http
request","z":"b660499a78c4e807","name":"HTTP request for Temperature Adeunis Com-
fort","method":"POST","ret":"txt","paytoqs":"ignore","url":"https://iot.dimension-
four.io/graph","tls":"","persist":false,"proxy":"","insecureHTTP-
Parser":false,"authType":"","senderr":false,"headers":[{"keyType":"Content-Type","key-
Value":"","valueType":"application/json","valueValue":""},{"keyType":"Accept","keyValue":"","valu-
eType":"application/json","valueValue":""},{"keyType":"Accept-Encoding","keyValue":"","valu-
eType":"gzip","valueValue":""},{"keyType":"other","keyValue":"x-tenant-id","valu-
eType":"other","valueValue":"usn1"},{"keyType":"other","keyValue":"x-tenant-key","valu-
eType":"other","value-
Value":"5af033180135f1b5a669c714"}],"x":770,"y":400,"wires":[["f1ac679f7640c7a4"]]},{"id":"f1ac679f
7640c7a4","type":"json","z":"b660499a78c4e807","name":"","property":"payload","ac-
tion":"","pretty":false,"x":650,"y":460,"wires":[["47dd61541f38bcea"]]},
```


Figure 11.56: Node-RED: importable specification for Dimension Four dashboard, part 1.

```
{"id":"04a2077c0998d50d","type":"ui_but-
ton","z":"b660499a78c4e807","name":"","group":"045549438407cc6d","or-
der":3,"width":0,"height":0,"passthru":false,"label":"Reset chart: Tempera-
ture","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","top-
icType":"msg","x":190,"y":500,"wires":[["6c63bceb39a4acb1"]]},{"id":"03e90a374045ca75","type":"ui_b
utton","z":"b660499a78c4e807","name":"","group":"045549438407cc6d","or-
der":6,"width":0,"height":0,"passthru":false,"label":"Reset chart: Humid-
ity","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","top-
icType":"msg","x":180,"y":540,"wires":[["cd9272fb311c4736"]]},{"id":"6c63bceb39a4acb1","type":"func
tion","z":"b660499a78c4e807","name":"Reset function","func":"msg.payload = [];\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":720,"y":500,"wi-
res":[["d05c2ffaf7d9db85"]]},{"id":"cd9272fb311c4736","type":"func-
tion","z":"b660499a78c4e807","name":"Reset function","func":"msg.payload = [];\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":720,"y":540,"wi-
res":[["e5a5db0da5160d55"]]},{"id":"d05c2ffaf7d9db85","type":"ui_chart","z":"b660499a78c4e807","nam
e":"","group":"045549438407cc6d","order":2,"width":0,"height":0,"label":"Temperature (C) prosessing
lab (Adeunis Comfort)","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"li-
near","nodata":"","dot":true,"ymin":"-20","ymax":"40","removeOlder":1,"removeOlderPoints":"","re-
moveOlderUnit":"86400","cutout":0,"useOneColor":false,"useUTC":true,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":1190,"y":500,"wi-
res":[[]]},{"id":"e5a5db0da5160d55","type":"ui_chart","z":"b660499a78c4e807","name":"","group":"045
549438407cc6d","order":5,"width":0,"height":0,"label":"Humidity (%) prosessing lab (Adeunis Com-
fort)","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","no-
data":"","dot":true,"ymin":"0","ymax":"100","removeOlder":1,"removeOlderPoints":"","removeOlder-
Unit":"86400","cutout":0,"useOneColor":false,"useUTC":true,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":1180,"y":540,"wi-
res":[[]]},{"id":"47dd61541f38bcea","type":"function","z":"b660499a78c4e807","name":"Process Tempe-
rature Adeunis Comfort","func":"function hex_to_dec(string_in) {\n    let temp = 0;\n    //Parse
hex and convert to signed int.\n    temp = parseInt(string_in.slice(4, 8), 16);\n    if ((temp &
0x8000) > 0) {\n        temp = temp - 0x10000;\n    }\n    return temp;\n}\nlet to_read =
msg.to_read;\n\nlet data_temp = [];\nlet data_entries_temp = [];\nlet x_val = null;\nlet y_val =
null;\nfor (let o = 0; o < to_read; o++) {\n    \n    x_val = msg.payload.data.signals.ed-
ges[o].node.timestamp;\n\n    if (msg.payload.data.signals.edges[o].node.data.numericValue == null)
{\n        y_val = hex_to_dec(msg.payload.data.signals.edges[o].node.metadata.payload_hex) / 10;\n
} else {\n        y_val = Number(msg.payload.data.signals.edges[o].node.data.numericValue);\n
}\n\n    data_entries_temp.push({ \"x\": x_val, \"y\": y_val
});\n}\n\ndata_temp.push(data_entries_temp);\nmsg.payload = [{\n                    \"series\":
[\"Temperature\"],\n                    \"data\": data_temp,\n                    \"labels\":
[\"\"]\n                }];\n\nreturn msg;","outputs":1,"noerr":0,"initialize":"","fi-
nalize":"","libs":[],"x":880,"y":460,"wi-
res":[["d05c2ffaf7d9db85"]]},{"id":"39fc8ddcf0ddccd2","type":"func-
tion","z":"b660499a78c4e807","name":"Prepare GraphQL request: Adeunis Temperature.","func":"let num
= msg.payload;\nif(num > 100){\n    num = 100;\n}else if(num < 0){\n    num = 0;\n}\n\nmsg.payload
= {\n    \"query\":\"query LATEST_SIGNALS {    signals(where: {        pointId: { _EQ:
\\\"6362c7defc18de3434dd2ab3\\\"}}     paginate: { last: \" + String(num) + \" }   ){     edges {
node {         id         timestamp         location{            lat            lon}         type
unit         pointId         data {           numericValue           rawValue         }
metadata       }     }   } }\"\n    };\nmsg.to_read = num;\nreturn msg;","outputs":1,"noerr":0,"in-
itialize":"","final-
ize":"","libs":[],"x":380,"y":200,"wires":[["4911739177dc2f28"]]},{"id":"32cce1a39f4ccad6","type":"
ui_text_input","z":"b660499a78c4e807","name":"","label":"Number of values to read (0 <= value <=
100)","tooltip":"100","group":"14c00f4d49d1f96b","or-
der":1,"width":0,"height":0,"passthru":false,"mode":"number","de-
lay":"0","topic":"to_read","sendOnBlur":false,"className":"","top-
icType":"str","x":270,"y":260,"wires":[["39fc8ddcf0ddccd2"]]},{"id":"7d32da130b13677c","type":"ui_t
ext_input","z":"b660499a78c4e807","name":"","label":"Temperature: number of values to read (0 <=
value <= 100)","tooltip":"","group":"045549438407cc6d","or-
der":1,"width":0,"height":0,"passthru":false,"mode":"number","de-
lay":"0","topic":"to_read","sendOnBlur":false,"className":"","top-
icType":"str","x":300,"y":460,"wires":[["8ab76c98b66254ba"]]},
```


Figure 11.57: Node-RED: importable specification for Dimension Four dashboard, part 2.

```
{"id":"8ab76c98b66254ba","type":"function","z":"b660499a78c4e807","name":"Prepare GraphQL request:
Temperatrue Adeunis Comfort.","func":"let num = msg.payload;\n\nif (num > 100) {\n    num = 100;\n}
else if (num < 0) {\n    num = 0;\n}\n\nmsg.payload = { \n    \"query\": \"query LATEST_SIGNALS {
signals(where: { pointId: { _EQ : \\\"6362ccc1fc18de50dedd2c78\\\"}, type: { _EQ: \\\"Tempera-
ture\\\" }} paginate: { last: \" + String(num) + \" }   ){     edges {       node {         id
timestamp         location{            lat            lon}         type         unit         poin-
tId         data {           numericValue           rawValue         }         metadata       }
}   } }\"\n};\n\nmsg.to_read = num;\nreturn msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":330,"y":400,"wires":[["d69fffb88ddb1c6f"]]},{"id":"5737398047eee345","type":"
ui_text_input","z":"b660499a78c4e807","name":"","label":"Humidity: number of values to read  (0 <=
value <= 100)","tooltip":"","group":"045549438407cc6d","or-
der":4,"width":0,"height":0,"passthru":false,"mode":"number","de-
lay":"0","topic":"to_read","sendOnBlur":false,"className":"","top-
icType":"msg","x":290,"y":580,"wires":[["7daef050dbdc7cae"]]},{"id":"7daef050dbdc7cae","type":"func
tion","z":"b660499a78c4e807","name":"Prepare GraphQL request: Humidity Adeunis Com-
fort.","func":"let num = msg.payload;\n\nif (num > 100) {\n    num = 100;\n} else if (num < 0) {\n
num = 0;\n}\n\nmsg.payload = {\n    \"query\": \"query LATEST_SIGNALS { signals(where: { pointId: {
_EQ : \\\"6362ccc1fc18de50dedd2c78\\\"}, type: { _EQ: \\\"Humidity\\\" }} paginate: { last: \" +
String(num) + \" }   ){     edges {       node {         id         timestamp         location{
lat            lon}         type         unit         pointId         data {           numericValue
rawValue         }         metadata       }     }   } }\"\n};\n\nmsg.to_read = num;\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":320,"y":640,"wi-
res":[["f830ee9ae867a16e"]]},{"id":"91201ba942d868d9","type":"func-
tion","z":"b660499a78c4e807","name":"Process Humidity Adeunis Comfort","func":"let to_read =
msg.to_read;\n\nlet data_hum = [];\nlet data_entries_hum = [];\nlet x_val = null;\nlet y_val =
null;\nfor (let o = 0; o < to_read; o++){\n    x_val = msg.payload.data.signals.edges[o].node.ti-
mestamp;\n    y_val = Number(msg.payload.data.signals.edges[o].node.data.numericValue);\n\n
data_entries_hum.push({ \"x\": x_val, \"y\": y_val
});\n}\n\ndata_hum.push(data_entries_hum);\nmsg.payload = [{\n        \"series\": [\"Humidity\"],\n
\"data\": data_hum,\n        \"labels\": [\"\"]\n    }];\n\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":860,"y":580,"wi-
res":[["e5a5db0da5160d55"]]},{"id":"eb365710aefaa215","type":"json","z":"b660499a78c4e807","name":"
","property":"payload","action":"","pretty":false,"x":650,"y":580,"wi-
res":[["91201ba942d868d9"]]},{"id":"f830ee9ae867a16e","type":"http re-
quest","z":"b660499a78c4e807","name":"HTTP request for Humidity Adeunis Com-
fort","method":"POST","ret":"txt","paytoqs":"ignore","url":"https://iot.di-
mensionfour.io/graph","tls":"","persist":false,"proxy":"","insecureHTTPPar-
ser":false,"authType":"","senderr":false,"headers":[{"keyType":"Accept","keyValue":"","value-
Type":"application/json","valueValue":""},{"keyType":"Content-Type","keyValue":"","value-
Type":"application/json","valueValue":""},{"keyType":"other","keyValue":"x-tenant-id","value-
Type":"other","valueValue":"usn1"},{"keyType":"other","keyValue":"x-tenant-key","valueType":"ot-
her","valueValue":"5af033180135f1b5a669c714"},{"keyType":"Accept-Encoding","keyValue":"","value-
Type":"gzip","valueValue":""}],"x":740,"y":640,"wi-
res":[["eb365710aefaa215"]]},{"id":"14c00f4d49d1f96b","type":"ui_group","name":"Adeunis Tempera-
ture","tab":"0937d99eb34cc44e","order":2,"disp":true,"width":"6","collapse":false,"class-
Name":""},{"id":"045549438407cc6d","type":"ui_group","name":"Adeunis Com-
fort","tab":"0937d99eb34cc44e","order":1,"disp":true,"width":"6","collapse":false,"class-
Name":""},{"id":"0937d99eb34cc44e","type":"ui_tab","name":"USN Campus in Porsgrunn HTTP Dimension
Four","icon":"dashboard","disabled":false,"hidden":false}]
```


Figure 11.58: Node-RED: importable specification for Dimension Four dashboard, part 3.

On the dashboard user triggers data request by entering a desired number of values to request, between zero and 100, into a text field. The limit is introduced because the maximum number of values that Dimension Four API may return is 100.

In the next step a GraphQL query is prepared and HTTP request node is activated. Output of the last node is converted into JSON object for convenience using JSON-node. After that procedure is similar as in previous sub-chapter. The values of interest are extracted and forwarded to chart node in the proper format.

Query for a GraphQL is shown in Figure 11.59. It is not similar to SQL, and, therefore, one must pay attention to use correct conditions.

```
query LATEST_SIGNALS {
  signals(
    where:{pointId:{_EQ:"6362ccc1fc18de50dedd2c78"},
    type: { _EQ: "Temperature" }}
    paginate: { last:10 }
  ){
    edges {
      node {
        id
        timestamp
        location{
           lat
           lon}
        type
        unit
        pointId
        data {
          numericValue
          rawValue
        }
        metadata
      }
    }
  }
}
```

Figure 11.59: Node-RED: Dimension Four: GraphQL query.

To return multiple rows, the query contains a pagination property, where user may specify how many rows it is required. The number must not exceed 100.

Return from such request has also specific format (Figure 11.60). The root object is “data”, it includes object “signals”, that was requested in the query. Property “edges“ represents an array of JSON objects. Each object can be considered as a separate row of a table.


```
{
  "data": {
    "signals": {
      "edges": [
        {
          "node": {
            "id": "11111111",
            "timestamp": "2021-11-17T14:30:21.000000000+0000",
            "location": null,
            "type": "air temperature",
            "unit": "CELSIUS_DEGREES",
            "pointId": "22222222222222",
            "data": {
              "numericValue": 22.200000762939453,
              "rawValue": "22.2"
            }
          }
        }
      ]
    }
  }
}
```


Figure 11.60: NodeRED: Dimension Four: response structure.

The query is constructed in function node and sent further as a payload (Figure 11.61). The only modification of the payload that is needed is adding of the number of values to request.

The id of the point to request values from, is hard-coded into the query for each of the devices. Therefore, there is a separate function node for each device. The number of values is also written into a custom element of the message, msg.to_read, to be used in the following nodes when necessary.

```
let num = msg.payload;
if(num > 100){
    num = 100;
}else if(num < 0){
    num = 0;
}

msg.payload = {
    "query":"query LATEST_SIGNALS {    signals(where: {        pointId: { _EQ:
\"6362c7defc18de3434dd2ab3\"}}     paginate: { last: " + String(num) + " }   ){     edges
{       node {          id          timestamp          loca-
tion{            lat            lon}         type         unit         pointId          data
{           numericValue           rawValue         }         metadata       }     }   } }"
    };
msg.to_read = num;
return msg;
```

Figure 11.61: NodeRED: Payload preparation for Adeunis temperature sensor.

To obtain data from Dimension Four one must send the query with a HTTP request to iot.dimensionfour.io/graph API using POST method (Figure 11.62). The request includes custom headers: “x-tenant-id” and “x-tenant-key”. Values for the headers can be obtained from tenant overview page on Dimension Four portal. It is important to note that while the portal uses secure HTTP protocol, “https”, connection is not secured with SSL/TLS, therefore, this box must not be marked.

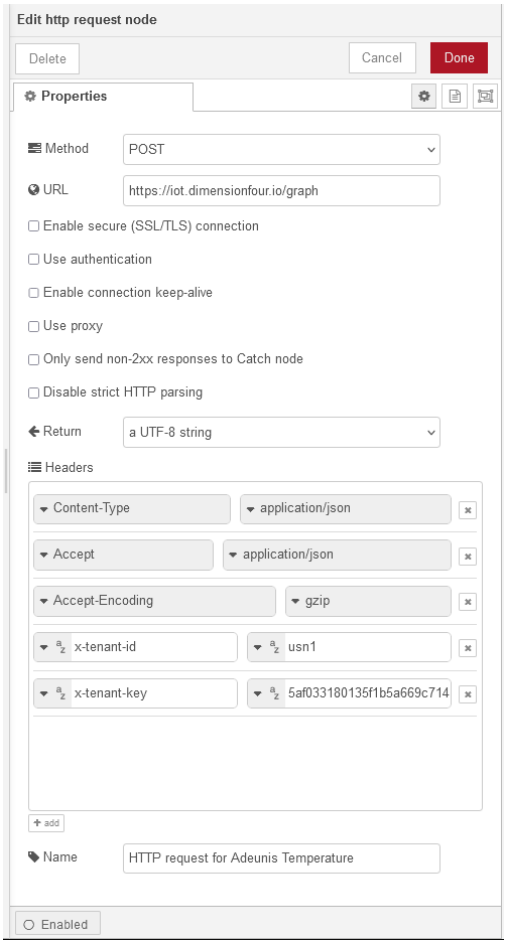


Figure 11.62: HTTP request settings.

Response from the server is converted to JSON object and parsed in subsequent function node (Figure 11.63) according to the example shown in Figure 11.60. In a similar way as in previous flow, the message is examined for validity of the measured value. If the value is null, then “payload_hex” string is sent to the function for value conversion. The values are forwarded to chart node in a slightly different manner, this allows to plot several time series in one chart if necessary.

```
function hex_to_dec(string_in){
    let temp = 0;
    //Parse hex and convert to signed int.
    temp = parseInt(string_in.slice(4, 8), 16);
    if ((temp & 0x8000) > 0) {
        temp = temp - 0x10000;
    }
    return temp;
}
let to_read = msg.to_read;

let data = [];
let data_entries = [];
for (let o = 0; o < to_read; o++){
    if (msg.payload.data.signals.edges[o].node.data.numericValue == null){
        data_entries.push({ "x": msg.payload.data.signals.edges[o].node.timestamp, "y":
hex_to_dec(msg.payload.data.signals.edges[o].node.metadata.payload_hex) / 10 });
    }else{
        data_entries.push({ "x": msg.payload.data.signals.edges[o].node.timestamp, "y": Num-
ber(msg.payload.data.signals.edges[o].node.data.numericValue) });
    }
}

data.push(data_entries);

msg.payload = [{
    "series": ["Temperature"],
    "data": data,
    "labels": [""]
}];
return msg;
```

Figure 11.63: NodeRED: Temperature sensor response processing.

```
let num = msg.payload;

if (num > 100) {
    num = 100;
} else if (num < 0) {
    num = 0;
}

msg.payload = {
    "query": "query LATEST_SIGNALS { signals(where: { pointId: { _EQ :
\"6362ccc1fc18de50dedd2c78\"}, type: { _EQ: \"Temperature\" }} paginate: { last: " +
String(num) + " }   ){     edges {       node {         id         timestamp         loca-
tion{            lat            lon}         type         unit         pointId         data
{           numericValue           rawValue         }         metadata       }     }   } }"
};

msg.to_read = num;
return msg;
```

Figure 11.64: NodeRED: Payload preparation for obtaining temperatrue from Adeunis Comfort sensor.

```
unction hex_to_dec(string_in) {
    let temp = 0;
    //Parse hex and convert to signed int.
    temp = parseInt(string_in.slice(4, 8), 16);
    if ((temp & 0x8000) > 0) {
        temp = temp - 0x10000;
    }
    return temp;
}
let to_read = msg.to_read;

let data_temp = [];
let data_entries_temp = [];
let x_val = null;
let y_val = null;
for (let o = 0; o < to_read; o++) {

    x_val = msg.payload.data.signals.edges[o].node.timestamp;

    if (msg.payload.data.signals.edges[o].node.data.numericValue == null) {
        y_val = hex_to_dec(msg.payload.data.signals.edges[o].node.metadata.payload_hex) / 10;
    } else {
        y_val = Number(msg.payload.data.signals.edges[o].node.data.numericValue);
    }

    data_entries_temp.push({ "x": x_val, "y": y_val });
}

data_temp.push(data_entries_temp);
msg.payload = [{
                    "series": ["Temperature"],
                    "data": data_temp,
                    "labels": [""]
                }];

return msg;
```

Figure 11.65: NodeRED: Adeunis Comfort sensor response processing, with respect to temperature.

```
let num = msg.payload;

if (num > 100) {
    num = 100;
} else if (num < 0) {
    num = 0;
}

msg.payload = {
    "query": "query LATEST_SIGNALS { signals(where: { pointId: { _EQ :
\"6362ccc1fc18de50dedd2c78\"}, type: { _EQ: \"Humidity\" }} paginate: { last: " + String(num)
+ " }   ){     edges {       node {         id         timestamp         loca-
tion{            lat            lon}         type         unit         pointId         data
{           numericValue           rawValue         }         metadata       }     }   } }"
};

msg.to_read = num;
return msg;
```

Figure 11.66: NodeRED: Payload preparation for obtaining humidity from Adeunis Comfort sensor.

```
let to_read = msg.to_read;

let data_hum = [];
let data_entries_hum = [];
let x_val = null;
let y_val = null;
for (let o = 0; o < to_read; o++){
    x_val = msg.payload.data.signals.edges[o].node.timestamp;
    y_val = Number(msg.payload.data.signals.edges[o].node.data.numericValue);

    data_entries_hum.push({ "x": x_val, "y": y_val });
}

data_hum.push(data_entries_hum);
msg.payload = [{
        "series": ["Humidity"],
        "data": data_hum,
        "labels": [""]
    }];

return msg;
```

Figure 11.67: NodeRED: Adeunis Comfort sensor response processing, with respect to humidity.

11.10 Appendix J. Node-RED dashboard “USN Campus in Porsgrunn Niotix”.

Figure 11.68, Figure 11.69, Figure 11.70, and Figure 11.71 represent a .json file content, they can be combined in one file and imported into Node-RED flow.


```
[{"id":"253389855d742e07","type":"tab","label":"LoRaWAN dashboard Niotix","disab-
led":false,"info":"","env":[]},{"id":"b7d570f8808e41cc","type":"http re-
quest","z":"253389855d742e07","name":"","method":"GET","ret":"txt","pay-
toqs":"body","url":"","tls":"","persist":false,"proxy":"","insecureHTTPPar-
ser":false,"authType":"","senderr":false,"headers":[{"keyType":"other","keyValue":"x-api-key","va-
lueType":"msg","valueValue":"apiKey"},{"keyType":"Accept","keyValue":"","valueType":"applica-
tion/json","valueValue":""}],"x":850,"y":240,"wi-
res":[["5a942fb9808c3cec"]]},{"id":"3021e9e1005ca1af","type":"func-
tion","z":"253389855d742e07","name":"Outdoor reader key","func":"msg.apiKey = \"82D714C9-DB85-45CA-
8A95-27B824A125F7\";\n//msg.payload = \"id=\"+msg.payload;\nmsg.url = msg.url + msg.payload;\nre-
turn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":620,"y":260,"wi-
res":[["b7d570f8808e41cc"]]},{"id":"516210d15e16fa35","type":"ui_button","z":"253389855d742e07","na
me":"","group":"850d19e1e65e5ea1","order":1,"width":0,"height":0,"passthru":true,"label":"Read Tem-
peratrue (Adeunis Temperature)","tooltip":"","color":"","bgcolor":"","className":"","icon":"","pay-
load":"220797","payloadType":"num","topic":"topic","topicType":"msg","x":700,"y":60,"wi-
res":[["3f63fc475ee044d2"]]},{"id":"3f63fc475ee044d2","type":"func-
tion","z":"253389855d742e07","name":"URL for states","func":"msg.url =
\"https://xapi.niota.io/xapi/v1/states/\";\nreturn msg;","outputs":1,"noerr":0,"initialize":"","fi-
nalize":"","libs":[],"x":600,"y":200,"wi-
res":[["3021e9e1005ca1af","e15a0b71380ce5e0"]]},{"id":"5a942fb9808c3cec","type":"json","z":"2533898
55d742e07","name":"","property":"payload","action":"","pretty":false,"x":850,"y":280,"wi-
res":[["79ab19f2b24191a5"]]},{"id":"bb12dc3204fd6d1d","type":"func-
tion","z":"253389855d742e07","name":"Process Adeunis Temperature","func":"function int_unsig-
ned_to_signed(val){\n    if (val && 0x8000) {\n        val -= 0x10000;\n        val/=10;\n    }\n
return val;\n}\n\nlet prev_data = flow.get(\"temp_temp\");\nprev_data = JSON.parse(JSON.string-
ify(prev_data));\n\nif (!Array.isArray(prev_data)) {\n    prev_data = [prev_data];\n    msg.com-
ment200 = \"if-test: !Array.isArray(prev_data)\";\n} else {\n    msg.comment200 = \"if-test:
Array.isArray(prev_data)\";\n}\nif (!prev_data[0]) {\n    prev_data[0] = [];\n    msg.comment201 =
\"if-test: !prev_data[0]\";\n} else {\n    msg.comment201 = \"if-test: prev_data[0]\";\n}\nif
(!Array.isArray(prev_data[0])) {\n    prev_data[0] = [prev_data[0]];\n    msg.comment202 = \"if-
test: !Array.isArray(prev_data[0])\";\n} else {\n    msg.comment202 = \"if-test: Array.isAr-
ray(prev_data[0])\";\n}\nif (prev_data[0].length == 0) {\n    prev_data[0] = [{ \"series\": [\"Tem-
perature\"], \"data\": [], \"labels\": [\"\"] }];\n    msg.comment203 = \"if-test: prev_data.length
== 0\";\n} else {\n    msg.comment203 = \"if-test: prev_data.length != 0\";\n}\nlet temp = Num-
ber(msg.payload.data.binding.value);\n\nif (temp*10 > 2000){\n    temp = int_unsigned_to_sig-
ned(temp*10);\n}\n\nif (!prev_data[0][0].data) {\n    prev_data[0][0].data = [];\n    msg.com-
ment204 = \"if-test: !prev_data[0][0].data\";\n} else {\n    msg.comment204 = \"if-test:
prev_data[0][0].data\";\n}\nmsg.prev_time = new Date(msg.payload.data.bin-
ding.time);\nmsg.prev_time_s = new Date(msg.payload.data.binding.time).getTime();\n\nlet time_flag
= 0;\nif (prev_data[0][0].data.length != 0) {\n    if (prev_data[0][0].data.at(-1)[0][\"x\"] != new
Date(msg.payload.data.binding.time).getTime()) {\n        msg.comment205 = \"if-test: new time: \"
+ msg.payload.data.binding.time;\n        time_flag = 1;\n    } else {\n        msg.comment205 =
\"if-test: same time: \" + msg.payload.data.binding.time;\n    }\n} else {\n    msg.comment205 =
\"if-test: prev_data[0][0].data.length == 0\";\n    time_flag = 1;\n    prev_data[0][0].data[0] =
[];\n}\nif (time_flag > 0) {\n    prev_data[0][0].data[0].push({ \"x\": new Date(msg.pay-
load.data.binding.time).getTime(), \"y\": temp });\n}\n\nmsg.payload = [{\n    \"series\": [\"Tem-
perature\"],\n    \"data\": prev_data[0][0].data,\n    \"labels\": [\"\"]\n}];\n\nmsg.prev_data =
prev_data;\n\nreturn msg;","outputs":1,"noerr":0,"initialize":"","fi-
nalize":"","libs":[],"x":1230,"y":240,"wires":[["8dcc82f5c9c6029e"]]},
```


Figure 11.68: Node-RED: importable specification for niotix dashboard, part 1.

```
{"id":"8dcc82f5c9c6029e","type":"ui_chart","z":"253389855d742e07","name":"","group":"850d19e1e65e5e
a1","order":3,"width":0,"height":0,"label":"Temperature (C) outside (Adeunis Temperature)","chart-
Type":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","no-
data":"","dot":true,"ymin":"-20","ymax":"40","removeOlder":1,"removeOlderPoints":"","removeOlder-
Unit":"86400","cutout":0,"useOneColor":false,"useUTC":false,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":1280,"y":200,"wi-
res":[["69f77023af128952"]]},{"id":"5a5eb890a8f15036","type":"ui_button","z":"253389855d742e07","na
me":"","group":"71d388d9fa9b88ac","order":1,"width":0,"height":0,"passthru":true,"label":"Read Tem-
peratrue (Adeunis Comfort)","tooltip":"","color":"","bgcolor":"","className":"","icon":"","pay-
load":"220795","payloadType":"str","topic":"topic","topicType":"msg","x":670,"y":100,"wi-
res":[["3f63fc475ee044d2"]]},{"id":"e15a0b71380ce5e0","type":"func-
tion","z":"253389855d742e07","name":"Processing hall reader key","func":"msg.apiKey = \"2A463053-
6194-40C8-82AE-E6988A81FAC6\";\n//msg.payload = \"id=\"+msg.payload;\nmsg.url = msg.url + msg.pay-
load;\nreturn msg;","outputs":1,"noerr":0,"initialize":"","fi-
nalize":"","libs":[],"x":640,"y":300,"wi-
res":[["b7d570f8808e41cc"]]},{"id":"79ab19f2b24191a5","type":"switch","z":"253389855d742e07","name"
:"","property":"payload.data.id","propertyType":"msg","ru-
les":[{"t":"eq","v":"220797","vt":"num"},{"t":"eq","v":"220795","vt":"num"},{"t":"eq","v":"220796",
"vt":"num"}],"checkall":"true","repair":false,"outputs":3,"x":1010,"y":260,"wi-
res":[["bb12dc3204fd6d1d","f06ecdcafcc00fa0"],["7764408494712810","1d4044e693040b41"],["5f7a2eda825
4d222","c4d1d2f61dc060b8"]]},{"id":"7764408494712810","type":"func-
tion","z":"253389855d742e07","name":"Process Temperature (Adeunis Comfort)","func":"let prev_data =
flow.get(\"comf_temp\");\nprev_data = JSON.parse(JSON.stringify(prev_data));\n\nif (!Array.isAr-
ray(prev_data)) {\n    prev_data = [prev_data];\n    msg.comment200 = \"if-test: !Array.isAr-
ray(prev_data)\";\n} else {\n    msg.comment200 = \"if-test: Array.isArray(prev_data)\";\n}\nif
(!prev_data[0]) {\n    prev_data[0] = [];\n    msg.comment201 = \"if-test: !prev_data[0]\";\n} else
{\n    msg.comment201 = \"if-test: prev_data[0]\";\n}\nif (!Array.isArray(prev_data[0])) {\n
prev_data[0] = [prev_data[0]];\n    msg.comment202 = \"if-test: !Array.isArray(prev_data[0])\";\n}
else {\n    msg.comment202 = \"if-test: Array.isArray(prev_data[0])\";\n}\nif (prev_data[0].length
== 0) {\n    prev_data[0] = [{ \"series\": [\"Temperature\"], \"data\": [], \"labels\": [\"\"]
}];\n    msg.comment203 = \"if-test: prev_data.length == 0\";\n} else {\n    msg.comment203 = \"if-
test: prev_data.length != 0\";\n}\nlet temp = Number(msg.payload.data.binding.value);\n\nif
(!prev_data[0][0].data) {\n    prev_data[0][0].data = [];\n    msg.comment204 = \"if-test:
!prev_data[0][0].data\";\n} else {\n    msg.comment204 = \"if-test:
prev_data[0][0].data\";\n}\nmsg.prev_time = new Date(msg.payload.data.bin-
ding.time);\nmsg.prev_time_s = new Date(msg.payload.data.binding.time).getTime();\n\nlet time_flag
= 0;\nif (prev_data[0][0].data.length != 0) {\n    if (prev_data[0][0].data.at(-1)[0][\"x\"] != new
Date(msg.payload.data.binding.time).getTime()) {\n        msg.comment205 = \"if-test: new time: \"
+ msg.payload.data.binding.time;\n        time_flag = 1;\n    } else {\n        msg.comment205 =
\"if-test: same time: \" + msg.payload.data.binding.time;\n    }\n} else {\n    msg.comment205 =
\"if-test: prev_data[0][0].data.length == 0\";\n    time_flag = 1;\n    prev_data[0][0].data[0] =
[];\n}\nif (time_flag > 0) {\n    prev_data[0][0].data[0].push({ \"x\": new Date(msg.pay-
load.data.binding.time).getTime(), \"y\": temp });\n}\n\nmsg.payload = [{\n    \"series\": [\"Tem-
perature\"],\n    \"data\": prev_data[0][0].data,\n    \"labels\": [\"\"]\n}];\n\nmsg.prev_data =
prev_data;\n\nreturn msg;","outputs":1,"noerr":0,"initialize":"","fi-
nalize":"","libs":[],"x":1270,"y":280,"wi-
res":[["240db391c7d43cfa"]]},{"id":"240db391c7d43cfa","type":"ui_chart","z":"253389855d742e07","nam
e":"","group":"71d388d9fa9b88ac","order":3,"width":0,"height":0,"label":"Temperature (C) prosessing
lab (Adeunis Comfort)","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"li-
near","nodata":"","dot":true,"ymin":"-20","ymax":"40","removeOlder":1,"removeOlderPoints":"","re-
moveOlderUnit":"86400","cutout":0,"useOneColor":false,"useUTC":false,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":1290,"y":340,"wi-
res":[["5b0fd4fd1c757f17"]]},
```


Figure 11.69: Node-RED: importable specification for niotix dashboard, part 2.

```
{"id":"692db24e6aee0049","type":"ui_button","z":"253389855d742e07","name":"","group":"71d388d9fa9b8
8ac","order":5,"width":0,"height":0,"passthru":true,"label":"Read Humidity (Adeunis Comfort)","to-
oltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"220796","pay-
loadType":"num","topic":"topic","topicType":"msg","x":660,"y":140,"wi-
res":[["3f63fc475ee044d2"]]},{"id":"ba12507bb52e01f3","type":"ui_chart","z":"253389855d742e07","nam
e":"","group":"71d388d9fa9b88ac","order":7,"width":0,"height":0,"label":"Humidity (%) prosessing
lab (Adeunis Comfort)","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"li-
near","nodata":"","dot":true,"ymin":"0","ymax":"100","removeOlder":1,"removeOlderPoints":"","re-
moveOlderUnit":"86400","cutout":0,"useOneColor":false,"useUTC":false,"co-
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"
outputs":1,"useDifferentColor":false,"className":"","x":1280,"y":540,"wi-
res":[["d28cc8dacc4c7851"]]},{"id":"273464f9bb8782a1","type":"func-
tion","z":"253389855d742e07","name":"Reset function","func":"msg.payload = [];\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":980,"y":200,"wi-
res":[["8dcc82f5c9c6029e"]]},{"id":"d754be6404aa47cd","type":"ui_button","z":"253389855d742e07","na
me":"","group":"850d19e1e65e5ea1","order":4,"width":0,"height":0,"passthru":false,"label":"Reset
chart","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","topicType":"msg","x":810,"y":200,"wi-
res":[["273464f9bb8782a1"]]},{"id":"c1abec4763e7446b","type":"ui_button","z":"253389855d742e07","na
me":"","group":"71d388d9fa9b88ac","order":4,"width":0,"height":0,"passthru":false,"label":"Reset
chart: Temperature","tooltip":"","color":"","bgcolor":"","className":"","icon":"","pay-
load":"","payloadType":"str","topic":"topic","topicType":"msg","x":770,"y":340,"wi-
res":[["aeaf58f16c08e0fa"]]},{"id":"43bb9e0e1f145a29","type":"ui_button","z":"253389855d742e07","na
me":"","group":"71d388d9fa9b88ac","order":8,"width":0,"height":0,"passthru":false,"label":"Reset
chart: Humidity","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","topicType":"msg","x":760,"y":540,"wi-
res":[["13ac2f4f92708c9d"]]},{"id":"aeaf58f16c08e0fa","type":"func-
tion","z":"253389855d742e07","name":"Reset function","func":"msg.payload = [];\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":980,"y":340,"wi-
res":[["240db391c7d43cfa"]]},{"id":"13ac2f4f92708c9d","type":"func-
tion","z":"253389855d742e07","name":"Reset function","func":"msg.payload = [];\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":980,"y":540,"wi-
res":[["ba12507bb52e01f3"]]},{"id":"d28cc8dacc4c7851","type":"func-
tion","z":"253389855d742e07","name":"Adeunis Comfort (Humidity) save plot va-
lues","func":"flow.set(\"comf_hum\", msg.payload);\nreturn flow;\n\n","outputs":1,"noerr":0,"ini-
tialize":"","finalize":"","libs":[],"x":1280,"y":580,"wi-
res":[[]]},{"id":"5f7a2eda8254d222","type":"function","z":"253389855d742e07","name":"Process Humi-
dity (Adeunis Comfort)","func":"let prev_data = flow.get(\"comf_hum\");\nprev_data =
JSON.parse(JSON.stringify(prev_data));\n\nif (!Array.isArray(prev_data)) {\n    prev_data =
[prev_data];\n    msg.comment200 = \"if-test: !Array.isArray(prev_data)\";\n} else {\n    msg.com-
ment200 = \"if-test: Array.isArray(prev_data)\";\n}\nif (!prev_data[0]) {\n    prev_data[0] = [];\n
msg.comment201 = \"if-test: !prev_data[0]\";\n} else {\n    msg.comment201 = \"if-test:
prev_data[0]\";\n}\nif (!Array.isArray(prev_data[0])) {\n    prev_data[0] = [prev_data[0]];\n
msg.comment202 = \"if-test: !Array.isArray(prev_data[0])\";\n} else {\n    msg.comment202 = \"if-
test: Array.isArray(prev_data[0])\";\n}\nif (prev_data[0].length == 0) {\n    prev_data[0] = [{
\"series\": [\"Humidity\"], \"data\": [], \"labels\": [\"\"] }];//prev_data[0] = [{ \"series\":
[\"Humidity\"], \"data\": new Array(), \"labels\": [\"\"] }];\n    msg.comment203 = \"if-test:
prev_data.length == 0\";\n} else {\n    msg.comment203 = \"if-test: prev_data.length !=
0\";\n}\nlet rh = Number(msg.payload.data.binding.value);\n\nif (!prev_data[0][0].data) {\n
prev_data[0][0].data = [];\n    msg.comment204 = \"if-test: !prev_data[0][0].data\";\n} else {\n
msg.comment204 = \"if-test: prev_data[0][0].data\";\n}\nmsg.prev_time = new Date(msg.pay-
load.data.binding.time);\nmsg.prev_time_s = new Date(msg.payload.data.binding.time).getTime();\nlet
time_flag = 0;\nif (prev_data[0][0].data.length != 0){\n    if (prev_data[0][0].data.at(-
1)[0][\"x\"] != new Date(msg.payload.data.binding.time).getTime()) {\n        msg.comment205 =
\"if-test: new time: \" + msg.payload.data.binding.time;\n        time_flag = 1;\n    }else{\n
msg.comment205 = \"if-test: same time: \" + msg.payload.data.binding.time;\n    } \n} else {\n
msg.comment205 = \"if-test: prev_data[0][0].data.length == 0\";\n    time_flag = 1;\n
prev_data[0][0].data[0] = [];\n}\nif(time_flag > 0){\n    prev_data[0][0].data[0].push({ \"x\": new
Date(msg.payload.data.binding.time).getTime(), \"y\": rh });\n}\n\nmsg.payload = [{\n    \"se-
ries\": [\"Humidity\"],\n    \"data\": prev_data[0][0].data,\n    \"labels\":
[\"\"]\n}];\n\nmsg.prev_data = prev_data;\n\nreturn msg;","outputs":1,"noerr":0,"ini-
tialize":"","finalize":"","libs":[],"x":1250,"y":500,"wires":[["ba12507bb52e01f3"]]},
```


Figure 11.70: Node-RED: importable specification for niotix dashboard, part 3.

```
{"id":"5b0fd4fd1c757f17","type":"function","z":"253389855d742e07","name":"Adeunis Comfort (Tempera-
true) save plot values","func":"flow.set(\"comf_temp\", msg.payload);\nreturn flow;\n\n","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1290,"y":380,"wi-
res":[[]]},{"id":"69f77023af128952","type":"function","z":"253389855d742e07","name":"Adeunis Tempe-
ratrue save plot values","func":"flow.set(\"temp_temp\", msg.payload);\nreturn flow;\n","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1260,"y":160,"wi-
res":[[]]},{"id":"e7af6b3a49525f33","type":"ui_gauge","z":"253389855d742e07","name":"","group":"71d
388d9fa9b88ac","order":6,"width":0,"height":0,"gtype":"gage","title":"Humidity (%) prosessing lab
(Adeunis Comfort)","label":"%","format":"{{value}}","min":0,"max":"100","co-
lors":["#00b500","#e6e600","#ca3838"],"seg1":"25","seg2":"75","diff":false,"class-
Name":"","x":1280,"y":660,"wires":[]},{"id":"c4d1d2f61dc060b8","type":"func-
tion","z":"253389855d742e07","name":"Prepare Humidity single value (Adeunis Com-
fort)","func":"msg.payload = Number(msg.payload.data.binding.value);\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1300,"y":620,"wi-
res":[["e7af6b3a49525f33"]]},{"id":"1d4044e693040b41","type":"func-
tion","z":"253389855d742e07","name":"Prepare Temperature single value (Adeunis Com-
fort)","func":"msg.payload = Number(msg.payload.data.binding.value);\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1310,"y":420,"wi-
res":[["da7158a72e955fe0"]]},{"id":"da7158a72e955fe0","type":"ui_gauge","z":"253389855d742e07","nam
e":"","group":"71d388d9fa9b88ac","order":2,"width":0,"height":0,"gtype":"gage","title":"Temperature
(C) prosessing lab (Adeunis Comfort)","label":"C","format":"{{value}}","min":"-20","max":"40","co-
lors":["#00b500","#e6e600","#ca3838"],"seg1":"0","seg2":"20","diff":false,"class-
Name":"","x":1290,"y":460,"wi-
res":[]},{"id":"d8707f55933a55a4","type":"ui_gauge","z":"253389855d742e07","name":"","group":"850d1
9e1e65e5ea1","order":2,"width":0,"height":0,"gtype":"gage","title":"Temperature (C) outside
(Adeunis Temperature)","label":"C","format":"{{value}}","min":"-20","max":"40","co-
lors":["#00b500","#e6e600","#ca3838"],"seg1":"0","seg2":"20","diff":false,"class-
Name":"","x":1280,"y":80,"wires":[]},{"id":"f06ecdcafcc00fa0","type":"func-
tion","z":"253389855d742e07","name":"Prepare Temperature single value (Adeunis Tempera-
ture)","func":"msg.payload = Number(msg.payload.data.binding.value);\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1330,"y":120,"wi-
res":[["d8707f55933a55a4"]]},{"id":"850d19e1e65e5ea1","type":"ui_group","name":"Adeunis Tempera-
ture","tab":"f67cd0c1fb381006","order":2,"disp":true,"width":"6","collapse":false,"class-
Name":""},{"id":"71d388d9fa9b88ac","type":"ui_group","name":"Adeunis Com-
fort","tab":"f67cd0c1fb381006","order":1,"disp":true,"width":"6","collapse":false,"class-
Name":""},{"id":"f67cd0c1fb381006","type":"ui_tab","name":"USN Campus in Porsgrunn
Niotix","icon":"dashboard","disabled":false,"hidden":false}]
```


Figure 11.71: Node-RED: importable specification for niotix dashboard, part 4.

Each button is setup to provide id of the state of interest in the payload (Figure 11.72), the id is available on niotix webpage in Digital Twin description.

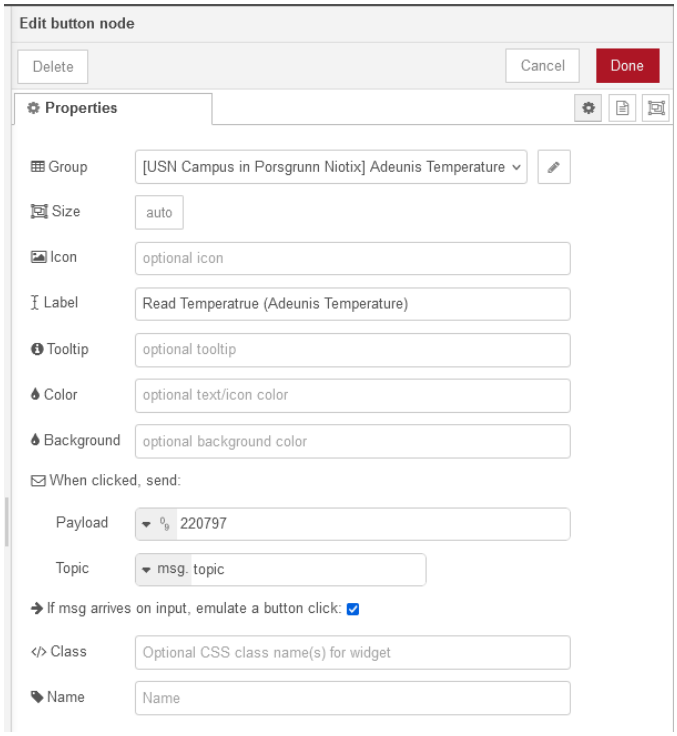


Figure 11.72: Node-RED: IoT Hub: button settings.

URL for states includes a general base address of the API (Figure 11.73). URL is saved into msg.url member to provide the address in the HTTP request node.

```
msg.url = "https://xapi.niota.io/xapi/v1/states/";
return msg;
```

Figure 11.73: Node-RED: IoT Hub: base URL.

In this function the state id is concatenated to the URL. Additionally, the key for reading the state is assigned to the dedicated member of the message (Figure 11.74). These two msg members when received by HTTP-node modify the corresponding field for outgoing request.

```
msg.apiKey = "82D714C9-DB85-45CA-8A95-27B824A125F7";
msg.url = msg.url + msg.payload;
return msg;
```

Figure 11.74: Node-RED: IoT Hub: state specific URL.

HTTP request is sent by HTTP request node (Figure 11.75) using GET method with predefined request URL and API key. msg.url will be automatically assigned to the URL field of the node, while security key must be manually assigned to custom header “x-api-key”.

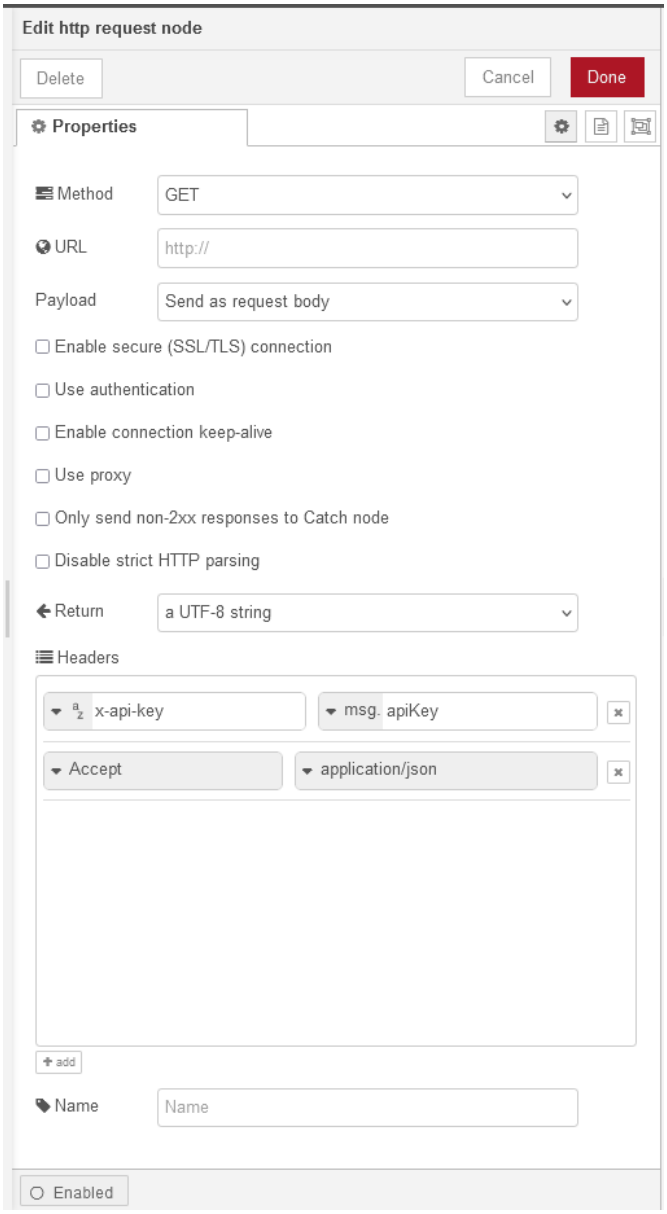


Figure 11.75: Node-RED: IoT Hub: HTTP request node setup.

The response from HTTP-node is forwarded to a JSON node to convert it into JSON object for convenience.

After conversion, the “Switch”-node distributes the output depending on the value in the specific member of incoming message (Figure 11.76). From documentation to the API, one can see that the output should contain the state id in msg.payload.data.id. The state id from the button was transmitted in “payload” member of msg and, therefore, is overwritten by the new payload in the consequent node. Because of that, it should be captured from the payload from HTTP node and not from the original payload from the button.

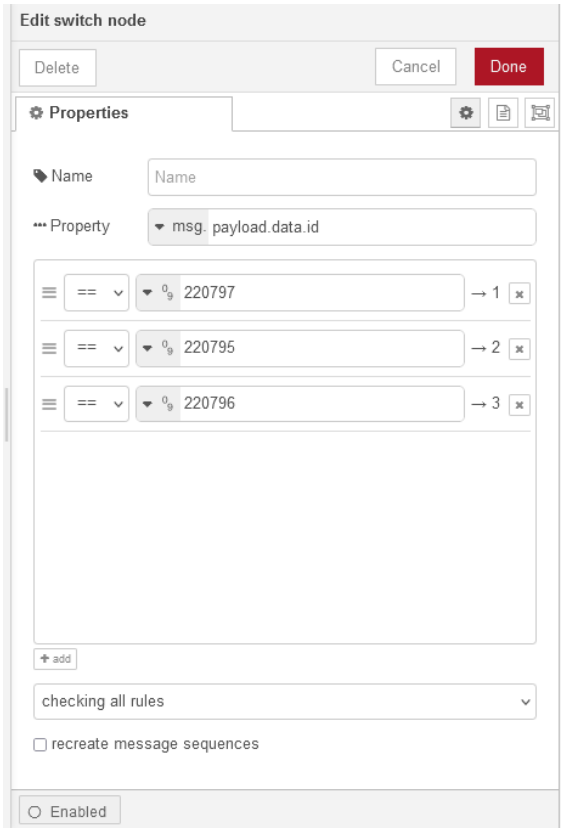


Figure 11.76: Node-RED: IoT Hub: switch node setup.

Preparing of the acquired value for plotting takes several steps (Figure 11.77). It is desired to have all previously requested values in one time series with the new value. For that purpose, the time series already stored in the plot is captured and saved in flow-variable available to all nodes in the flow (Figure 11.78).
If the current run is the first one and there are no values in the chart the flow-variable will be an empty array. In this case, an array of the specific structure must be constructed first, and then the new value is appended to it.
The algorithm also examines if the newly read value is already present in the chart to avoid duplicates. API responses with only one last value from the state, with a timestamp showing when the measurements had been taken. The duplicates are found by comparing the timestamp of the new value with the stored timestamps.

```
function int_unsigned_to_signed(val){
    if (val && 0x8000) {
        val -= 0x10000;
        val/=10;
    }
    return val;
}

let prev_data = flow.get("temp_temp");
prev_data = JSON.parse(JSON.stringify(prev_data));

if (!Array.isArray(prev_data)) {
    prev_data = [prev_data];
}
if (!prev_data[0]) {
    prev_data[0] = [];
}
if (!Array.isArray(prev_data[0])) {
    prev_data[0] = [prev_data[0]];
}
if (prev_data[0].length == 0) {
    prev_data[0] = [{ "series": ["Temperature"], "data": [], "labels": [""] }];
}
let temp = Number(msg.payload.data.binding.value);

if (temp*10 > 2000){
    temp = int_unsigned_to_signed(temp*10);
}

if (!prev_data[0][0].data) {
    prev_data[0][0].data = [];
}

let time_flag = 0;
if (prev_data[0][0].data.length != 0) {
    if (prev_data[0][0].data.at(-1)[0]["x"] != new Date(msg.payload.data.binding.time).get-
Time()) {
        time_flag = 1;
    }
} else {
    time_flag = 1;
    prev_data[0][0].data[0] = [];
}
if (time_flag > 0) {
    prev_data[0][0].data[0].push({ "x": new Date(msg.payload.data.binding.time).getTime(),
"y": temp });
}

msg.payload = [{
    "series": ["Temperature"],
    "data": prev_data[0][0].data,
    "labels": [""]
}];

return msg;
```

Figure 11.77: Node-RED: IoT Hub: response processing for Adeunis Temperature sensor.

```
flow.set("temp_temp", msg.payload);
return flow;
```

Figure 11.78: Node-RED: IoT Hub: save plot value into flow-variable for Adeunis Temperature sensor.

The single value for the gauge widget is simply acquired from the corresponding member of the payload and converted to a numeric value (Figure 11.79). This process is identical for all sensors.

```
msg.payload = Number(msg.payload.data.binding.value);
return msg;
```

Figure 11.79: Node-RED: IoT Hub: acquire single value for gauge node.

Gauge widget does not require specific settings, only those that make monitoring intuitive and convenient (Figure 11.80). Values range has been defined from −20℃ to +40℃ as the most realistic. The widget itself has also been assigned to the group related to the sensor the data is read from.

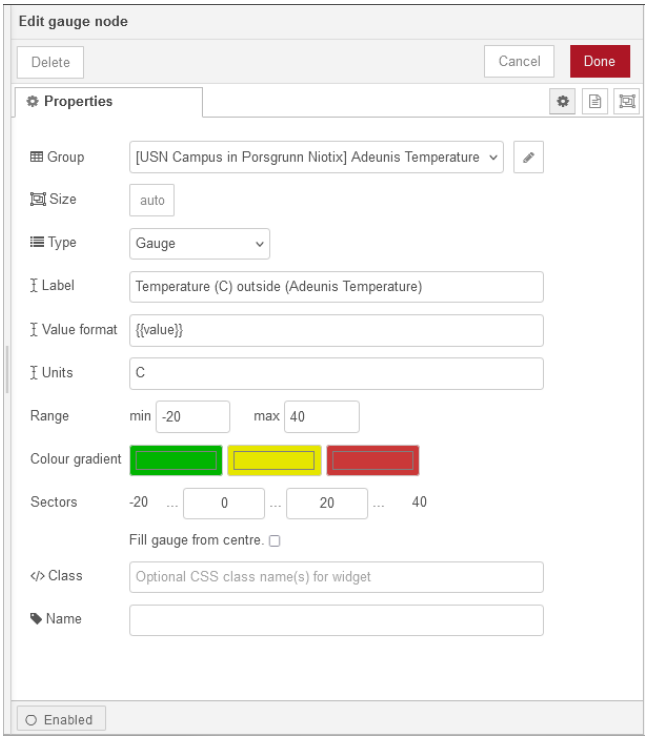


Figure 11.80: Node-RED: IoT Hub: gauge node setup.

```
msg.apiKey = "82D714C9-DB85-45CA-8A95-27B824A125F7";
msg.url = msg.url + msg.payload;
return msg;
```

Figure 11.81: Node-RED: IoT Hub: Outdoor Digital Twin reader key.

```
msg.apiKey = "2A463053-6194-40C8-82AE-E6988A81FAC6";
msg.url = msg.url + msg.payload;
return msg;
```

Figure 11.82: Node-RED: IoT Hub: Processing hall Digital Twin reader key.

```
flow.set("temp_temp", msg.payload);
return flow;
```

Figure 11.83: Node-RED: IoT Hub: saving data into flow-variable for Adeunis Temperature sensor.

```
let prev_data = flow.get("comf_temp");
prev_data = JSON.parse(JSON.stringify(prev_data));

if (!Array.isArray(prev_data)) {
    prev_data = [prev_data];
}
if (!prev_data[0]) {
    prev_data[0] = [];
}
if (!Array.isArray(prev_data[0])) {
    prev_data[0] = [prev_data[0]];
}
if (prev_data[0].length == 0) {
    prev_data[0] = [{ "series": ["Temperature"], "data": [], "labels": [""] }];
}
let temp = Number(msg.payload.data.binding.value);

if (!prev_data[0][0].data) {
    prev_data[0][0].data = [];
}
msg.prev_time = new Date(msg.payload.data.binding.time);
msg.prev_time_s = new Date(msg.payload.data.binding.time).getTime();

let time_flag = 0;
if (prev_data[0][0].data.length != 0) {
    if (prev_data[0][0].data.at(-1)[0]["x"] != new Date(msg.payload.data.binding.time).get-
Time()) {
        time_flag = 1;
    }
} else {
    time_flag = 1;
    prev_data[0][0].data[0] = [];
}
if (time_flag > 0) {
    prev_data[0][0].data[0].push({ "x": new Date(msg.payload.data.binding.time).getTime(),
"y": temp });
}

msg.payload = [{
    "series": ["Temperature"],
    "data": prev_data[0][0].data,
    "labels": [""]
}];

msg.prev_data = prev_data;

return msg;
```

Figure 11.84: Node-RED: IoT Hub: response processing for temperature from Adeunis Comfort sensor.

```
flow.set("comf_temp", msg.payload);
return flow;
```

Figure 11.85: Node-RED: IoT Hub: saving temperature into flow-variable for Adeunis Comfort sensor.

```
let prev_data = flow.get("comf_hum");
prev_data = JSON.parse(JSON.stringify(prev_data));

if (!Array.isArray(prev_data)) {
    prev_data = [prev_data];
}
if (!prev_data[0]) {
    prev_data[0] = [];
}
if (!Array.isArray(prev_data[0])) {
    prev_data[0] = [prev_data[0]];
}
if (prev_data[0].length == 0) {
    prev_data[0] = [{ "series": ["Humidity"], "data": [], "labels": [""] }];
}
let rh = Number(msg.payload.data.binding.value);

if (!prev_data[0][0].data) {
    prev_data[0][0].data = [];
}
msg.prev_time = new Date(msg.payload.data.binding.time);
msg.prev_time_s = new Date(msg.payload.data.binding.time).getTime();
let time_flag = 0;
if (prev_data[0][0].data.length != 0){
    if (prev_data[0][0].data.at(-1)[0]["x"] != new Date(msg.payload.data.binding.time).get-
Time()) {
        time_flag = 1;
    }
} else {
    time_flag = 1;
    prev_data[0][0].data[0] = [];
}
if(time_flag > 0){
    prev_data[0][0].data[0].push({ "x": new Date(msg.payload.data.binding.time).getTime(),
"y": rh });
}

msg.payload = [{
    "series": ["Humidity"],
    "data": prev_data[0][0].data,
    "labels": [""]
}];

msg.prev_data = prev_data;

return msg;
```

Figure 11.86: Node-RED: IoT Hub: response processing for humidity from Adeunis Comfort sensor.

```
flow.set("comf_hum", msg.payload);
return flow;
```

Figure 11.87: Node-RED: IoT Hub: saving humidity into flow-variable for Adeunis Comfort sensor.

11.11 Appendix K. Node-RED dashboard “USN Campus in Porsgrunn InfluxDB”.

Figure 11.88, Figure 11.89, Figure 11.90, Figure 11.91, Figure 11.92, and Figure 11.93 represent a .json file content, they can be combined in one file and imported into Node-RED flow.


```
[{"id":"8802621c69ad7966","type":"tab","label":"LoRaWAN dashboard InfluxDB","disa-
bled":false,"info":"","env":[]},{"id":"81776dc0bd796c7e","type":"http re-
quest","z":"8802621c69ad7966","name":"","method":"POST","ret":"txt","paytoqs":"ig-
nore","url":"","tls":"cc11521750f5f168","persist":false,"proxy":"","insecureHTTP-
Parser":false,"authType":"","senderr":false,"headers":[{"keyType":"other","keyValue":"x-api-
key","valueType":"msg","valueValue":"apiKey"},{"keyType":"Accept","keyValue":"","valueType":"appli-
cation/json","value-
Value":""}],"x":450,"y":400,"wires":[["ae28aa580313a50d"]]},{"id":"ae28aa580313a50d","type":"json",
"z":"8802621c69ad7966","name":"","property":"payload","ac-
tion":"","pretty":false,"x":590,"y":400,"wires":[["f5e93e05b79442f1"]]},{"id":"840803fe55db359d","t
ype":"function","z":"8802621c69ad7966","name":"Reset function","func":"msg.payload = [];\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":300,"y":160,"wires":[["0e67683d525eaa00"]]},{"id":"118eb14ac0b90f2e","type":"
ui_button","z":"8802621c69ad7966","name":"","group":"adf87389b95fb318","or-
der":5,"width":0,"height":0,"passthru":false,"label":"Reset
chart","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","top-
icType":"msg","x":90,"y":160,"wires":[["840803fe55db359d"]]},{"id":"986df1d6464ba608","type":"ui_da
te_picker","z":"8802621c69ad7966","name":"","label":"Time start (Adeunis Tempera-
ture)","group":"adf87389b95fb318","or-
der":1,"width":0,"height":0,"passthru":false,"topic":"topic","topicType":"msg","class-
Name":"","x":150,"y":40,"wires":[["31f3d30650fd95f1"]]},{"id":"d76b2538dbfa25bc","type":"ui_date_pi
cker","z":"8802621c69ad7966","name":"","label":"Time end (Adeunis Tempera-
ture)","group":"adf87389b95fb318","or-
der":2,"width":0,"height":0,"passthru":true,"topic":"topic","topicType":"msg","class-
Name":"","x":150,"y":80,"wires":[["3c6e5d67a189676d"]]},{"id":"0e67683d525eaa00","type":"ui_chart",
"z":"8802621c69ad7966","name":"","group":"adf87389b95fb318","order":4,"width":0,"height":0,"la-
bel":"Temperature (C) outside (Adeunis Temperature)","chartType":"line","legend":"false","xfor-
mat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"-20","ymax":"40","remove-
Older":1,"removeOlderPoints":"","removeOlderUnit":"86400","cut-
out":0,"useOneColor":false,"useUTC":true,"col-
ors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"o
utputs":1,"useDifferentColor":false,"class-
Name":"","x":980,"y":160,"wires":[["9c98e46dd418ced1"]]},{"id":"51d10ce042a87577","type":"ui_but-
ton","z":"8802621c69ad7966","name":"","group":"adf87389b95fb318","or-
der":3,"width":0,"height":0,"passthru":false,"label":"Read Tempera-
true","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"220797","pay-
loadType":"num","topic":"topic","top-
icType":"msg","x":110,"y":120,"wires":[["090cc968728bd1b6"]]},{"id":"31f3d30650fd95f1","type":"func
tion","z":"8802621c69ad7966","name":"Define start time","func":"msg.time_start = msg.pay-
load;\nmsg.parts = {\n    \"id\" : 1,\n    \"index\" : 1,\n    \"count\" : 2,\n    \"type\" : Ob-
ject,\n    \"key\" : \"time\"\n}\nreturn msg; ","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":400,"y":40,"wires":[["2031cb2416c23b60"]]},{"id":"3c6e5d67a189676d","type":"f
unction","z":"8802621c69ad7966","name":"Define end time","func":"msg.time_end = msg.pay-
load;\nmsg.parts = {\n    \"id\": 1,\n    \"index\": 2,\n    \"count\": 2,\n    \"type\": Object,\n
\"key\": \"time\"\n}\nreturn msg; ","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":400,"y":80,"wires":[["2031cb2416c23b60"]]},
```


Figure 11.88: Node-RED: importable specification for InfluxDB dashboard, part 1.

```
{"id":"090cc968728bd1b6","type":"function","z":"8802621c69ad7966","name":"Prepare URL for Tempera-
ture","func":"msg.time_start_temp_temp =
flow.get(\"time_start_temp_temp\");\nmsg.time_start_temp_temp = new
Date(msg.time_start_temp_temp);\nmsg.time_start_temp_temp = msg.time_start_temp_temp.get-
Time();\nmsg.time_end_temp_temp = flow.get(\"time_end_temp_temp\");\nmsg.time_end_temp_temp = new
Date(msg.time_end_temp_temp);\nmsg.time_end_temp_temp = msg.time_end_temp_temp.get-
Time();\n\nmsg.url = \"https://xapi.niota.io/xapi/v1/influxdb/query?epoch=ms&q=\";\n//msg.query =
\"SELECT mean(\\\"value_number\\\") FROM \\\"states_history\\\" WHERE time >= now() - 6h GROUP BY
time(15s) fill(null)\"\nmsg.query = \"SELECT first(\\\"value_number\\\") FROM \\\"states_his-
tory\\\" \\\nWHERE(\\\"state_id\\\" = '\"+ msg.payload + \"') AND time >= \" +
msg.time_start_temp_temp + \"ms and time <= \" + msg.time_end_temp_temp + \"ms \\\nGROUP BY
time(15m) fill(null)\"\nmsg.apiKey = \"82D714C9-DB85-45CA-8A95-27B824A125F7\";  // Outdoor\nmsg.url
= msg.url + msg.query;\nreturn msg;\n\n//SELECT first(\"value\") FROM \"°C\" WHERE (\"device_eui\"
= '0018B21000003BBF') AND time >= now() - 24h GROUP BY time(10m) fill(null)\n//SELECT
first(\"value_number\") FROM \"states_history\" WHERE (\"dtwin_id\" = '24514' AND \"state_id\" =
'220795') AND time >= now() - 24h GROUP BY time(10m) fill(null)\n//SELECT first(\"value_number\")
FROM \"states_history\" WHERE (\"dtwin_id\" = '24514' AND \"state_id\" = '220795') AND time >=
1680300000000ms and time <= 1680559199000ms GROUP BY time(15m) fill(null)\n//Request
url\n//https://xapi.niota.io/xapi/v1/influxdb/query?epoch=s&q=SELECT%20first%28%22value_num-
ber%22%29%20FROM%20%22states_his-
tory%22%20WHERE%20%28%22dtwin_id%22%20%3D%20%2724514%27%20AND%20%22state_id%22%20%3D%20%27220795%27
%29%20AND%20time%20%3E%3D%201680300000000ms%20and%20time%20%3C%3D%201680559199000ms%20GROUP%20BY%20
time%2815m%29%20fill%28null%29\n\n","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":350,"y":120,"wires":[["9a0bac3cc6e89ce2"]]},{"id":"2031cb2416c23b60","type":"
join","z":"8802621c69ad7966","name":"","mode":"auto","build":"object","property":"payload","proper-
tyType":"msg","key":"topic","joiner":"\\n","joinerType":"str","accumu-
late":true,"timeout":"","count":"","reduceRight":false,"reduceExp":"","reduceInit":"","re-
duceInitType":"","re-
duceFixup":"","x":550,"y":60,"wires":[["9cb8c1c901e95fa6"]]},{"id":"9cb8c1c901e95fa6","type":"func-
tion","z":"8802621c69ad7966","name":"Set time limits in flow varia-
ble","func":"flow.set(\"time_start_temp_temp\", msg.time_start);\nflow.set(\"time_end_temp_temp\",
msg.time_end);\nreturn msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":740,"y":60,"wires":[[]]},{"id":"64318f531604d63a","type":"func-
tion","z":"8802621c69ad7966","name":"Prepare URL for Temperature (Adeunis Com-
fort)","func":"msg.time_start_comf_temp =
flow.get(\"time_start_comf_temp\");\nmsg.time_start_comf_temp = new
Date(msg.time_start_comf_temp);\nmsg.time_start_comf_temp = msg.time_start_comf_temp.get-
Time();\nmsg.time_end_comf_temp = flow.get(\"time_end_comf_temp\");\nmsg.time_end_comf_temp = new
Date(msg.time_end_comf_temp);\nmsg.time_end_comf_temp = msg.time_end_comf_temp.getTime();\n\n//let
state_temp = 220795;  // Temperature\n\n//msg.url = \"https://xapi.niota.io/xapi/v1/in-
fluxdb/\";\nmsg.url = \"https://xapi.niota.io/xapi/v1/influxdb/query?epoch=ms&q=\";\n//msg.query =
\"SELECT mean(\\\"value_number\\\") FROM \\\"states_history\\\" WHERE time >= now() - 6h GROUP BY
time(15s) fill(null)\"\nmsg.query = \"SELECT first(\\\"value_number\\\") FROM \\\"states_his-
tory\\\" \\\nWHERE(\\\"state_id\\\" = '\"+msg.payload+\"') AND time >= \"+ msg.time_start_comf_temp
+ \"ms and time <= \" + msg.time_end_comf_temp+\"ms \\\nGROUP BY time(15m) fill(null)\"\n//SELECT
first(\"value\") FROM \"°C\" WHERE (\"device_eui\" = '0018B21000003BBF') AND time >= now() - 24h
GROUP BY time(10m) fill(null)\n//SELECT first(\"value_number\") FROM \"states_history\" WHERE
(\"dtwin_id\" = '24514' AND \"state_id\" = '220795') AND time >= now() - 24h GROUP BY time(10m)
fill(null)\n//SELECT first(\"value_number\") FROM \"states_history\" WHERE (\"dtwin_id\" = '24514'
AND \"state_id\" = '220795') AND time >= 1680300000000ms and time <= 1680559199000ms GROUP BY
time(15m) fill(null)\n//Request url\n//https://xapi.niota.io/xapi/v1/influxdb/query?epoch=s&q=SE-
LECT%20first%28%22value_number%22%29%20FROM%20%22states_his-
tory%22%20WHERE%20%28%22dtwin_id%22%20%3D%20%2724514%27%20AND%20%22state_id%22%20%3D%20%27220795%27
%29%20AND%20time%20%3E%3D%201680300000000ms%20and%20time%20%3C%3D%201680559199000ms%20GROUP%20BY%20
time%2815m%29%20fill%28null%29\n\nmsg.apiKey = \"2A463053-6194-40C8-82AE-E6988A81FAC6\";  // Pro-
cessing lab\nmsg.url = msg.url + msg.query;\nreturn msg;","outputs":1,"noerr":0,"initial-
ize":"","finalize":"","libs":[],"x":350,"y":440,"wires":[["81776dc0bd796c7e"]]},
```


Figure 11.89: Node-RED: importable specification for InfluxDB dashboard, part 2.

```
{"id":"f5e93e05b79442f1","type":"function","z":"8802621c69ad7966","name":"Prepare data for Tempera-
ture (Adeunis Comfort)","func":"let data = [];\nlet data_element = [];\nfor (let idx = 0; idx <
msg.payload.results[0].series[0].values.length; idx++){\n    data_element.push({ \"x\": msg.pay-
load.results[0].series[0].values[idx][0], \"y\": msg.payload.results[0].series[0].val-
ues[idx][1]});\n}\ndata.push(data_element);\nmsg.payload = [{\n    \"series\": [\"Temperature\"],\n
\"data\": data,\n    \"labels\": [\"\"]\n}];\nreturn msg;","outputs":1,"noerr":0,"initial-
ize":"","final-
ize":"","libs":[],"x":850,"y":400,"wires":[["220b69ba605c048a"]]},{"id":"220b69ba605c048a","type":"
ui_chart","z":"8802621c69ad7966","name":"","group":"5be966968ca3837c","or-
der":4,"width":"0","height":"0","label":"Temperature (C) prosessing lab (Adeunis Comfort)","chart-
Type":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","no-
data":"","dot":true,"ymin":"-20","ymax":"40","removeOlder":"1","removeOlderPoints":"","removeOlder-
Unit":"86400","cutout":0,"useOneColor":false,"useUTC":true,"col-
ors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"o
utputs":1,"useDifferentColor":false,"class-
Name":"","x":850,"y":440,"wires":[["ebce13a47d16c8c3"]]},{"id":"08556cd689c97716","type":"ui_date_p
icker","z":"8802621c69ad7966","name":"","label":"Time start for Temperature (Adeunis Com-
fort)","group":"5be966968ca3837c","or-
der":1,"width":0,"height":0,"passthru":false,"topic":"topic","topicType":"msg","class-
Name":"","x":190,"y":240,"wires":[["0b55886b0e8fab65"]]},{"id":"8b50e15f300bf19b","type":"ui_date_p
icker","z":"8802621c69ad7966","name":"","label":"Time end for Temperature (Adeunis Com-
fort)","group":"5be966968ca3837c","or-
der":2,"width":0,"height":0,"passthru":false,"topic":"topic","topicType":"msg","class-
Name":"","x":190,"y":280,"wires":[["3c8036f4365df480"]]},{"id":"0b55886b0e8fab65","type":"func-
tion","z":"8802621c69ad7966","name":"Define start time for Temperature (Adeunis Com-
fort)","func":"msg.time_start = msg.payload;\nmsg.parts = {\n    \"id\": 1,\n    \"index\": 1,\n
\"count\": 2,\n    \"type\": Object,\n    \"key\": \"time\"\n}\nreturn msg; ","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":590,"y":240,"wires":[["7a8eb945cb8f42a4"]]},{"id":"3c8036f4365df480","type":"
function","z":"8802621c69ad7966","name":"Define end time for Temperature (Adeunis Com-
fort)","func":"msg.time_end = msg.payload;\nmsg.parts = {\n    \"id\": 1,\n    \"index\": 2,\n
\"count\": 2,\n    \"type\": Object,\n    \"key\": \"time\"\n}\nreturn msg; ","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":590,"y":280,"wires":[["7a8eb945cb8f42a4"]]},{"id":"0e77f80fac6275c7","type":"
function","z":"8802621c69ad7966","name":"Set time limits in flow variable Temperature (Adeunis Com-
fort)","func":"flow.set(\"time_start_comf_temp\",
msg.time_start);\nflow.set(\"time_end_comf_temp\", msg.time_end);\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":1170,"y":260,"wires":[[]]},{"id":"7a8eb945cb8f42a4","type":"join","z":"880262
1c69ad7966","name":"","mode":"auto","build":"object","property":"payload","property-
Type":"msg","key":"topic","joiner":"\\n","joinerType":"str","accumu-
late":true,"timeout":"","count":"","reduceRight":false,"reduceExp":"","reduceInit":"","re-
duceInitType":"","re-
duceFixup":"","x":870,"y":260,"wires":[["0e77f80fac6275c7"]]},{"id":"d04d99733621b685","type":"join
","z":"8802621c69ad7966","name":"","mode":"auto","build":"object","property":"payload","property-
Type":"msg","key":"topic","joiner":"\\n","joinerType":"str","accumu-
late":"false","timeout":"","count":"","re-
duceRight":false,"x":870,"y":340,"wires":[["7be8f72fcda89495"]]},{"id":"8bcbf53ea2a75e55","type":"f
unction","z":"8802621c69ad7966","name":"Define start time for Humidity (Adeunis Com-
fort)","func":"msg.time_start = msg.payload;\nmsg.parts = {\n    \"id\": 1,\n    \"index\": 1,\n
\"count\": 2,\n    \"type\": Object,\n    \"key\": \"time\"\n}\nreturn msg; ","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":580,"y":320,"wires":[["d04d99733621b685"]]},{"id":"8282b38aa1c2cd10","type":"
function","z":"8802621c69ad7966","name":"Define end time for Humidity (Adeunis Com-
fort)","func":"msg.time_end = msg.payload;\nmsg.parts = {\n    \"id\": 1,\n    \"index\": 3,\n
\"count\": 2,\n    \"type\": Object,\n    \"key\": \"time\"\n}\nreturn msg; ","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":570,"y":360,"wires":[["d04d99733621b685"]]},{"id":"7be8f72fcda89495","type":"
function","z":"8802621c69ad7966","name":"Set time limits in flow variable Humidity (Adeunis Com-
fort)","func":"flow.set(\"time_start_comf_hum\", msg.time_start);\nflow.set(\"time_end_comf_hum\",
msg.time_end);\nreturn msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":1150,"y":340,"wires":[[]]},{"id":"9741c74c2d9e9c68","type":"ui_date_picker","
z":"8802621c69ad7966","name":"","label":"Time start for Humidity (Adeunis Com-
fort)","group":"5be966968ca3837c","or-
der":7,"width":0,"height":0,"passthru":true,"topic":"topic","topicType":"msg","class-
Name":"","x":180,"y":320,"wires":[["8bcbf53ea2a75e55"]]},
```


Figure 11.90: Node-RED: importable specification for InfluxDB dashboard, part 3.

```
{"id":"4e9803aca5163ac6","type":"ui_date_picker","z":"8802621c69ad7966","name":"","label":"Time end
for Humidity (Adeunis Comfort)","group":"5be966968ca3837c","or-
der":8,"width":0,"height":0,"passthru":true,"topic":"topic","topicType":"msg","class-
Name":"","x":180,"y":360,"wires":[["8282b38aa1c2cd10"]]},{"id":"2ddfe4608a4775c4","type":"ui_but-
ton","z":"8802621c69ad7966","name":"","group":"5be966968ca3837c","or-
der":3,"width":0,"height":0,"passthru":false,"label":"Read Temperatrue (Adeunis Com-
fort)","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"220795","pay-
loadType":"num","topic":"topic","top-
icType":"msg","x":170,"y":400,"wires":[["64318f531604d63a"]]},{"id":"0d59d8ea1015227c","type":"ui_b
utton","z":"8802621c69ad7966","name":"","group":"5be966968ca3837c","or-
der":9,"width":0,"height":0,"passthru":false,"label":"Read Humidity (Adeunis Com-
fort)","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"220796","pay-
loadType":"num","topic":"topic","top-
icType":"msg","x":160,"y":540,"wires":[["a5d140c81c3c0b8e"]]},{"id":"a5d140c81c3c0b8e","type":"func
tion","z":"8802621c69ad7966","name":"Prepare URL for Humidity (Adeunis Com-
fort)","func":"msg.time_start_comf_hum =
flow.get(\"time_start_comf_hum\");\nmsg.time_start_comf_hum = new
Date(msg.time_start_comf_hum);\nmsg.time_start_comf_hum = msg.time_start_comf_hum.get-
Time();\nmsg.time_end_comf_hum = flow.get(\"time_end_comf_hum\");\nmsg.time_end_comf_hum = new
Date(msg.time_end_comf_hum);\nmsg.time_end_comf_hum = msg.time_end_comf_hum.getTime();\n\n//let
state_hum = 220796; // Humidity\n\n//msg.url = \"https://xapi.niota.io/xapi/v1/in-
fluxdb/\";\nmsg.url = \"https://xapi.niota.io/xapi/v1/influxdb/query?epoch=ms&q=\";\n//msg.query =
\"SELECT mean(\\\"value_number\\\") FROM \\\"states_history\\\" WHERE time >= now() - 6h GROUP BY
time(15s) fill(null)\"\nmsg.query = \"SELECT first(\\\"value_number\\\") FROM \\\"states_his-
tory\\\" \\\nWHERE(\\\"state_id\\\" = '\"+ msg.payload + \"') AND time >= \" +
msg.time_start_comf_hum + \"ms and time <= \" + msg.time_end_comf_hum + \"ms \\\nGROUP BY time(15m)
fill(null)\"\n//SELECT first(\"value\") FROM \"°C\" WHERE (\"device_eui\" = '0018B21000003BBF') AND
time >= now() - 24h GROUP BY time(10m) fill(null)\n//SELECT first(\"value_number\") FROM
\"states_history\" WHERE (\"dtwin_id\" = '24514' AND \"state_id\" = '220795') AND time >= now() -
24h GROUP BY time(10m) fill(null)\n//SELECT first(\"value_number\") FROM \"states_history\" WHERE
(\"dtwin_id\" = '24514' AND \"state_id\" = '220795') AND time >= 1680300000000ms and time <=
1680559199000ms GROUP BY time(15m) fill(null)\n//Request url\n//https://xapi.niota.io/xapi/v1/in-
fluxdb/query?epoch=s&q=SELECT%20first%28%22value_number%22%29%20FROM%20%22states_his-
tory%22%20WHERE%20%28%22dtwin_id%22%20%3D%20%2724514%27%20AND%20%22state_id%22%20%3D%20%27220795%27
%29%20AND%20time%20%3E%3D%201680300000000ms%20and%20time%20%3C%3D%201680559199000ms%20GROUP%20BY%20
time%2815m%29%20fill%28null%29\n\nmsg.apiKey = \"2A463053-6194-40C8-82AE-E6988A81FAC6\";  // Pro-
cessing lab\nmsg.url = msg.url + msg.query;\nreturn msg;","outputs":1,"noerr":0,"initial-
ize":"","final-
ize":"","libs":[],"x":330,"y":580,"wires":[["b5108c265310d45e"]]},{"id":"b5108c265310d45e","type":"
http request","z":"8802621c69ad7966","name":"","method":"POST","ret":"txt","paytoqs":"ig-
nore","url":"","tls":"cc11521750f5f168","persist":false,"proxy":"","insecureHTTP-
Parser":false,"authType":"","senderr":false,"headers":[{"keyType":"other","keyValue":"x-api-
key","valueType":"msg","valueValue":"apiKey"},{"keyType":"Accept","keyValue":"","valueType":"appli-
cation/json","value-
Value":""}],"x":430,"y":540,"wires":[["0e397f5a8cd303ef"]]},{"id":"08ea43940ad4acea","type":"func-
tion","z":"8802621c69ad7966","name":"Prepare data for Humidity (Adeunis Comfort)","func":"let data
= [];\nlet data_element = [];\nfor (let idx = 0; idx < msg.payload.results[0].series[0].val-
ues.length; idx++) {\n    data_element.push({ \"x\": msg.payload.results[0].series[0].val-
ues[idx][0], \"y\": msg.payload.results[0].series[0].values[idx][1] });\n}\ndata.push(data_ele-
ment);\nmsg.payload = [{\n    \"series\": [\"Humidity\"],\n    \"data\": data,\n    \"labels\":
[\"\"]\n}];\nreturn msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":810,"y":540,"wires":[["0e1e0e5a1c004403"]]},{"id":"0e397f5a8cd303ef","type":"
json","z":"8802621c69ad7966","name":"","property":"payload","ac-
tion":"","pretty":false,"x":570,"y":540,"wires":[["08ea43940ad4acea"]]},{"id":"0e1e0e5a1c004403","t
ype":"ui_chart","z":"8802621c69ad7966","name":"","group":"5be966968ca3837c","or-
der":10,"width":0,"height":0,"label":"Humidity (%) prosessing lab (Adeunis Comfort)","chart-
Type":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","no-
data":"","dot":true,"ymin":"0","ymax":"100","removeOlder":1,"removeOlderPoints":"","removeOlderU-
nit":"86400","cutout":0,"useOneColor":false,"useUTC":true,"col-
ors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"o
utputs":1,"useDifferentColor":false,"class-
Name":"","x":820,"y":580,"wires":[["5ec877a7b0bd8327"]]},{"id":"4a93e1c36d7f419a","type":"ui_but-
ton","z":"8802621c69ad7966","name":"","group":"5be966968ca3837c","or-
der":11,"width":0,"height":0,"passthru":false,"label":"Reset chart: Humid-
ity","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","topicType":"msg","x":120,"y":620,"wires":[["193f545a0367e64a"]]},
```


Figure 11.91: Node-RED: importable specification for InfluxDB dashboard, part 4.

```
{"id":"4168158854ada8e8","type":"ui_but-
ton","z":"8802621c69ad7966","name":"","group":"5be966968ca3837c","or-
der":5,"width":0,"height":0,"passthru":false,"label":"Reset chart: Tempera-
ture","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","top-
icType":"msg","x":130,"y":480,"wires":[["4335f57fa38be0cf"]]},{"id":"4335f57fa38be0cf","type":"func
tion","z":"8802621c69ad7966","name":"Reset function","func":"msg.payload = [];\nreturn msg;","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":360,"y":480,"wires":[["220b69ba605c048a"]]},{"id":"193f545a0367e64a","type":"
function","z":"8802621c69ad7966","name":"Reset function","func":"msg.payload = [];\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":360,"y":620,"wires":[["0e1e0e5a1c004403"]]},{"id":"9a0bac3cc6e89ce2","type":"
http request","z":"8802621c69ad7966","name":"","method":"POST","ret":"txt","paytoqs":"ig-
nore","url":"","tls":"cc11521750f5f168","persist":false,"proxy":"","insecureHTTP-
Parser":false,"authType":"","senderr":false,"headers":[{"keyType":"other","keyValue":"x-api-
key","valueType":"msg","valueValue":"apiKey"},{"keyType":"Accept","keyValue":"","valueType":"appli-
cation/json","value-
Value":""}],"x":570,"y":120,"wires":[["dcea13a2bdae701b"]]},{"id":"118a8a3be40a276d","type":"func-
tion","z":"8802621c69ad7966","name":"Prepare data for Temperature (Adeunis Tempera-
ture)","func":"function int_unsigned_to_signed(val) {\n    if (val && 0x8000) {\n        val -=
0x10000;\n        val /= 10;\n    }\n    return val;\n}\n\nlet data = [];\nlet data_element =
[];\nlet temp = 0;\nfor (let idx = 0; idx < msg.payload.results[0].series[0].values.length; idx++)
{\n    temp = msg.payload.results[0].series[0].values[idx][1];\n    if (temp*10>2000){\n
temp = int_unsigned_to_signed(temp*10);\n    }\n    \n    data_element.push({ \"x\": msg.pay-
load.results[0].series[0].values[idx][0], \"y\": temp});\n}\ndata.push(data_element);\nmsg.payload
= [{\n    \"series\": [\"Temperature\"],\n    \"data\": data,\n    \"labels\": [\"\"]\n}];\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":980,"y":120,"wires":[["0e67683d525eaa00","dd99f7e340d82d3c"]]},{"id":"dcea13a
2bdae701b","type":"json","z":"8802621c69ad7966","name":"","property":"payload","ac-
tion":"","pretty":false,"x":710,"y":120,"wires":[["118a8a3be40a276d"]]},{"id":"df7721eb8af1e5bc","t
ype":"csv","z":"8802621c69ad7966","name":"","sep":",","hdrin":"","hdrout":"all","multi":"mult","ret
":"\\n","temp":"time,humidity","skip":"0","strings":false,"include_empty_strings":"","in-
clude_null_val-
ues":true,"x":770,"y":660,"wires":[["43efdc85e43f0875"]]},{"id":"43efdc85e43f0875","type":"file","z
":"8802621c69ad7966","name":"Save Humidity to .csv (Adeunis Comfort)","file-
name":"/home/pi/logs/Adeunis_Comfort_humidity_log_file.csv","filenameType":"str","appendNew-
line":true,"createDir":true,"overwriteFile":"true","encod-
ing":"none","x":1000,"y":660,"wires":[[]]},{"id":"4495708a4b40a521","type":"ui_but-
ton","z":"8802621c69ad7966","name":"","group":"5be966968ca3837c","or-
der":12,"width":0,"height":0,"passthru":false,"label":"Save Humidity (Adeunis Com-
fort)","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","top-
icType":"msg","x":150,"y":660,"wires":[["6ad7e79e5a2a1af0"]]},{"id":"6ad7e79e5a2a1af0","type":"func
tion","z":"8802621c69ad7966","name":"Prepare Humidity data for .csv (Adeunis Comfort)","func":"let
data_tmp = flow.get(\"data_comf_hum\");\nmsg.payload = [];\nfor (let idx = 0; idx <
data_tmp.length; idx++){\n    if (data_tmp[idx].y == null){\n        msg.pay-
load.push([data_tmp[idx].x, -9999]);\n    }else{\n        msg.payload.push([data_tmp[idx].x,
data_tmp[idx].y]);\n    }\n}\nreturn msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":510,"y":660,"wires":[["df7721eb8af1e5bc"]]},{"id":"5ec877a7b0bd8327","type":"
function","z":"8802621c69ad7966","name":"Save Humidity data to flow variable (Adeunis Com-
fort)","func":"flow.set(\"data_comf_hum\", msg.payload[0].data[0]);\n//return msg;","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":840,"y":620,"wires":[[]]},{"id":"ebce13a47d16c8c3","type":"func-
tion","z":"8802621c69ad7966","name":"Save Temperature data to flow variable (Adeunis Com-
fort)","func":"flow.set(\"data_comf_temp\", msg.payload[0].data[0]);\n//return msg;","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":880,"y":480,"wires":[[]]},{"id":"694a6f0e0a28f9da","type":"func-
tion","z":"8802621c69ad7966","name":"Prepare Temperature data for .csv (Adeunis Com-
fort)","func":"let data_tmp = flow.get(\"data_comf_temp\");\nmsg.payload = [];\nfor (let idx = 0;
idx < data_tmp.length; idx++) {\n    if (data_tmp[idx].y == null){\n        msg.pay-
load.push([data_tmp[idx].x, -999]);\n    }else{\n        msg.payload.push([data_tmp[idx].x,
data_tmp[idx].y]);\n    }\n}\nreturn msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":540,"y":700,"wires":[["bfe55ce4cde56ea6"]]},{"id":"bfe55ce4cde56ea6","type":"
csv","z":"8802621c69ad7966","name":"","sep":",","hdrin":"","hdrout":"all","multi":"mult","ret":"\\n
","temp":"time,temperature","skip":"0","strings":true,"include_empty_strings":"","include_null_val-
ues":"","x":810,"y":700,"wires":[["5e7f0bf2d0b19117"]]},
```


Figure 11.92: Node-RED: importable specification for InfluxDB dashboard, part 5.

```
{"id":"5e7f0bf2d0b19117","type":"file","z":"8802621c69ad7966","name":"Save Temperature to .csv
(Adeunis Comfort)","filename":"/home/pi/logs/Adeunis_Comfort_temperature_log_file.csv","file-
nameType":"str","appendNewline":true,"createDir":true,"overwriteFile":"true","encod-
ing":"none","x":1050,"y":700,"wires":[[]]},{"id":"a5eb95ea86605437","type":"ui_but-
ton","z":"8802621c69ad7966","name":"","group":"5be966968ca3837c","or-
der":6,"width":0,"height":0,"passthru":false,"label":"Save Temperature (Adeunis Com-
fort)","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","top-
icType":"msg","x":170,"y":700,"wires":[["694a6f0e0a28f9da"]]},{"id":"9c98e46dd418ced1","type":"func
tion","z":"8802621c69ad7966","name":"Save Temperature data to flow variable (Adeunis Tempera-
ture)","func":"flow.set(\"data_temp_temp\", msg.payload[0].data[0]);\n//return msg;","out-
puts":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":1030,"y":200,"wires":[[]]},{"id":"a46e63c042370b06","type":"func-
tion","z":"8802621c69ad7966","name":"Prepare Temperature data for .csv (Adeunis Tempera-
ture)","func":"let data_tmp = flow.get(\"data_temp_temp\");\nmsg.payload = [];\nfor (let idx = 0;
idx < data_tmp.length; idx++) {\n    if (data_tmp[idx].y == null) {\n        msg.pay-
load.push([data_tmp[idx].x, -999]);\n    } else {\n        msg.payload.push([data_tmp[idx].x,
data_tmp[idx].y]);\n    }\n}\nreturn msg;","outputs":1,"noerr":0,"initialize":"","final-
ize":"","libs":[],"x":590,"y":740,"wires":[["e2d3eaee55e490a0"]]},{"id":"721326f570dcfe29","type":"
ui_button","z":"8802621c69ad7966","name":"","group":"adf87389b95fb318","or-
der":5,"width":0,"height":0,"passthru":false,"label":"Save Temperature (Adeunis Tempera-
ture)","tooltip":"","color":"","bgcolor":"","className":"","icon":"","payload":"","pay-
loadType":"str","topic":"topic","top-
icType":"msg","x":180,"y":740,"wires":[["a46e63c042370b06"]]},{"id":"e2d3eaee55e490a0","type":"csv"
,"z":"8802621c69ad7966","name":"","sep":",","hdrin":"","hdrout":"all","multi":"mult","ret":"\\n","t
emp":"time,temperature","skip":"0","strings":true,"include_empty_strings":"","include_null_val-
ues":false,"x":890,"y":740,"wires":[["12e884a8786750ca","d47dd29d36e2ed3a"]]},{"id":"12e884a8786750
ca","type":"file","z":"8802621c69ad7966","name":"Save Temperature to .csv (Adeunis Tempera-
ture)","filename":"/home/pi/logs/Adeunis_Temperature_temperature_log_file.csv","file-
nameType":"str","appendNewline":true,"createDir":false,"overwriteFile":"true","encod-
ing":"none","x":1150,"y":740,"wires":[[]]},{"id":"d47dd29d36e2ed3a","type":"de-
bug","z":"8802621c69ad7966","name":"debug 16","active":true,"tosidebar":true,"con-
sole":false,"tostatus":false,"complete":"true","targetType":"full","statusVal":"","sta-
tusType":"auto","x":1040,"y":800,"wires":[]},{"id":"dd99f7e340d82d3c","type":"de-
bug","z":"8802621c69ad7966","name":"debug 17","active":true,"tosidebar":true,"con-
sole":false,"tostatus":false,"complete":"true","targetType":"full","statusVal":"","sta-
tusType":"auto","x":1240,"y":60,"wires":[]},{"id":"cc11521750f5f168","type":"tls-con-
fig","name":"","cert":"","key":"","ca":"","certname":"","keyname":"","caname":"","server-
name":"","verifyservercert":false,"alpnproto-
col":""},{"id":"adf87389b95fb318","type":"ui_group","name":"Adeunis Tempera-
ture","tab":"c1921e21877ecfd6","order":2,"disp":true,"width":"6","collapse":false,"class-
Name":""},{"id":"5be966968ca3837c","type":"ui_group","name":"Adeunis Com-
fort","tab":"c1921e21877ecfd6","order":1,"disp":true,"width":"6","collapse":false,"class-
Name":""},{"id":"c1921e21877ecfd6","type":"ui_tab","name":"USN Campus in Porsgrunn In-
fluxDB","icon":"dashboard","order":5,"disabled":false,"hidden":false}]
```


Figure 11.93: Node-RED: importable specification for InfluxDB dashboard, part 6.

Nodes “Time start” and “Time end” represents widgets for choosing dates in a calendar. These widgets are supposed to be coupled so that the time range has defined start and end. However, functionality of the nodes is such that if any of the dates is chosen, they trigger next node without waiting for the other date node. To overcome the issue, the values are combined with a special structure that allows to process them together (Figure 11.94). Both values are added to a “parts” element of msg with id 1 of 2 or 2 of 2. Then the msg is forwarded to Join-node. The latter triggers the next nodes only when both parts have arrived. This approach is identical for all parameters in the flow.

```
msg.time_start = msg.payload;
msg.parts = {
    "id" : 1,
    "index" : 1,
    "count" : 2,
    "type" : Object,
    "key" : "time"
}
return msg;
```

```
msg.time_end = msg.payload;
msg.parts = {
    "id": 1,
    "index": 2,
    "count": 2,
    "type": Object,
    "key": "time"
}
return msg;
```

Figure 11.94: Node-RED: InfluxDB: time values preparation in separate nodes for further combination in one message.

If the message is properly prepared as described above, the Join-node does not need any setup, it will function in automatic mode (Figure 11.95).

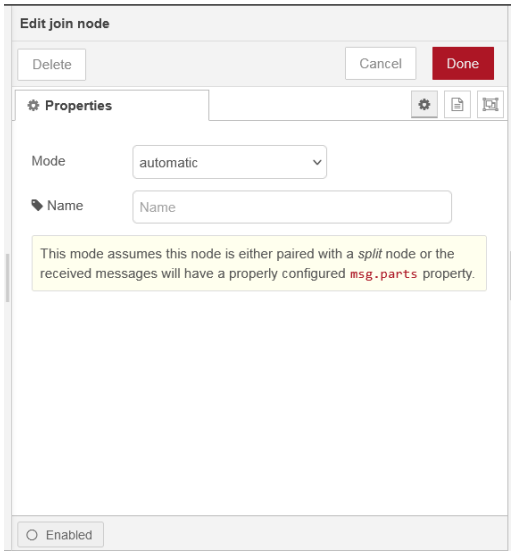


Figure 11.95: Node-RED: InfluxDB: Join-node with automatic mode.

After both values have been received, they are sent to function-node and saved into flow-variable to be accessible by other nodes of the flow (Figure 11.96).

```
flow.set("time_start_temp_temp", msg.time_start);
flow.set("time_end_temp_temp", msg.time_end);
return msg;
```

Figure 11.96: Node-RED: InfluxDB: save time into flow-variable for Adeunis Temperature sensor.

To initiate the actual reading user must activate a button, which sends message with the state id of interest in payload, as in the flow for IoT Hub.

This process includes reading of the flow-variable containing date and preparing of the URL and query for HTTP request (Figure 11.97). The query is similar to SQL with some variations, it includes state id and the time range. Additionally, it requires grouping of values

by time sample frequency, this value is hard-coded. The query is then concatenated to the URL itself and forwarded to HTTP request-node together with API-key as previously.

```
msg.time_start_temp_temp = flow.get("time_start_temp_temp");
msg.time_start_temp_temp = new Date(msg.time_start_temp_temp);
msg.time_start_temp_temp = msg.time_start_temp_temp.getTime();
msg.time_end_temp_temp = flow.get("time_end_temp_temp");
msg.time_end_temp_temp = new Date(msg.time_end_temp_temp);
msg.time_end_temp_temp = msg.time_end_temp_temp.getTime();

msg.url = "https://xapi.niota.io/xapi/v1/influxdb/query?epoch=ms&q=";
msg.query = "SELECT first(\"value_number\") FROM \"states_history\" \
WHERE(\"state_id\" = '"+ msg.payload + "') AND time >= " + msg.time_start_temp_temp + "ms and
time <= " + msg.time_end_temp_temp + "ms \
GROUP BY time(15m) fill(null)"
msg.apiKey = "82D714C9-DB85-45CA-8A95-27B824A125F7";
msg.url = msg.url + msg.query;
return msg;
```

Figure 11.97: Node-RED: InfluxDB: query preparation for temperature for Adeunis Temperature sensor.

URL and necessary headers for the following HTTP request-node will be assigned to the corresponding fields. The node must be setup to use POST method and enable SSL/TLS function (Figure 11.98).

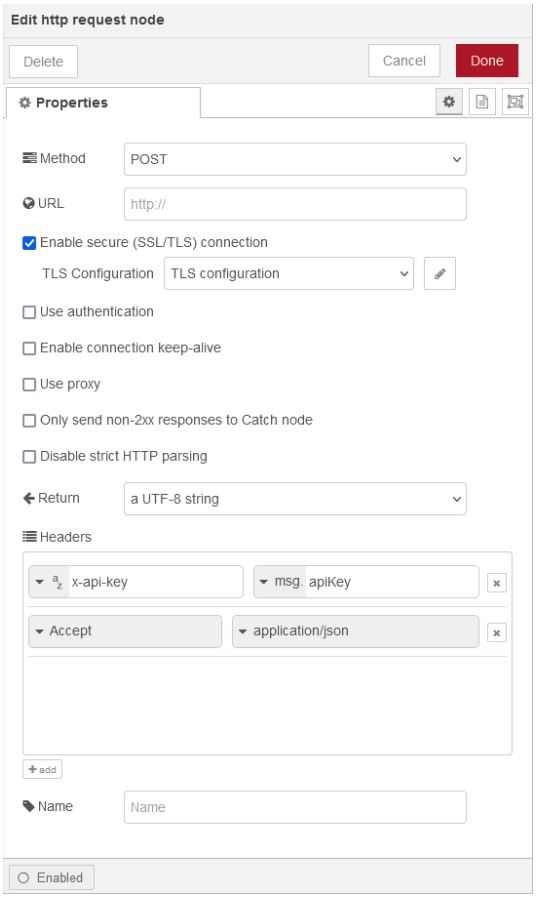


Figure 11.98: Node-RED: InfluxDB: HTTP request-node.

The response from the API is processed in a similar way as before. Nevertheless, additional functionality has been added. Measurements in the response are unsigned float values and may be larger than 6000. This is unrealistic value, and it is converted into a signed float type in a similar way as values from “payload_hex” (Figure 11.99).

```
function int_unsigned_to_signed(val) {
    if (val && 0x8000) {
        val -= 0x10000;
        val /= 10;
    }
    return val;
}

let data = [];
let data_element = [];
let temp = 0;
for (let idx = 0; idx < msg.payload.results[0].series[0].values.length; idx++) {
    temp = msg.payload.results[0].series[0].values[idx][1];
    if (temp*10>2000){
        temp = int_unsigned_to_signed(temp*10);
    }

    data_element.push({ "x": msg.payload.results[0].series[0].values[idx][0], "y": temp});
}
data.push(data_element);
msg.payload = [{
    "series": ["Temperature"],
    "data": data,
    "labels": [""]
}];
return msg;
```

Figure 11.99: Node-RED: InfluxDB: HTTP response processing from Adeunis Temparture sensor.

Data from chart is also automatically saved into flow-variable for use with logging function (Figure 11.100).

```
flow.set("data_temp_temp", msg.payload[0].data[0]);
```

Figure 11.100: Node-RED: InfluxDB: save data from chart into flow-variable for Adeunis Temperature sensor.

Saving of data is triggered by activating a button “Save Temperature” or “Save Humidity”. All the measurements plotted on the corresponding chart will be first acquired from the flow-variable and then formatted for saving (Figure 11.101). If any of the values are missing or are null, they will be replaced with -999. Similar replacement is commonly used in natural science for identifying bad data. The data for saving is compiled into an array of arrays, where each sub-array represents a row with time and measurement.

```
let data_tmp = flow.get("data_temp_temp");
msg.payload = [];
for (let idx = 0; idx < data_tmp.length; idx++) {
    if (data_tmp[idx].y == null) {
        msg.payload.push([data_tmp[idx].x, -999]);
    } else {
        msg.payload.push([data_tmp[idx].x, data_tmp[idx].y]);
    }
}
return msg;
```

Figure 11.101: Node-RED: InfluxDB: format data for saving into .csv for Adeunis Temperature sensor.

The prepared values are then forwarded to a csv-node that parses all values, adds column names and sends further as a single text in message payload to a save it to file node (Figure 11.102).

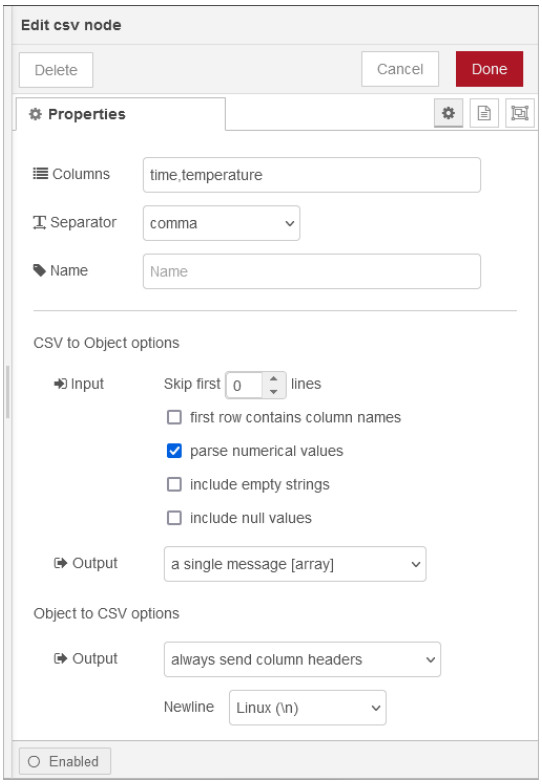


Figure 11.102: Node-RED: InfluxDB: csv node settings for temperature.

For saving data into a physical file it is required to specify an absolute path to a directory (Figure 11.103). It is possible to choose if the new data must overwrite what is already in the existing output file or to be appended to it.

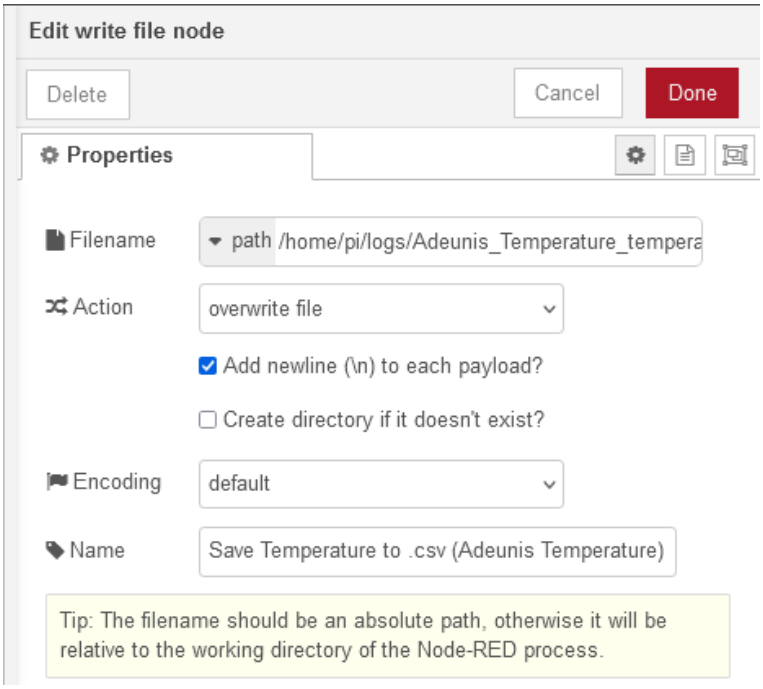


Figure 11.103: Node-RED: InfluxDB: save to file node settings.

Functions for defining start and end time for temperature and humidity components of Adeunis Comfort sensor are identical to the ones for the Adeunis Temperature sensor.

```
flow.set("time_start_comf_temp", msg.time_start);
flow.set("time_end_comf_temp", msg.time_end);
return msg;
```

Figure 11.104: Node-RED: InfluxDB: save time for temperature into flow-variable for Adeunis Comfor tsensor.

```
msg.time_start_comf_temp = flow.get("time_start_comf_temp");
msg.time_start_comf_temp = new Date(msg.time_start_comf_temp);
msg.time_start_comf_temp = msg.time_start_comf_temp.getTime();
msg.time_end_comf_temp = flow.get("time_end_comf_temp");
msg.time_end_comf_temp = new Date(msg.time_end_comf_temp);
msg.time_end_comf_temp = msg.time_end_comf_temp.getTime();
msg.url = "https://xapi.niota.io/xapi/v1/influxdb/query?epoch=ms&q=";
msg.query = "SELECT first(\"value_number\") FROM \"states_history\" \
WHERE(\"state_id\" = '"+msg.payload+"') AND time >= "+ msg.time_start_comf_temp + "ms and
time <= " + msg.time_end_comf_temp+"ms \
GROUP BY time(15m) fill(null)"
msg.apiKey = "2A463053-6194-40C8-82AE-E6988A81FAC6";
msg.url = msg.url + msg.query;
return msg;
```

Figure 11.105: Node-RED: InfluxDB: query preparation for temperature for Adeunis Comfort sensor.

```
let data = [];
let data_element = [];
for (let idx = 0; idx < msg.payload.results[0].series[0].values.length; idx++){
    data_element.push({ "x": msg.payload.results[0].series[0].values[idx][0], "y": msg.pay-
load.results[0].series[0].values[idx][1]});
}
data.push(data_element);
msg.payload = [{
    "series": ["Temperature"],
    "data": data,
    "labels": [""]
}];
return msg;
```

Figure 11.106: Node-RED: InfluxDB: HTTP response processing for temperature from Adeunis Comfort sensor.

```
flow.set("data_comf_temp", msg.payload[0].data[0]);
```

Figure 11.107: Node-RED: InfluxDB: save data from chart into flow-variable for temperature for Adeunis Comfort sensor.

```
let data_tmp = flow.get("data_comf_temp");
msg.payload = [];
for (let idx = 0; idx < data_tmp.length; idx++) {
    if (data_tmp[idx].y == null){
        msg.payload.push([data_tmp[idx].x, -999]);
    }else{
        msg.payload.push([data_tmp[idx].x, data_tmp[idx].y]);
    }
}
return msg;
```

Figure 11.108: Node-RED: InfluxDB: format data for saving into .csv for temperature for Adeunis Comfort sensor.

```
flow.set("time_start_comf_hum", msg.time_start);
flow.set("time_end_comf_hum", msg.time_end);
return msg;
```

Figure 11.109: Node-RED: InfluxDB: save time for humidity into flow-variable for Adeunis Comfor tsensor.

```
msg.time_start_comf_hum = flow.get("time_start_comf_hum");
msg.time_start_comf_hum = new Date(msg.time_start_comf_hum);
msg.time_start_comf_hum = msg.time_start_comf_hum.getTime();
msg.time_end_comf_hum = flow.get("time_end_comf_hum");
msg.time_end_comf_hum = new Date(msg.time_end_comf_hum);
msg.time_end_comf_hum = msg.time_end_comf_hum.getTime();
msg.query = "SELECT first(\"value_number\") FROM \"states_history\" \
WHERE(\"state_id\" = '"+ msg.payload + "') AND time >= " + msg.time_start_comf_hum + "ms and
time <= " + msg.time_end_comf_hum + "ms \
GROUP BY time(15m) fill(null)"

msg.apiKey = "2A463053-6194-40C8-82AE-E6988A81FAC6";
msg.url = msg.url + msg.query;
return msg;
```

Figure 11.110: Node-RED: InfluxDB: query preparation for humidity for Adeunis Comfort sensor.

```
let data = [];
let data_element = [];
for (let idx = 0; idx < msg.payload.results[0].series[0].values.length; idx++) {
    data_element.push({ "x": msg.payload.results[0].series[0].values[idx][0], "y": msg.pay-
load.results[0].series[0].values[idx][1] });
}
data.push(data_element);
msg.payload = [{
    "series": ["Humidity"],
    "data": data,
    "labels": [""]
}];
return msg;
```

Figure 11.111: Node-RED: InfluxDB: HTTP response processing for humidity from Adeunis Comfort sensor.

```
let data_tmp = flow.get("data_comf_hum");
msg.payload = [];
for (let idx = 0; idx < data_tmp.length; idx++){
    if (data_tmp[idx].y == null){
        msg.payload.push([data_tmp[idx].x, -9999]);
    }else{
        msg.payload.push([data_tmp[idx].x, data_tmp[idx].y]);
    }
}
return msg;
```

Figure 11.112: Node-RED: InfluxDB: format data for saving into .csv for humidity for Adeunis Comfort sensor.