Tuukka Talasmo

Environment measurement application for battery-operated IoT devices

Master's Thesis
in Information Technology
December 10, 2022

University of Jyväskylä

Faculty of Information Technology

Kokkola University Consortium Chydenius

Author: Tuukka Talasmo
Contact information: tuukka.talasmo@gmail.com
Title: Environment measurement application for battery-operated IoT devices
Työn nimi: Ympäristönmittaussovellus paristokäyttöisille IoT-laitteille
Project: Master's Thesis in Information Technology
Page count: 62+24
Abstract: In this thesis, a system was implemented for battery-powered IoT devices that measures environmental conditions, stores the data in a database for later analysis and displays the measurement results in a graphical user interface. The LoRaWAN protocol was used to transfer the measurement results from the devices to the IP network, but the functionality of sigfox and NB-IoT protocols was also investigated. During the implementation, the IoT-A reference architecture model was used, which provides good guidelines for design and reporting.
Suomenkielinen tiivistelmä: Tässä tutkielmassa toteutettiin paristokäyttöisille IoT-laitteille järjestelmä, joka mittaa ympäristöoloja, tallentaa tiedot tietokantaan myöhempää analysointia varten sekä näyttää mittaustulokset graaffisessa käyttöliittymässä. Mittaustulosten siirtoon laitteilta IP-verkkoon asti käytettiin LoRaWAN protokollaa mutta myös Sigfox ja NB-IoT protokollien toimivuutta tutkittiin. Toteutuksen aikana käytettiin hyväksi IoT-A referenssi arkkitehtuurimallia, joka antaa hyvät suuntaviivat suunnitteluun ja raportointiin
Keywords: IoT, LPWAN, LoRaWAN, Sensor networks, IoT-A
Avainsanat: IoT, LPWAN, LoRaWAN, Sensoriverkot, IoT-A

Glossary

3D-UNB	Triple Diversity Ultra Narrow Band
AWS	Amazon Web Services
ARM	IoT Architecture reference model
CSS	Chirp Spread Spectrum
D-BPSK	Differential Binary Phase Shift Keying
eDRX	Extended Discontinuous Reception Mode
GFSK	Gaussian Frequency Shift Keying
IoT	Internet of Things
ISM Band	Industrial, Scientific and Medical Band
LPWAN	Low-Power Wide Area Network
LTE	Long Term Evolution
M-IoT	Multimedia Internet of Things
M2M	Machine to Machine
M2H	Machine to Humans
MiH	Machine in/or Humans
MVP	Minimum Viable Product
PSM	Power saving mode
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying modulation
SaaS	Software-as-a-Service
SF	Spreading Factor
SolaaS	Solution-as-a-service

Contents

1 Introduction

Monitoring the environment and responding to changes can be crucial in order to save money and resources. If heating of a building malfunctions during winter, water pipe breaks or unauthorized person tries to gain physical access to an area with valuable artifacts, not responding immediately can be extremely costly. Unwanted events will happen and preparing for them is important. Internet of Things (IoT) gives possibility to monitor areas of interest and notify users when needed. IoT applications can also take action without need for human interaction which makes them very cost efficient.

United Nations defined 17 sustainable development goals in 2015 to bring peace and prosperity for the people and the planet[49]. Increasing energy consumption requires more and more from the planet. Sustainable development goals include two goals that are clearly affected by energy consumption. Affordable and clean energy goal states that progress in energy efficiency needs to speed up to achieve global climate goals. Climate action goal states that energy related CO^2 emissions have reached the highest level ever in 2021 and will the emissions need to be reduced in order to stop the rising of global temperatures.

Improving energy efficiency of buildings is the main motivation for this thesis. Summer and winter seasons affect buildings differently and the energy efficiency can be improved with better building insulation but it is also important to consider airflow to prevent bad air quality. Therefore, it is important that we constantly measure the environment to understand the places were the energy efficiency can be improved as shown in Figure 1.1.

Some companies offer temperature and humidity mapping services, for example Vaisala which manufactures environment measurement instruments[50]. With mapping services, customers want to understand the current state of their buildings and validate if defined requirements are met and act if not. These measurements are usual done only from few weeks to months, which might be sufficient for their pur-

pose but can be expensive. Ordinary people need a cheap solution to measure their homes for throughout the year, which is the goal of this thesis.

Good insulation prevents heat leaking from buildings during cold seasons but can cause high relative humidity to build up during warm seasons. Constant high relative humidity can cause molding in buildings. Therefore, it is important to measure both temperature and relative humidity to make best possible decisions on how to improve the energy efficiency of buildings. Measuring CO^2 can be used to monitor the air quality but accurate sensors can be expensive, therefore CO^2 measurement is left out of this thesis.

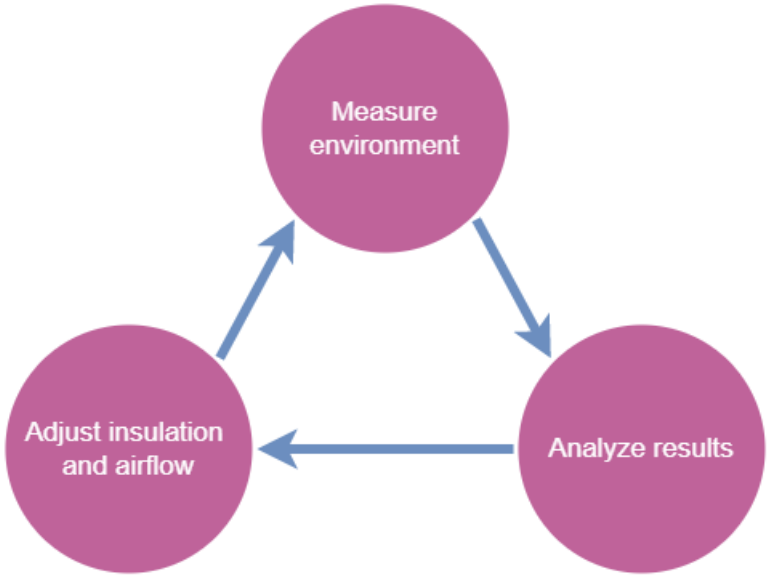

Figure 1.1: Steps to constantly improve energy efficiency of buildings

In this thesis, cost-efficient environment measurement application using low-power wide area network (LPWAN) technology was developed. First, introduction to Internet of Things is given in Chapter 2 and then considered LPWAN technologies are introduced and compared in Chapter 3. Methods used in the application design process were divided into Four Stage IoT Architecture Model introduced in Section 4.1 and IoT Architecture reference model (ARM) introduced in Section 4.2.

Details how the application development was done, are described in Chapter 5. Requirements are set in Section 5.1 and design decisions to fulfil the requirements are described in Section 5.2. The architecture of developed architecture is presented

in Section 5.3.

The application performance was tested in Tokyo, Japan and the test results are shown in Section 5.4 which also includes the evaluation against the requirements. Finally, conclusion of the whole thesis and future development needs are given in Chapter 6.

2 Internet of Things

Internet of Things (IoT) is made of small and big devices that are connected to the internet and are interacting with each other directly or via humans[9, p.1]. These devices can be for example RFID tags, cameras or displays and have various different use cases such as sensing the environment, security of closed areas, protecting people, keeping users informed and automatic control of actuators. IoT can be categorized to three types of communication applications: Machine to Machine (M2M), Machine to Humans (M2H) and Machine in/or Humans (HiM)[32, p.2].

Some applications require high bandwidth and high Quality of Service (QoS) and IoT devices are usually restricted with resources such as battery, memory and computational power[32, p.2]. High bandwidth requirements of multimedia applications such as audio and video streaming services, restricts the communication protocols that can be used. The variety of IoT applications means that a single data transfer technology cannot meet requirements of every IoT application. This thesis concentrates on technologies that are used for battery operated and low-data rate applications, therefore not suitable for most multimedia applications.

2.1 IoT Market

According to IoT market research done by Transforma Insights, the global market for IoT will grow to $1.5trn annual revenue by year 2030[31]. There were already 7.6 billion active devices by the end of 2019 and is expected to grow to 24.1 billion devices by 2030. Most of the devices are in the consumer sector and use short-ranged communication technologies like Wi-Fi, Bluetooth and ZigBee. Some of the use cases for enterprises that can use long-range communication technologies are monitoring power grids, water supplies and asset tracking.

IoT is used in agriculture as well where maximizing the growth of crops is the goal. Remote controlled drones and sensors are be used to gather data from the fields and then the data is processed using machine learning. Farmers get essential informa-

tion that helps them to make correct decisions. According to a research concentrating on agricultural IoT market done by Meticulous Research in 2021, the market will be worth $32.75 billion by 2027[28].

Microsoft and Hypothesis published IoT Signals research in October 2021, where they interviewed more than 3000 IoT professionals around the world[30, p. 57]. The main objective of the research was to understand the current and future state of IoT. COVID-19 pandemic in 2021 increased the amount of investments towards IoT according to the research[30, p. 24]. 44% of the organizations responded that they will invest more to IoT in 2021, compared to 31% in 2020. The use of IoT also brought challenges to the organizations. Technical complexity and need for business transformation were seen as the biggest challenges[30, p. 16]. The most common stated reasons for IoT proof of concept projects to fail were high cost of scaling with 32% and lack of necessary technology with 26%[30, p. 18].

2.2 Existing IoT services

There are currently various IoT services offered for private and enterprise use[17]. These services can be for example Solution-as-a-Service (SolaaS), where everything from measuring nodes to presenting the data is included in the service. Software-as-a-Service (SaaS) is a service model where the service provider takes care of software applications running in a cloud server but customers have to provide the data and use the defined communication protocols. The IoT service examples presented next uses these two service models.

Kiwi Technologies offers cloud based software-as-a-service[20]. Customers can buy various Kiwi products, set up a private LoRaWAN gateway and send data to public or private cloud servers with Kiwi's AIOT SaaS Cloud Platform installed. Available products include temperature and humidity instruments as well as I/O LoRaWAN to RS485/RS232 converters that can be used to add LoRaWAN connectivity to cloud non-Kiwi products. The accuracy of the instruments offered by Kiwi Technologies might not be enough for some applications, therefore scaling of an application might be expensive when each more accurate measurement instrument needs to be connected to an I/O LoRaWAN to RS485/RS232 converter. AIOT SaaS Cloud Platform supports MQTT and RESTful communication protocols which allows integration of

customers' own applications.

Vaisala's solution-as-a-service consists of instruments that measure the environment and a cloud or private server to present the data using viewLinc[51] web application. Vaisala's measurement instruments use various communication technologies to transfer measured values to a viewLinc server. Measurement instruments using LoRa need a LoRa gateway to forward data to the viewLinc server, where as instruments using Wi-Fi or Ethernet can directly connect to the server by using the IP network. The system only works with Vaisala measurement instruments and other manufacturer's instruments cannot be added without using analog converters. However, non-Vaisala systems can connect to a viewLinc server by using viewLinc Application Programming Interface.

Microsoft's cloud service Azure has IoT hub[29]. Customers can connect nodes to IoT hub using HTTPS, AMQP and MQTT protocols. Nodes can be managed using Azure IoT hub. For example, it is possible to update firmware of connected nodes over-the-air. Pricing is dependent on the amount of nodes connected and each operation of an node. Azure IoT hub does not provide communication technology to transfer data from network nodes to the cloud, for example LPWAN technology. Network nodes need to be connected to the internet and be able to use the previously mentioned communication protocols or a gateway with the same capabilities need to be a translator between the network nodes and the cloud.

Amazon's AWS IoT Core is a similar IoT service as Microsoft's Azure IoT hub[4]. AWS IoT Core supports the same communication protocols HTTPS, AMQP and MQTT. In addition, it supports directly connected LoRaWAN gateways which simplifies the network architecture compared to Azure IoT hub, if LoRaWAN is used as the communication technology for the connected nodes. AWS IoT Device Management is a web application to manage the fleet of connected nodes. It also allows over-the-air firmware update, same as Azure IoT hub. Pricing of AWS IoT Core consist of bulk adding/removing nodes, search actions for the exists fleet and a fee of each remote action per node.

AskSensors is a light weight IoT platform compared to Azure IoT hub and AWS IoT Core[8]. It does not have same AI processing capabilities as the other two mentioned

services and it supports only up to 60 nodes per a customer. Data can be saved to the askSensors cloud by using HTTP, HTTPS or MQTT. Measured data from nodes can be shown as graphs or askSensors can be used to control actuators. AskSensors get API for retrieving data supports up to 50 latest measurements, therefore it cannot be used for long term processing and evaluation of data. Pricing is monthly based and depends on the amount of connected nodes.

3 Low-Power Wide Area Network

Low-Power Wide Area Network (LPWAN) technologies aim to support wast amount of battery operated low-cost, low data-rate devices than can be in use for years without human interaction[14, p.3]. The technologies allow multiple-kilometer range with a cost of data bandwidth.

LPWAN end-devices are not directly connected to the internet but need to have a gateway capable of transmitting and receiving data using the specific LPWAN technology and also forward data between the LPWAN end-devices and the internet.

Various LPWAN technologies have been developed[2, p. 1]. Ingenu is a technology developed in the 2.4GHz band with high-data rate but also high power consumption and short range of 5-6km, compared to other LPWAN technologies. Weightless-W, Weightless-N and Weightless-P open-source technologies that were developed by The Weightless Special Interest Group. A part from Weightless-W using 470-790 MHz for two-way data transfer and Weightless-N using 800-900 Mhz band for one-way data transfer, the differences with the two standards are in data-rate, battery lifetime and range. Weightless-P is designed to be improvement over the previous two standards using multiple bands from 169 MHz to 923 MHz.

Three LPWAN technologies were considered in this thesis and those are LoRaWAN, sigfox and NB-IoT. The technologies will be introduced next.

3.1 LoRaWAN

LoRAWAN is a LPWAN technology that is being developed and maintained by the LoRa Alliance. LoRa Alliance has big companies as members such as Cisco, Microsoft and AMAZON Web Services[24, p. 42]. LoRaWAN is a communication layer on top of LoRa physical layer. LoRa was developed by Cycleo which was later acquired by Semtech. LoRaWAN offers long battery lifetime of 10 years and long communication range of 2-5 km in urban areas and 15 km in suburban areas[2, p. 2].

Used network topology is star-of-stars, in which end-devices send data wirelessly with a single hop to gateways connected to the internet.

The physical layer LoRa uses Chirp Spread Spectrum (CSS) modulation in which frequency the signal is modulated in short chirp pulses[2]. This increases the robustness compared to other modulation methods such as amplitude modulation. Used Industrial, Scientific and Medical (ISM) band varies depending on the region where LoRaWAN is being used. For example, 433 MHz and 868 MHz are used in the EU and 915 MHz in the Americas[14, p. 5]. To increase the communication range with the cost of data-rate, different Spreading Factors (SF) can be used. There are six possible SF values from 7 to 12[2, p. 2]. SF value is calculated using symbol rate(R_s) and chip rate(R_c)[2]. The equation for this is shown in Equation 3.1.

$$SF = log_2 \frac{R_c}{R_s} \tag{3.1}$$

As seen in Table 3.1, increasing the spreading factor increases the number of chips used for sending a single symbol. This means that the time on air for a packet increases and bit rate decreases. However, increasing the spreading factor improves the receiver sensitivity, therefore the communication range is also increased. When end-devices are using different spreading factors, they are able to send data at the same time due to the spreading factors being orthogonal to each other[53, p. 3].

LoRaWAN protocol does not have direct restriction on how many times per day can a message be sent but duty-cycle is limited[23, p. 49]. Local regulations might also limit duty-cycles and LoRaWAN devices need to follow the limits whichever is more constraining. Due to the duty-cycle limitations and how the spreading factor affects the time on air, it is possible to send 20(SF12) to 500(SF7) messages per day[47].

3.1.1 LoRaWAN end-device classes

LoRaWAN end-devices are categorized to three classes: A, B and C[23, p. 7]. Class A end-devices include all the features that are required for a LoRaWAN end-device and class B and C end-devices should implement the features as well[23, p. 11]. Class A, B and C differ from each other by the length of receiving windows. Class A end-devices only listen for incoming data for a short while after sending data as seen in Figure 3.1. Class B end-devices have fixed time slots for receiving to decrease

Table 3.1: Spreading factor effect on bit rate[35].

Spreading Factor	**Chips/Symbol**	**Time on Air for 10 byte packet (ms)**	**Bit Rate (bps)**
7	128	56	5470
8	256	103	3125
9	512	205	1758
10	1024	371	977
11	2048	741	537
12	4096	1483	293

the latency[23, p. 51]. Class C end-devices are continuously listening for incoming messages unless transmitting[23, p. 75]. Since increasing the listening time of an end-device increases the power consumption, the class should be selected to fit the needs of an application.

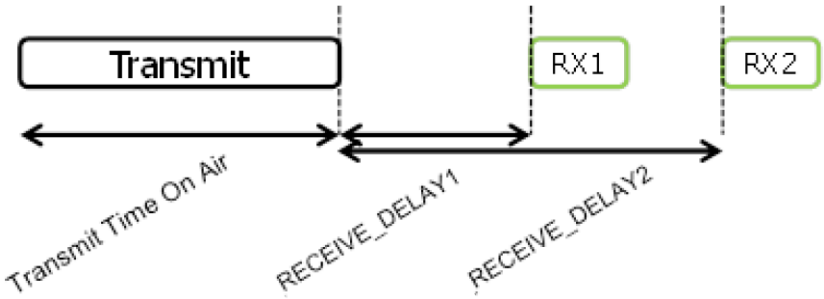

Figure 3.1: Receive windows RX1 and RX2 after uplink transmission [23, p. 12].

3.1.2 Existing LoRaWAN use cases

LoRaWAN technology is already widely used. Uses cases listed by the LoRa Alliance stands at more than 140 at the end of 2021[24, p. 20].

The things Network is a public LoRaWAN network, operated by The Things Industries, that anyone can join and add nodes or gateways to expand the coverage[48]. The coverage varies greatly depending on the region. For example at the start of the

year 2022, a big city like Tokyo had only two gateways [46]. The Things Network allows direct integration of other services like AWS IoT Core, LoRa Cloud or any other service with the use of MQTT server[48].

Yokogawa has created sensor named Sushi Sensor, shown in Figure 3.2, which monitors signs of equipment abnormalities[24, p. 20]. With the help of machine learning, it can prevent unexpected equipment failures by identifying abnormalities and notifying human operators.

In India, SenRa and McWane are deploying 10,000 water meters using LoRaWAN network to monitor water consumption and wastage in urban areas[24, p. 20]. This is to improve problems related to water cycle management. They have plans to expand to Middle East, Asia and Africa within the next five years.

In U.S and Canada, Machine Q and CoreKinet designed and implemented LoRaWAN inventory management application with tracking of 350,000 high-value assets[24, p. 20]. This enables the users to optimize their workflow by having real-time inventory and location information of their assets.

Figure 3.2: Yokogawa's Sushi Sensor for monitoring equipment [24, p. 20].

3.2 Sigfox

Sigfox is the name of the LPWAN technology as well as the network operator[41, p. 4]. Sigfox 0G network operates over 72 countries and has over 18 million connected devices[40, p. 4]. In contrast to LoRaWAN where users can freely create new LoRaWAN networks, Sigfox users are required to use the network provided by Sigfox 0G network. Architecture of Sigfox 0G network can be seen in Figure 3.3. Connected objects can be any Sigfox certified devices. Data from the devices is sent to Sigfox cloud via radio hubs that are operated by Sigfox. Data is then routed to a user's own application server.

Sigfox communication rules are called Triple Diversity Ultra Narrow Band (3D-UNB)[41, p. 4]. 3D-UNB implements a differential binary phase shift keying (D-BPSK) modulation for uplink messages[41, p. 9] and Gaussian Frequency Shift Keying (GFSK) modulation for downlink messages[41, p. 11]. Same as LoRaWAN, Sigfox operates in the license-free ISM bands (ex. 433 MHz and 868 MHz in Europe) which differ in each country[41, p. 5]. Sigfox has better range than LoRaWAN and NB-IoT with with one radio hub covering over 40 km[27, p. 5].

Each data payload frame is transmitted 3 times with different frequencies[41, p. 27], as seen in Figure 3.4. Uplink frame #1 (UL Frame #1) is duplicated and transmitted with higher frequency of ΔF_{MF} and then with lower frequency of ΔF_{MF}. Value of ΔF_{MF} is dependent on the region where the technology is being used. Duplicating frames increases the energy consumption required to transmit data but it also improves the quality of service. Sigfox has an option to receive downlink frames after transmission which also different from the previous frequencies.

3.2.1 Existing Sigfox use cases

Sigfox has various use cases listed on their web-page[39]. Uses case examples include supply chain logistics, retail, smart cities and waste management.

Utility management such as monitoring water, gas and energy consumption is labor intensive, if done manually. For remote online monitoring, Laager developed a RF module that can be installed to existing manual water, gas and energy meters[38]. Data is collected and send to Data Management Control Center. Operators can then

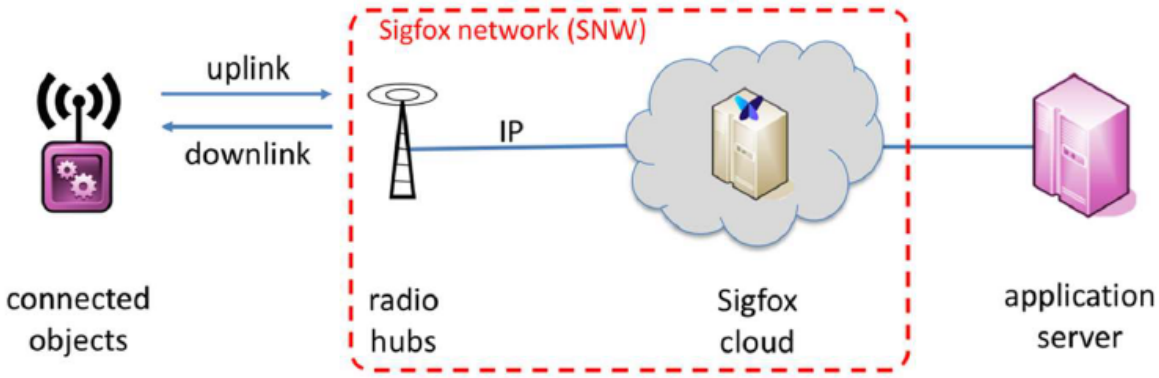

Figure 3.3: Sigfox network architecture [41, p. 4].

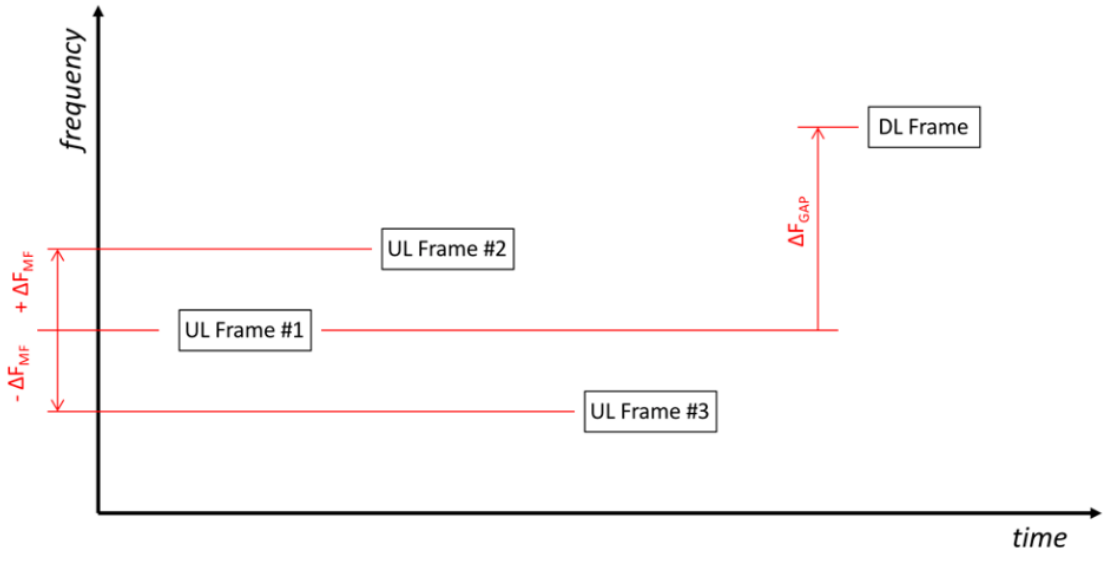

Figure 3.4: Sigfox frequency changes[41, p. 27].

get up to date reports from the management application.

KS Technologies created a waste management system for a amusement park chain with multiple parks[37]. Park operators are monitoring the fill level of their trash bins. The waste management system helps them to keep the parks clean by enabling them to empty the trash bins only when required.

Sigfox 0G network is also used for animal conservation. "Now Rhinos Speak" project, which received e-Environment price at WSIS 2019 forum, embedded low-cost trackers to rhinos' horns to send their location to rangers[40, p. 3]. The rangers could then use the information to get a mobile map of the rhinos.

3.3 Narrow Band IoT

Narrow Band IoT (NB-IoT) is an extension to Long Term Evolution (LTE) used in the cellular networks[26, p. 1]. NB-IoT is developed by the 3rd Generation Partnership Project (3GPP) which unites seven telecommunication standard development organizations around the world[1]. NB-IoT uses the same base stations as LTE, therefore it is already available in many parts of the world[27, p. 5]. In contrast to mobile phones that use LTE, NB-IoT devices are meant to be stationary, delay-tolerant and send smaller amount of data[26, p. 1].

NB-IoT operates in licensed frequency bands using Quadrature Phase Shift Keying modulation (QPSK) and with synchronous protocol, which gives it better Quality of Service (QoS) compared to LoRaWAN and Sigfox[27, p. 4]. However, licensing fees and shorter range of less than 10 km per a base station, increases the costs of scaling NB-IoT networks.

Simplified NB-IoT transmit cycle is shown in Figure 3.5. QoS parameters in NB-IoT can be modified based of the needs of an application, which affects the lengths of Extended Discontinuous Reception Mode (eDRX) and Power Saving Mode (PSM) cycles. During paging time window which can be between 2.56 s to 40.96 s, a node is listening for downlink messages [26, p. 2]. Activity timer T3324 defines how many eDRX cycles are done before the node enters PSM state. In PSM state, radio is turned off to conserve energy which can be up to 413 days.

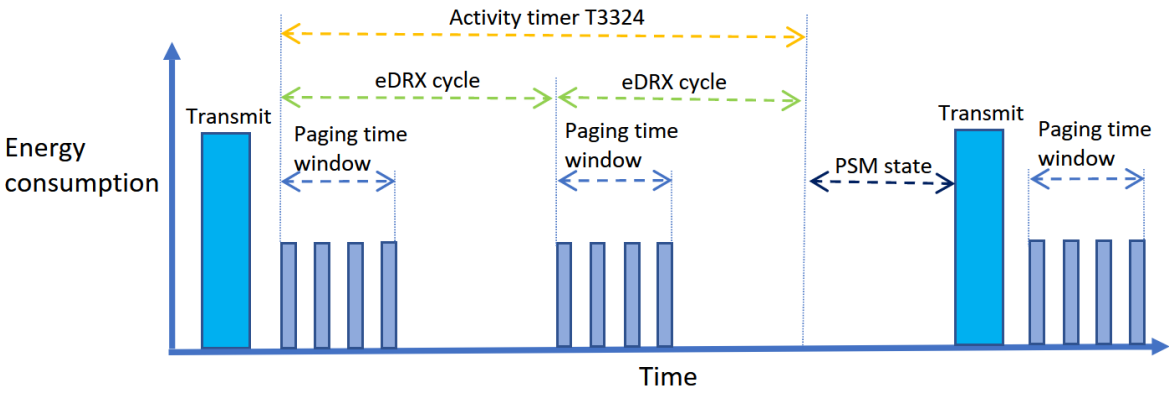

Figure 3.5: Simplified NB-IoT transmit cycle [26, p. 3]

3.3.1 Existing NB-IoT uses cases

NB-IoT is still relatively new compared to LoRaWAN and Sigfox but there are some demonstrations of using the technology.

In Denmark, solution for managing marina spots in harbours was developed[25]. Sensors are used to monitor free marina spots to inform approaching ships where they could land. As shown in Figure 3.6, ships can use a phone application that connects to the server managing the harbour sensor information. The server periodically checks the status of sensors by connecting to them using NB-IoT.

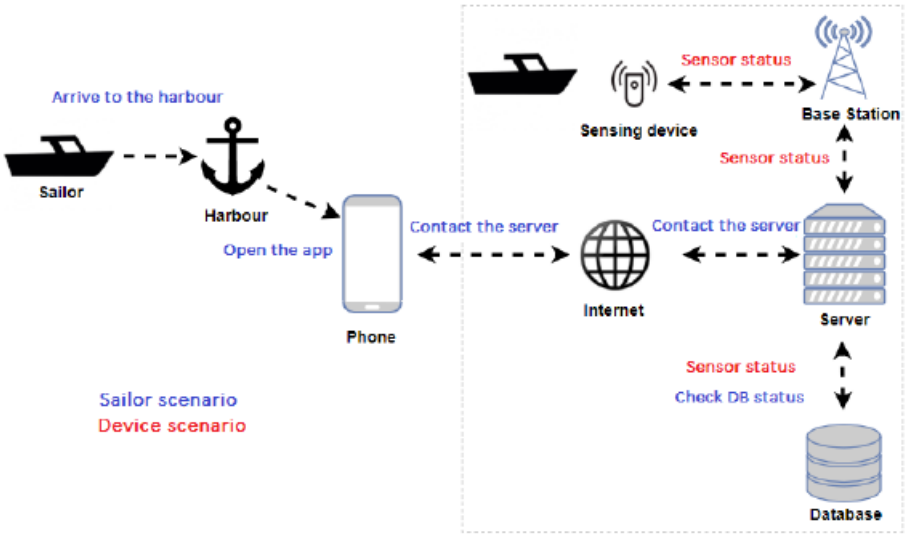

Figure 3.6: Maritime demonstration in harbour using NB-IoT[25, p. 2].

In China, researchers created a self-powered remote fire monitoring application for places that are not easily accessible[16]. The solution has temperature, pressure and carbon monoxide sensors to detect fire and NB-IoT module to transfer the data to a cloud server[16, p. 6]. Instead of using batteries, wind is harvested to give enough power for the solution. If a fire occurs, the people nearby can get a notification from the application as shown in Figure 3.7.

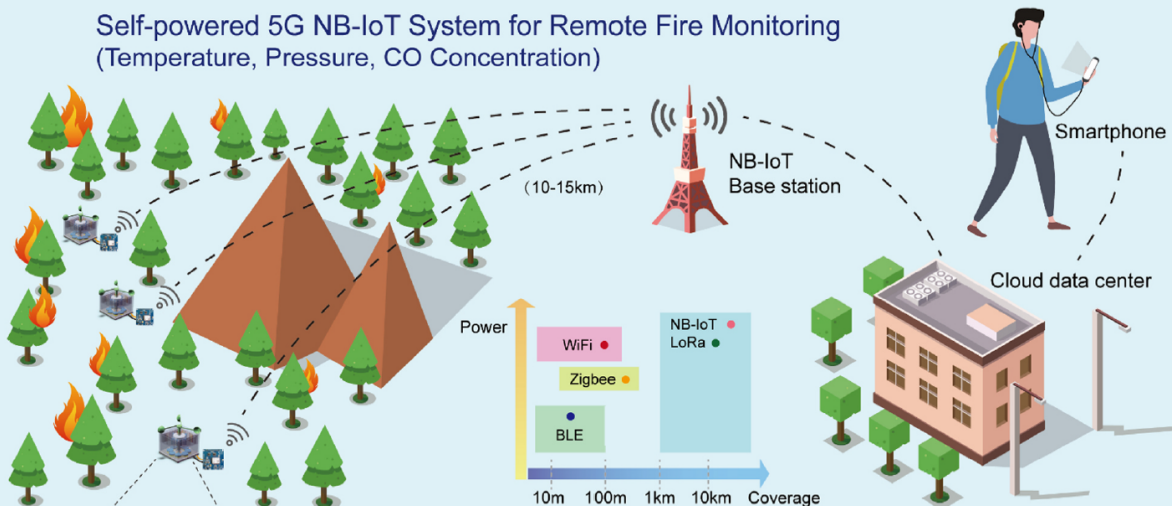

Figure 3.7: Self-powered 5G NB-IoT Remote Fire Monitoring application[16, p. 2].

NB-IoT is also used in urban areas of China[15, p. 9]. China Unicom is deploying 170,000 smoke and toxic gas detectors using NB-IoT to rental rooms in the Yuhang district of Hangzhou. In case of fire or a toxic gas leak, fire fighters can get real-time information in addition to the application automatically warning nearby residents. Smoke levels, power consumption and network signal strength from the devices are sent periodically to the maintenance platform so that maintenance personnel can take action if a device malfunctions or needs to have its battery replaced.

3.4 Power consumption of LPWAN technologies

As shown in the previous existing use cases sections, battery operated IoT nodes are often deployed in hard to reach places which makes maintaining them difficult. Therefore, it is essential to minimize the power consumption using communication technologies that meets the application requirements. Hardware and software design also play a crucial role in power consumption. Callebaut et. al demonstrated how these three design aspects effects the overall battery life[11]. In the demonstration, a LoRaWAN node was used to send 16 bytes payload packets with SF12 every hour and power consumption and total energy consumption were measured. As seen in Figure 3.8, wireless communication drains the battery more than the other stages. However, Figure 3.9 shows that the overall energy consumption is greatly effected by the sleep stage. This is because the node spends most of it's time in sleep stage and wakes up only when it is necessary.

Singh et. al[42] measured the power consumption of LoRaWAN, Sigfox and NB-IoT nodes with specific settings. The results are shown in Table 3.2. Fixed packet length of 5 bytes was used and nodes were powered with voltage level 3.7V. LoRaWAN node was configured to use SF9. Increasing the scaling factor to 12 would increase the needed transmission time and therefore increase the power consumption. NB-IoT measurements were done with eDRX set as 31s which is the time a node waits for downlink messages from a server after each transmission. NB-IoT node goes to sleep mode after eDRX time is over. Sigfox settings cannot be modified, therefore default settings were used.

The results show that LoRaWAN with SF9 has the lowest power consumption when measurements are sent in 15 minutes intervals. Sigfox node had the lowest power

consumption in sleep mode but higher power consumption compared to LoRaWAN when transmitting. Sigfox sends each packet 3 times with different frequencies which increases the robustness of communication but consumes more energy[11, p. 6]. NB-IoT was measured to have the highest overall power consumption but actual power consumption is highly dependent of eDRX value. Estimating power consumption of different settings can be done but actual measurements are needed to get accurate results due to imperfections of hardware and software. For this thesis, the effect of eDRX when using the lowest possible value of 2.56s was calculated to show the estimated effect on the power consumption. NB-IoT protocol with eDRX value of 2.56s would have lower overall power consumption than Sigfox but still higher than LoRaWAN.

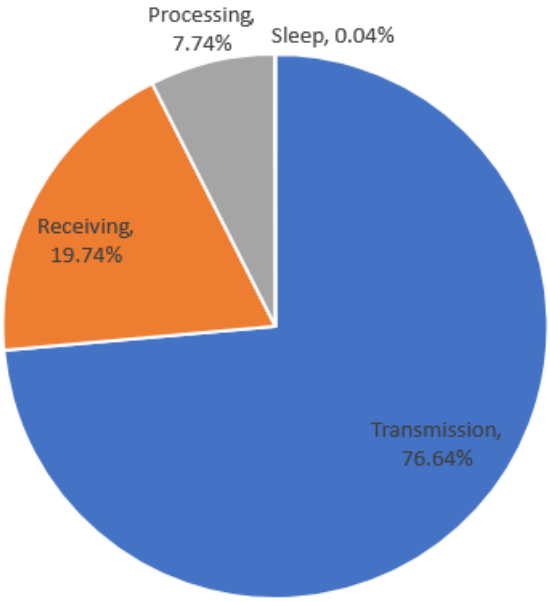

Figure 3.8: Power consumption of different stages in a LoRaWAN node[11, p. 4].

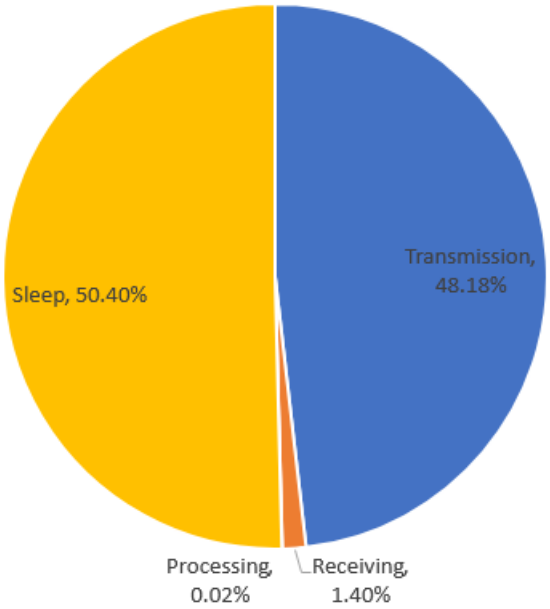

Figure 3.9: Total energy consumption of different stages in a LoRaWAN node[11, p. 4].

Table 3.2: Energy consumption of different protocols with 15 minutes intervals and 5 bytes payload[42].

	LoRaWAN (SF9)	**Sigfox**	**NB-IoT (eDRX=31.0s)**	**NB-IoT (eDRX=2.56s)**
Energy per transmission (J)	0.04	0.98	1.15	0.15
Transmission time (ms)	255.89	5396.00	373.00	373.00
Active state energy use per day (J)	4.22	94.53	110.70	14.81
Sleep state energy use per day (J)	25.83	0.02	30.85	31.86
Total energy (J) per day	30.04	94.55	141.55	46.67

4 Application design

4.1 Four stage IoT architecture model

Multiple IoT architecture models have been created. For this thesis, four stage architecture model is used which is a mix of models presented by Alec Jahnke in The 4 Stages of IoT Architecture article[19](model A) and Preeti Agarwal et al. in Open Service Platforms for IoT article[3](model B). Comparison table of the two models is shown in Table 4.1. Both models divide the architecture to four layers but the differ slightly on the layer 3 and 4. Layer 3 of model A focuses on the pre-processing of data where as model B does more deeper analysis and provides APIs for application developers. In the layer 4, model A does the in-depth data analysis and provides the required services. However, model B providing services and user interfaces of already processed data. The architecture model used in this thesis is shown in Figure 4.1 and explained in the next sections.

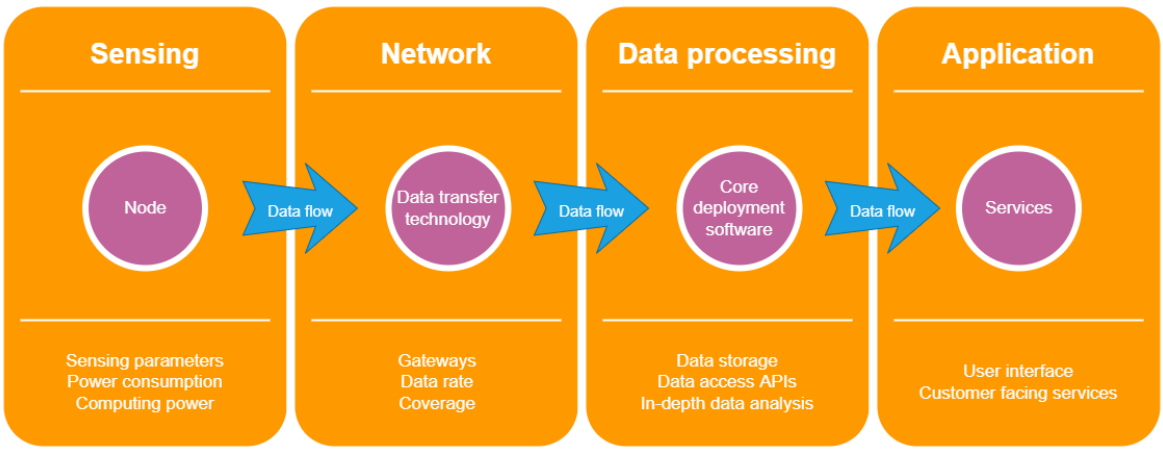

Figure 4.1: Four stage Internet of Things architecture

Table 4.1: Comparison of IoT architecture models by Alec Jahnke[19] (model A) and by Preeti Agarwal et al.[3] (model B).

Layer	The 4 Stages of IoT Architecture by Alec Jahnke	IoT Reference Architecture by Preeti Agarwal et al.
1	**Sensors and Actuators** - Actual hardware that operates in the environment	**Sensor Layer** - Actual hardware that operates in the environment
2	**Internet Gateways and Data Acquisition Systems** - Technology to transfer data to the next stages	**Gateway and Network Layer** - Technology to transfer data to the next stages
3	**Analytics at the Edge** - Preprocessing data before transferring it to final storage and analysis	**Service Management Layer** - API to retrieve data and analyze it
4	**In-depth Analysis in the Cloud or Data-center** - Storage, in-depth analysis and user interface	**Application Layer** - User interface

4.1.1 Sensing layer

The sensing layer is where raw data is collected. Raw data can be for example measuring environmental parameters like temperature, humidity and pressure or tracking location of assets. Depending on the application, response time, measurement accuracy and measurement interval requirements changes. Application might require that a sensing node is battery operated to keep deployment costs low. Some applications might require high measurement accuracy, short measurement interval and fast response time which increases the power consumption and requires the node to be connected to a power source.

4.1.2 Network layer

Acquired data is transferred from nodes to servers in the network layer. Requirements of an application greatly effect the technology used to transfer the data. Low-power wide-area network(LPWAN) technologies can be used if the application has low data rate and does not require fast response time. High data rate applications require a technology with enough bandwidth, for example 5G or different Wi-Fi

technologies shown in Figure 4.2 within the "Cellular" box. Range capability of an technology effects the overall cost of network deployment due to the amount of base stations needed to create enough network coverage. Therefore, LPWAN technologies shown in Figure 4.2 within the "LPWA" box have the lowest network deployment cost.

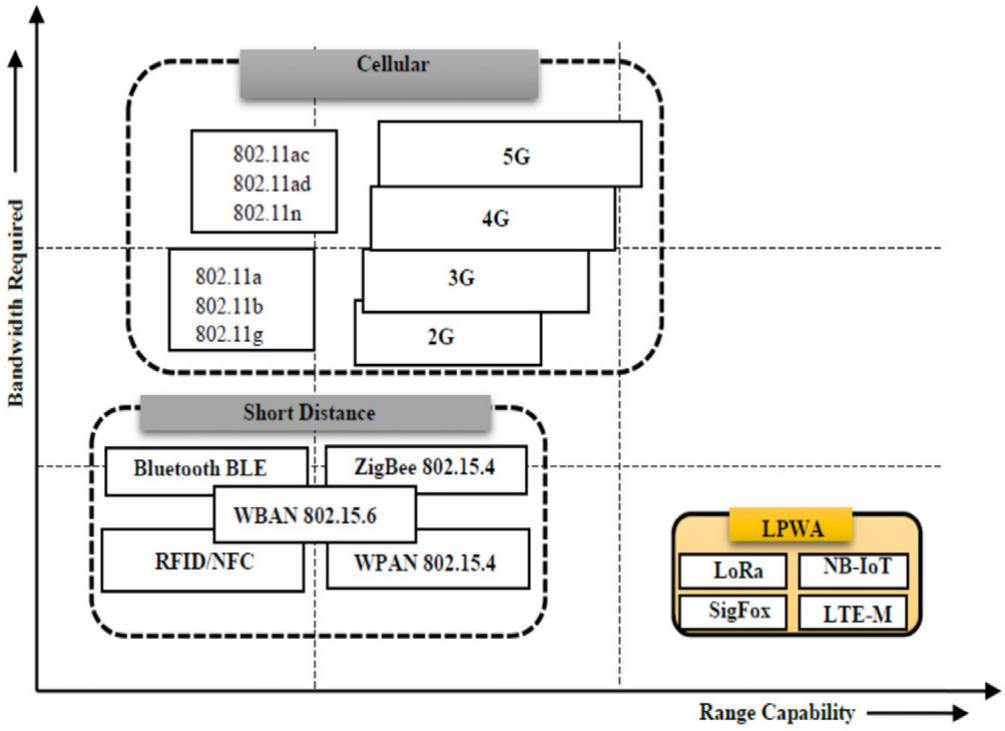

Figure 4.2: Data transfer bandwidth and range of different technologies [43, p. 2]

4.1.3 Data processing layer

Raw data can be pre-processed before sending it to a server for data storage and in-depth data analysis. This is called edge computing and it can be already done in sensing nodes or in gateways that forward data from nearby nodes to servers. Edge computing allows faster response time than in-depth data analysis on the server side. It is also used to reduce the amount of raw data that needs to be transferred and stored.

The cost of a single node is a critical part of an IoT application. As it was stated in IoT Signals research by Microsoft and Hypothesis, high cost of scaling was the

most common reason for IoT projects to fail[30, p. 24]. Therefore to keep the costs down, nodes tend to have the minimum required capabilities and this includes computing power which is needed for edge computing. The amount of gateways is far less than nodes and do not usually have power consumption restrictions. Gateways can also have high computing power, which makes them a good choice for edge computing.

Data storage and data processing can be done centralized or decentralized way. Centralized data storage can be a cloud server or a physical server. Where as decentralized data storage can be that the data is distributed between multiple servers or like in blockchain networks, stored on multiple consistent nodes in the network[22, p. 7]. Centralized data storage and processing is easier to maintain than decentralized data storage and processing but there is a bigger risk for data unavailability due to a single point of failure.

Lockl et al.[22] created design principles for blockchain-based IoT applications. The design principles where demonstrated with a application that saves environmental measurements done by Raspberry Pi computer to a private Ethereum blockchain. The application was evaluated by experts and they concluded that it was superior to non-blockchain-based applications in terms of data availability[22, p. 24]. However, the experts also concluded that high operational cost approximately $ 56.16 per hour in public Ethereum blockchain for a single blockchain node was a major disadvantage of the application and could limit the potential use cases[22, p. 25]. Also, data in public blockchain applications is transparent for all users, therefore organizations that require their data to be confidential should refrain from using them[22, p. 28]. It is possible to create private blockchain application to reduce operational costs and keep data within certain organization but private blockchain applications has far less nodes than public ones, which creates a security risk. If a blockchain application has only few nodes, one node with unproportional large processing power could change state of the blockchain application, so called 51% attack[22, p. 31].

4.1.4 Application layer

Application layer is the final layer of the four-stage architecture model and it includes all the required services that are shown to users. Data can be further analyzed in the application layer or shown as it is. Analyzed data can be shown as graphs for

human operators to enable data driven business decisions or used to control devices without need for human interaction. Data can be collected and analyzed but if it is not presented in a way that it can be used effectively, the value of the whole application becomes questionable.

4.2 IoT architecture reference model

IoT Architecture reference model (ARM) was released in 2013 as the result of the European Lighthouse Integrated Project IoT-A[10, p. 14]. IoT-A project was a collaboration of scientists from more than 20 companies. The goal was to improve the current state of IoT development where IoT solutions are created without common set of guidelines, which hinders the scalability and interoperability of the solutions. IoT ARM framework provides a structure and guidelines to develop, use and analyze IoT solutions. This helps developers and stakeholders to speak the same language.

IoT ARM defines architectural views and perspectives, which can be used to create concrete IoT architectures[10, p. 164-165]. Architectural views are used to provide views of the architecture of an application that are conceptionally isolated. Since there are design aspects that needs to be considered throughout the whole application, for example information security, architectural perspective gives a collection of activities, tactics and guidelines to address these aspects.

Architectural views are divided to

- Functional View
- Information View
- Deployment and Operation View

Architectural perspectives are divided to

- Evolution and Interoperability
- Performance and Scalability
- Information Security

- Availability and Resilience

The goal of this the thesis is not to develop an full solution for a known issue but to create a platform that is modular and can be expanded to fit the needs of various solutions. Therefore, only parts of IoT ARM framework that are within the scope of this thesis are considered.

4.2.1 Functional view

Functional view of an IoT application, divides the functional components but does not define the interactions between the components due to high number of possibilities[10, p. 166]. The functional view is shown in Figure 4.3. Application and Device functional groups are out-of-scope of IoT ARM. Management and Security functional groups interact with all other groups where as other groups interact only with the closest ones. Functional groups and their purposes are defined in Table 4.2.

Table 4.2: Functional groups in Functional views[10, p. 168-183]

Functional group	**Purpose**
Management	Combines functions that are needed to manage an IoT application
Service organization	Communication hub between other functional groups using notification services
IoT process management	Provides concepts and interfaces to augment business processes to the IoT world
Virtual entity	Contains functions and services for interacting with an IoT application using virtual entities
IoT service	Exposes an IoT device for the rest of the application to enable information exchange and control of actuators
Communication	Provides interface for IoT service using various technologies to connect IoT devices to the IP network
Security	Information security and privacy of an IoT application

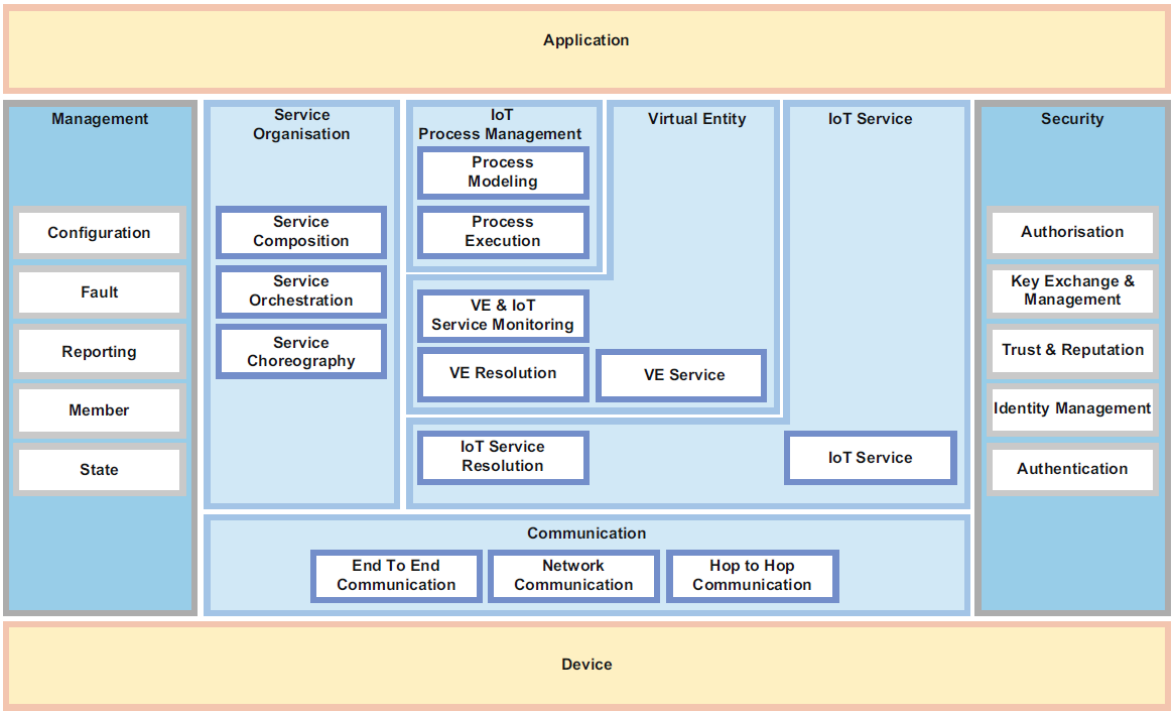

Figure 4.3: IoT-A Functional view[10, p. 168].

4.2.2 Information view

Information view describes virtual entities that model physical entities or "Things" of an IoT application and it is divided to information description, information handling and information life cycle sections[10, p. 183-196]. It defines static information structure and dynamic information flow. Viepoints and their purposes are listed in Table 4.3.

Table 4.3: Viewpoints in information view[10, p. 183-196]

Viewpoint	**Purpose**
Information description	Describes what kind of data and an IoT application has.
Information handling	Describes how information flows within an IoT application.
Information life cycle	Describes when a measurement of a device is triggered and how long it is stored.

Information description includes the descriptions of virtual entities. With UML diagrams, virtual entities can be presented as classes with attributes and information flow between virtual entities as associations. There are two types of models that can be used to explain virtual entities which are flat and hierarchical entity type models. As an example presenting the information structure of an logistic application, Figure 4.4 shows flat entity type model and Figure 4.5 shows hierarchical entity type model. The described logistic application consists of logistics personnel and material. Virtual entities have attributes such as name for driver, worker and manager as well as ID for box and pallet. In the flat entity model, each virtual entity type can have the same attribute and similarities between them are not considered. Hierarchical entity type model groups entity types with similarities together. For example, name attribute is common for Driver, Worker and Manager which can be grouped under Human.

Figure 4.4: Example flat entity type model of virtual entities in logistics[10, p. 184].

Information handling section defines which service handles what data and how information flows between the services. Four information flow patterns are described in this section and those are Push, Request/Response, Subscribe/Notify and Publish/Subscribe patterns.

The Push-pattern is a one-way communication pattern in which one side sends data to predefined receiver. This can be for example server controlling an actuator or measurement instrument sending data to a server. The Push-pattern is shown in Figure 4.6.

The Request/Response-pattern, shown in Figure 4.7, is a two-way synchronous communication in which each message sent requires a response. This can cause delays when multiple clients are sending requests to a single server and it takes time for the server to send response for each client.

The Subscribe/Notify-pattern, shown in Figure 4.8, allows a client to decide when

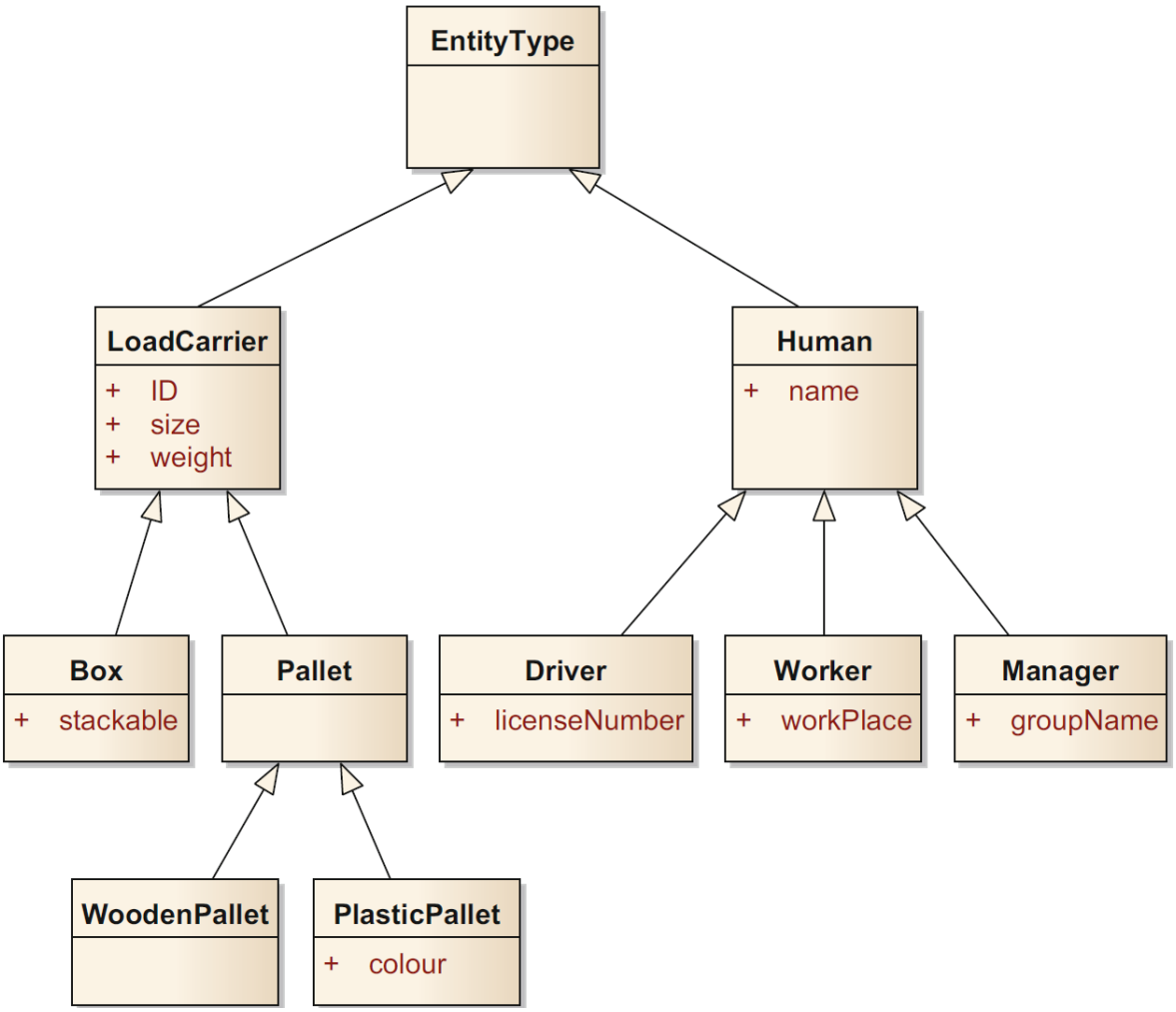

Figure 4.5: Example hierarchical entity type model of virtual entities in logistics[10, p. 185].

to start and stop receiving notification. This is asynchronous communication, therefore a client does not need to wait for a response from the server and only needs to act after receiving a notification. The Subscribe/Notify-pattern requires more powerful server than in the the Push or Request/Response -patterns since the server needs to keep track of the subscribed clients

The Publish/Subscribe, shown in Figure 4.9, is similar to the Subscribe/Notify-pattern but a broker is used between the server and clients. This allows multiple information providing servers to publish their information to a single broker and clients that are interested in a specific information can subscribe to it.

Figure 4.6: Push[10, p. 188]

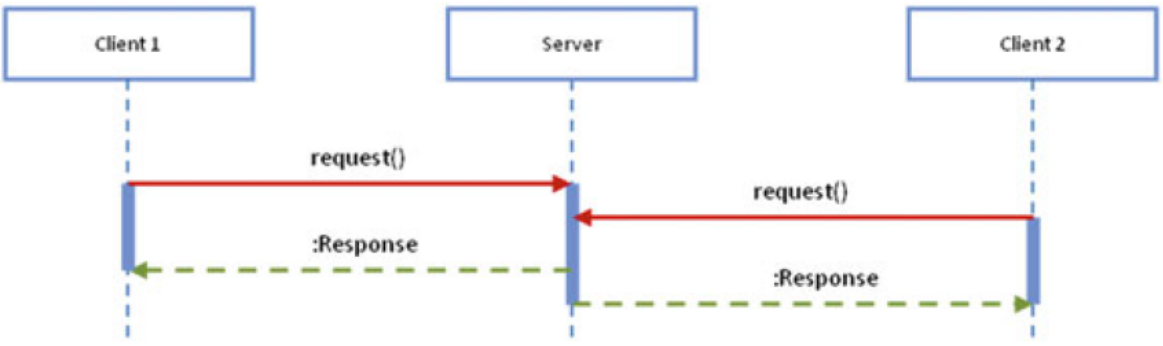

Figure 4.7: Request response[10, p. 189]

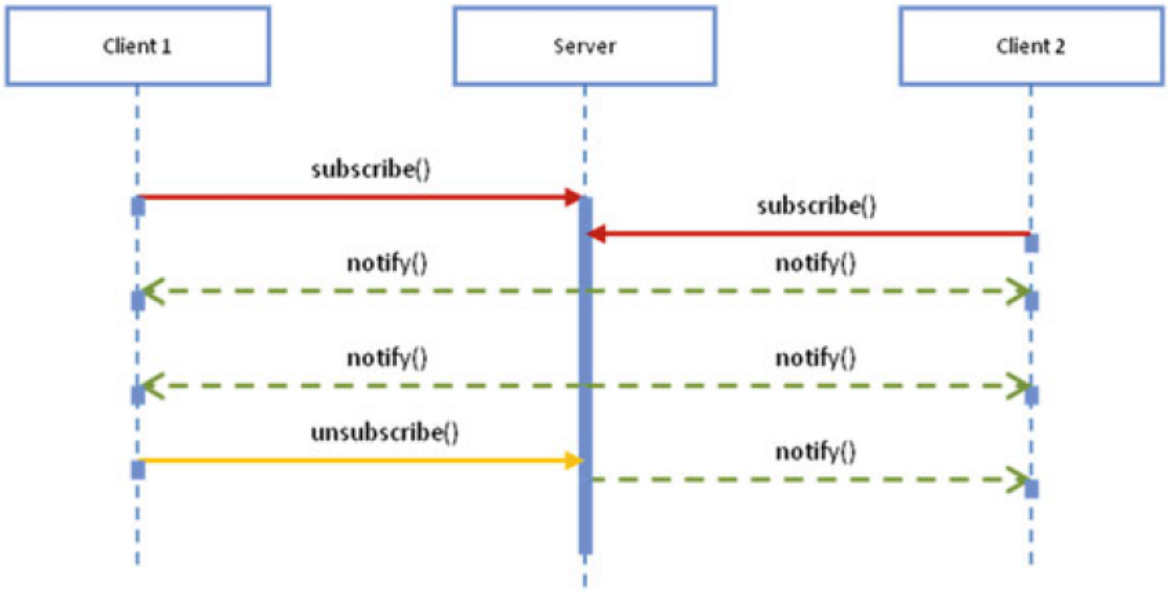

Figure 4.8: subscribe notify[10, p. 189]

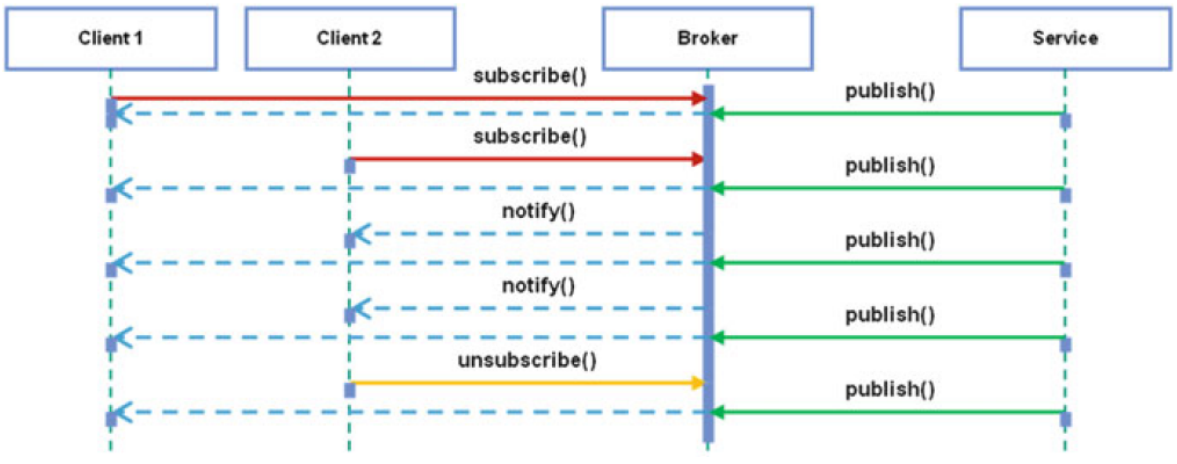

Figure 4.9: publish subscribe[10, p. 191]

Information life cycle section defines when a measurement of a device is triggered and how long it is stored. Measurements can be taken with specified intervals, triggered by environmental change or by an external request. Measurement process is defined based on the needs of an application. Measurement interval based approach can be for example that the temperature of a freezer storage is measured every 30 minutes to make sure that environmental conditions are within certain limits. Measurement action triggered by an environmental change can be for example when changes in movement or sound is detected by a sensor and a photo needs to be taken for a security purpose.

Information storage capacity is limited. Resolution of information affects the amount of storage capacity needed, therefore information can be reduced by analyzing it before storing. Information in a storage can be removed periodically or further reduce the resolution of information as it gets older.

4.2.3 Deployment and operational view

Deployment and operation view gives guidelines what to consider during the design process of an IoT application[10, p. 198-204]. Different technologies are not described in detail since technologies evolve over time, which would make the guidelines outdated after few years. Design choices to be made according to deployment and operational views are listed Table 4.4.

Table 4.4: Design choices in deployment and operational view[10, p. 198-204]

Design choices to make	**Description**
Device hardware	Computational power of devices.
Device connectivity	A single IoT application can use multiple communication technologies.
Communication protocol	Which communication protocols to use.
Services and resources	Where to deploy a service or a resource.
Information storage	Where to storage the acquired information.
Core engine software deployment	Where the core engine software is deployed.

Device computational power and connectivity are the first decisions that one has to make. Both are heavily dependent on the target application[10, p. 199-200]. Different technologies can have advantages over another, therefore a single application might require multiple technologies to be used. Typical technologies for connectivity in IoT applications are:

- Sensor and Actuator Networks
- RFIDs and smart tags
- WiFi or other unconstrained technologies
- Cellular networks

Once the connectivity technology has been decided, the next is the communication protocol[10, p. 200]. IoT-A framework recommends protocols that are compatible with IP, but performance requirements of an application might force the use of technologies that are specifically design in high performance in mind regardless of interoperability.

Third decision is the deployment location of services and resources, which are parts of the IoT application software that gather information and analyze it before it is stored[10, p. 200]. This decision is affected by the first two decisions, since choosing a measurement device that has low computational power requires information to be sent forward before it can be analyzed. The three options for this are:

- IoT devices - Lightweight services that can be run in a single IoT device
- Gateways - More complex services can be run in gateways that have more computational power
- The cloud - Complex services can be run on the cloud and used to analyze data from big number of sources

Information collected by an IoT application needs to be securely stored in a way that it is available when needed, cannot be changed without authorization and enough data redundancy to meet the application requirements[10, p. 201]. Options to store information in an IoT application are:

- Local only - Information is stored locally on the device and retrieved only when required. There is no need for complex databases but there is a high risk that the information is lost if the device malfunctions.
- Web only - Information is sent to a web server as soon as it is measured but it is not stored locally on the device. This reduces the cost of measurement devices since there is no need to have dedicated memory for information storage.
- Local with web cache - Information is stored in many layers of an IoT application to reduce the possibility that information is lost due to a malfunction in a part of the application.

The last decision to make is the core engine software deployment[10, p. 201]. This is the main service or "brains" of an IoT application. Purpose of this service to retrieve information other services, analyze it and show it to the users. There are two options for deployment location.

- Internal deployment - The core engine software is deployed to a server that is dedicated to the IoT application. The core engine has direct access to the other services and resources within the server. Future development and maintaining of the core engine is done by the organization that owns the IoT application.
- External deployment - Development and maintaining of the core engine software is done by a third party. An IoT application using external core engine software needs to use APIs from the third party service. This can hinder the flexibility of the IoT application but can reduce the maintaining costs.

4.2.4 Architectural Perspectives

Each architectural perspective includes activities, tactics and guidelines[10, p. 204-208]. Unified requirements for each perspective have been taken from the IoT-A project website which is kept up to date[18]. Architectural perspectives are not answering questions that are directly linked to a technical implementation of an IoT application, but to questions that come up when a life cycle of the IoT application needs be planned. Table 4.5 shows for which question each of the perspectives gives guidelines for. Detailed actions and activities for each question are listed in Appendix A.

Technologies evolve continuously. If interoperability was taken into consideration while designing an IoT application, new technologies can be added to the IoT application and take an advantage of them. Requirements for an IoT application can also change and can cause the IoT application to lose its value in case it cannot adapt. Activities and tactics to be taken when designing an IoT application for future changes in requirements or technologies are listed in Table A.1.

Table 4.5: Architectural perspectives and related questions[10, p. 204-208]

Perspective	**Question**	**Unified requirements**
Evolution and Interoperability	How changes in requirements and evolving technologies are handled?	UNI.003 UNI.012 UNI.014 UNI.047 UNI.071 UNI.093 UNI.095 UNI.241 UNI.313 UNI.326
Availability and Resilience	What can go wrong and how to deal with problems?	UNI.005 UNI.065 UNI.308
Performance and Scalability	How should the application perform at launch and when data volume increases?	UNI.008 UNI.100 UNI.101 UNI.102
Security	How to information security issues are handled with resource constrained devices?	UNI.062

IoT applications that are crucial to businesses need to have as high availability as possible. Amazon Web Services (AWS) defines the availability as in Equation 4.1[5, p. 4]. Increasing the availability percentage usually increases the costs as well, therefore AWS divides the common application availability percentages to five categories, as seen in Table 4.6[5, p. 4].

$$Availability = \frac{AvailableforUseTime}{TotalTime} \tag{4.1}$$

Resiliency is about automatically recovering from disruptions and how to mitigate them[5, p. 3]. IoT applications are commonly distributed from small measurement devices to analyzing services on the cloud so there are many possible failure

points[10, p. 208]. IoT ARM framework's guidelines for availability and resiliency are listed in Table A.2.

Table 4.6: Availability categories defined by AWS[5, p. 4]

Availability	**Maximum Unavailability (per year)**	**Application Categories**
99%	3 days 15 hours	Batch processing, data extraction, transfer, and load jobs
99.9%	8 hours 45 minutes	Internal tools like knowledge management, project tracking
99.95%	4 hours 22 minutes	Online commerce, point of sale
99.99%	52 minutes	Video delivery, broadcast workloads
99.999%	5 minutes	ATM transactions, telecommunication workloads

Performance requirements for an IoT application should be set so that the application can handle business requirements without increasing the costs too much. Even though performance requirements might be clear when launching the IoT application, the increase of devices or users might require the application capacity to be scaled up. AWS instructs designers to monitor the capacity needs and increase it automatically if needed[5, p. 2]. This helps the IoT application to optimize running cost as well as manage possible denial-of-service attacks[5, p. 2]. Activities and tactics from IoT ARM framework on performance and scalability are listed in Table A.3.

Information security perspective of IoT ARM framework guides designers on how to create a secure IoT application and it includes data confidentiality, integrity and authorization[10, p. 210]. Activities and tactics from IoT ARM framework on information security are listed in Table A.4.

5 IoT application development

As explained in Chapter 1, the main motivation to develop the application is to have cost effective way to constantly measure the conditions within buildings and be able to make data driven decisions how to improve the energy efficiency. High level user expectation for the application is shown in Figure 5.1. Measurement nodes are placed in different parts of a building and the measurements are uploaded to the cloud for users to view them.

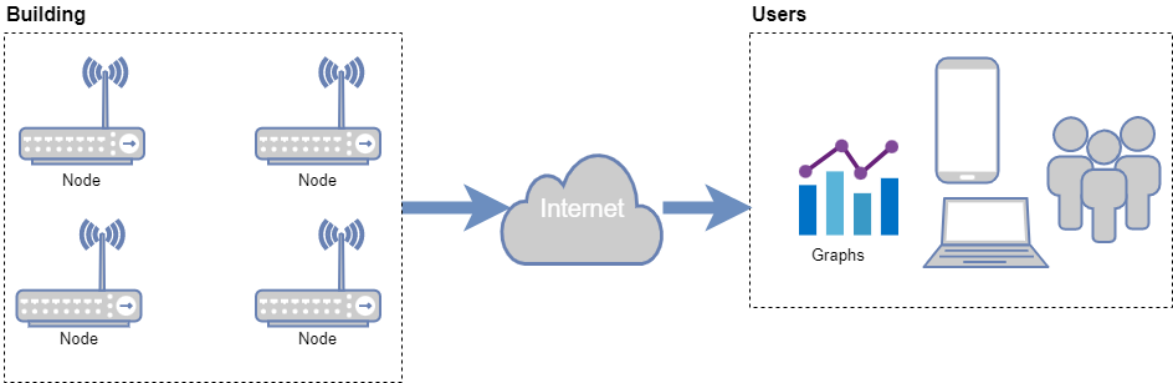

Figure 5.1: High level user expectation

In addition to the use case presented in this thesis, it is clear that IoT is already a big market and will keep growing in the future as presented in Section 2.1. Developing an application for a single use case would limit its value in the IoT world. Therefore, the application should be designed in way that it will be able to handle different kind of sensors and could be utilized in various situations for example home automation, farming, asset tracking or storage management. In the next sections, flexible IoT application is developed using the IoT ARM framework described in Section 4.2.

Requirements for the application in this thesis were first gathered from various sources. The requirements were considered when developing the application to make sure that it will bring value to users, enable scaling up and further development of new features. The requirements are presented in Section 5.1. After requirements were clear, design decisions needed to be made which are then presented in

Section 5.2. The developed application is explained in detail using IoT ARM guidelines described in Subsection 4.2.2. In the last part of this chapter, Section 5.4, the application is evaluated and compared with the defined requirements.

5.1 Requirements

In order to develop an application that does what we want it to do, it is important to define the requirements for it and the reasons behind them. The requirements for the application are derived from the IoT ARM framework[18] as well as from the conclusions made from the IoT market research presented in Section 2.1. Application requirements and applicable unified requirements are listed in Table 5.1. All IoT ARM framework requirements that were considered are listed in Appendix B.

Table 5.1: Application requirements

REQ ID	**Description**	**Unified requirements**
REQ1	Can be connected to other services using standard APIs	UNI.014, UNI.047, UNI.071, UNI.093, UNI.095, UNI.241, UNI.313, UNI.326
REQ2	Scaling cost is low	UNI.102
REQ3	Maintenance cost is low	UNI.003, UNI.093
REQ4	Reliable communication with battery operated devices	UNI.065, UNI.308
REQ5	Autonomous operation of devices	UNI.005

Most of the IoT-A unified requirements that were found to be applicable for the IoT application in this thesis are related to "Evolution and Interoperability" of IoT-A architectural perspective. Therefore it is clear that commonly used standard API technologies are needed to fulfill the interoperability requirements which is the first requirement REQ1.

The second requirement REQ2 for the application that the scaling cost is low. As the research commissioned by Microsoft stated in Section 2.1, the number one failure reason for IoT Proof of concept projects is the high cost of scaling up applications. UNI.102 requirement states that some of the computing required can be done in the cloud. This can be costly for a small IoT application but can make it easier to increase resources when the application grows bigger.

Depending on the lifetime of an application, maintenance cost will most likely be greater than the cost of development until launch. Maintaining costs can be divided into running costs such as maintaining servers and to further developing the application with new features. Therefore, it is important for the architecture of the application to be well thought to enable future development and managing of connected devices. This is the third requirement REQ3. Both UNI.003 and UNI.093 requirements assess the need for an IoT application to prepare for the future.

The application in this thesis is aimed for battery operated devices to lower the deployment cost. In addition, reliable communication is required to enable wide range of application areas. Unified requirements related to these are UNI.065 and UNI.308. This is the fourth requirement REQ4 for the application.

The last requirement REQ5 for the application is that the connected devices should operate autonomously. This includes measuring the environment periodically and forwarding the data to the next layer. All this without the need for a user to initiate the actions.

5.2 Architecture design decisions

Architecture design decision process follows the IoT ARM framework's deployment and operational view presented in Table 4.4. Architectural perspective questions

and unified requirements, presented in Table 4.5, are considered when applicable.

5.2.1 Sensing and network layers

According to IoT ARM framework's deployment and operational view, the first design choices are related to IoT devices and listed below. When compared to the four stage IoT architecture model, the choices belongs to sensing and network layers.

- Device hardware
- Device connectivity
- Communication protocol

The design of the sensing layer needs to be done considering the network layer and vice versa. Since it was clear at the start of this thesis that the application to be designed should have the possibility to be battery operated (the sensing layer), only LPWAN technologies (the network layer) were considered over other technologies that have higher data rate. This is because LPWAN technologies are known to have low power consumption. The technologies are Sigfox, LoRaWAN and NB-IoT, which are compared in Table 5.3 between the main features that are related to this application. Three out of five requirements defined in Section 5.1 are related to sensing and network layers and are listed in Table 5.2.

Table 5.2: Summary of communication technology requirements

Technology requirement	**LoRaWAN**	**Sigfox**	**NB-IoT**
Scaling cost is low	>€233	>€415	>€15,000
Maintenance cost is low	None	€10 / year / device	€3.8 / year / device
Reliable communication with battery operated device	Yes	Yes	Yes

When an application grows in size, increasing the network coverage becomes necessary. Users can expand the Sigfox coverage to indoor areas by using Access Station Micro gateway with a rental price of approximately €268/year in Japan[21] or one time price approximately of €415 in the US but unfortunately the gateway is not available globally[13]. Same as with Sigfox, LoRaWAN coverage can be expanded to areas where the global coverage does not reach. LoRaWAN gateway price starts at €233 in Finland[12]. The price of a NB-IoT base station is more than €15,000 and since NB-IoT uses the licensed LTE frequency bands as seen in Table 5.3, it is not possible for individual users to expand the coverage by themselves[27, p. 5].

Maintenance costs can be significant during the lifetime of an application. Sigfox connectivity for one device with 50 message per day cost around €10/year excluding tax depending on the country of use[36]. Olivia operator offers Vodafone NB-IoT connectivity for one device with 50 MB data and €3.8/year[33]. The Things Network built on LoRaWAN is a global open network which does not have a subscription cost[48].

Last requirement related to sensing and network layers is the reliable communication with battery operated devices. Unified requirement UNI.093 states that "An application built using the ARM shall be extensible for future technologies"[18]. All the three technologies are continuously being developed and can be connected to the IP network. Therefore it can be said that this requirement has been met.

All the three communication technologies are calculated to have over 10 years of battery lifetime[27, p. 1]. LoRa and NB-IoT offers also the possibility to decrease latency by using non-battery operated devices[27, p. 4].

Reliability of the considered technologies have been proven in the uses cases presented the sections for each technology but NB-IoT has the best quality of service due to use of LTE frequency bands[27, p. 4].

Table 5.3: Comparison of considered LPWAN technologies[27, p. 3].

	Sigfox	**LoRaWAN**	**NB-IoT**
Modulation	D-BPSK (Uplink) / GFSK (Downlink)	CSS	QPSK
Frequency	Unlicensed ISM bands	Unlicensed ISM bands	Licensed LTE frequency bands
Bandwidth	100 Hz	250 kHz and 125 kHz	200 kHz
Maximum data rate	100 bps	50 kbps	200 kbps
Maximum messages/day	140 (Uplink) / 4 (Downlink)	20 - 500(see Section 3.1)	Unlimited
Maximum payload length	12 bytes (Uplink) / 8 bytes (Downlink)	243 bytes	1600 bytes
Range	10 km (urban) / 40 km (rural)	5 km (urban) / 20 km (rural)	1 km (urban) / 10 km (rural)
Allows private network	No	Yes	No
Standardization	Sigfox company is collaborating with ETSI	LoRa-Alliance	3GPP

Table 5.4: Design choices for sensing and network layers

Design choices	**Decision**
Device hardware	Arduino MKR WAN 1310
Device connectivity	LoRa
Communication protocol	LoRaWAN

Design choices for the sensing and network layers are shown in Table 5.4. **LoRaWAN** was decided to be used as the communication technology from IoT devices towards the server due to non-existing subscription costs and expanding the coverage is relatively cheap. As it was stated by the research mentioned in Section 2.1, the most common reason for proof of concept projects to fail is the high cost of scaling. Therefore it is most important to choose a communication technology that is the most versatile in regarding to expanding coverage and types of applications it can be used in. LoRaWAN has the lowest maximum data rate of the three communication technologies with 50 kbps (see Table 5.3) but this is not an issue since the application is aimed for instruments with low data rate measurements. The possibility to send up to 500 messages per day with LoRaWAN, compared to maximum of 140 message per day (with the most expensive subscription model) for Sigfox 0G network, is also an advantage. Same as the other LPWAN technologies, LoRaWAN requires that a message sent by a device is forwarded to the IP network and for this **The Things Network**, explained in Subsection 3.1.2, will be used.

Once the communication technology has been set, the sensing node platform can be decided. **Arduino MKR WAN 1310** offers built-in LoRA connectivity, analog inputs for various sensors and it can be powered using batteries. Arduino offers sensor shields that can be attached to MKR WAN 1310. MKR ENV shield will be used in this demonstration which has various sensors such as temperature, humidity, pressure and illuminance.

5.2.2 Data processing and application layers

Design choices defined in IoT ARM framework's deployment and operational view relating to data processing and application layers are listed below.

- Services and resources
- Information storage
- Core engine software deployment

The need for the data processing layer depends on the capabilities of the network and application layers. Low data rate of the network layer might force the data to be pre-processed before sending. As explained in Subsection 4.1.3, storage limitations of the application layer might require that the raw data is pre-processed and reduced before storing to the final destination. Three out of five requirements defined in Section 5.1 are directly affected by the technologies used in data processing and application layers and those are which are listed below.

- Can be connected to other services using standard APIs
- Scaling cost is low
- Maintenance cost is low

Application layer services introduced in Section 2.2, Azure IoT hub, AWS IoT Core and askSensors were considered. Kiwi technologies' and Vaisala's solutions were left out since adding new types of devices is more restricted than with the other three services. All the considered IoT cloud services use standard APIs that can be used to integrate any new device or application.

The main cost for scaling an application comes from the core engine software described in Subsection 4.2.3, which can be either internal deployment or external deployment[10, p. 201]. Using Azure IoT Hub and AWS IoT Core allows the core engine software and all needed services to be deployed within the same cloud server where as AskSensors would require the core engine software to be deployed to a separate server[29][4][8].

Use of standard APIs, has a cost reducing effect on scaling of an application, since there is no need to develop new interfaces. The core engine software as well as any

used services would need to use standard APIs to fulfill this requirement.

Running costs such as information storage and use of external services increases the maintenance cost. All the three paid cloud services offer unlimited information storage but with extra cost with AskSensors being the cheapest by $9/month[29][4][8].

Blockchain-based information storage, as described in Subsection 4.1.4, can be used to decentralize and improve the availability of the information storage. Accessing the data can be costly, therefore increasing the maintenance cost of an IoT application.

Table 5.5: Design choices for data processing and application layers

Design choices	**Decision**
Core engine software deployment	Local server
Information storage	Local and AskSensors
Services and resources	AskSensors

Design choices for the data processing and application layers are shown in Table 5.5. **Local server** was chosen to be used as the core engine software deployment model to keep development cost of the IoT application as minimum as possible. The developed IoT application can be deployed to any cloud service if there is a need to scale up the application in the future.

Local storage is used as the main information storage to enable detailed data analyzing. Information is stored also to **AskSensors** but the service does not offer data analyzing functions.

Centralized information storage was chosen over a blockchain-based application. As one of the application requirements listed in Section 5.1 is "Maintenance cost is low", already high price of a blockchain currency would increase the running costs significantly and due to volatility of blockchain currencies, costs could suddenly become unbearable. Blockchain-based application requires fairly high computational power from the connected IoT devices and high computational power requirement is a disadvantage, if an IoT devices is using batteries as a power source.

As described in Subsection 4.2.1, IoT Service functional group exposes the IoT devices to the rest of the application. For the IoT application developed for this thesis, this means the user interface where the measurement results for each IoT device can be seen. It is also possible to use standard APIs to retrieve the information up to 50 latest measurements and further develop new functions[8]. In addition to APIs provided by AskSensors cloud service, local server as the core engine software deployment model allows further development of new APIs.

5.3 Architecture of developed application

The developed application divided according to the four-stage layer architecture model is shown in Figure 5.2 and next sections are divided in to these layers where the details of each layer are described. Overview of the application, and how users can interact with it, is shown in Figure 5.3. Raspberry PI server is the main information hub that stores received data, processes it and forwards it for users to see the data in graphs via AskSensors or directly accessing the data using REST API.

Four types of communication patterns were introduced in Subsection 4.2.2. Push communication pattern (see Figure 4.6) is used from measurement nodes to the RAK LoRaWAN gateway. Between the RAK LoRaWAN gateway and the back-end server, publish-subscribe communication pattern (see Figure 4.9) is used . Push communication pattern is again used when the back-end server forwards the processed data to AskSensors service. Request-response communication pattern (see Figure 4.7) is used when REST API is used to retrieve data directly from the back-end server.

5.3.1 Sensing layer

MKR WAN 1310 with MKR ENV shield is used as the measuring node. MKR WAN 1310 has built-in LoRaWAN connectivity and MKR ENV shield adds temperature, humidity, atmospheric pressure, illuminance, UVa intensity, UVb intensity and UV index[7][6]. Two identical measurement nodes, shown in Figure 5.4, were assembled for this thesis.

MKR ENV shield has built-in HTS221 relative humidity and temperature sensor[6].

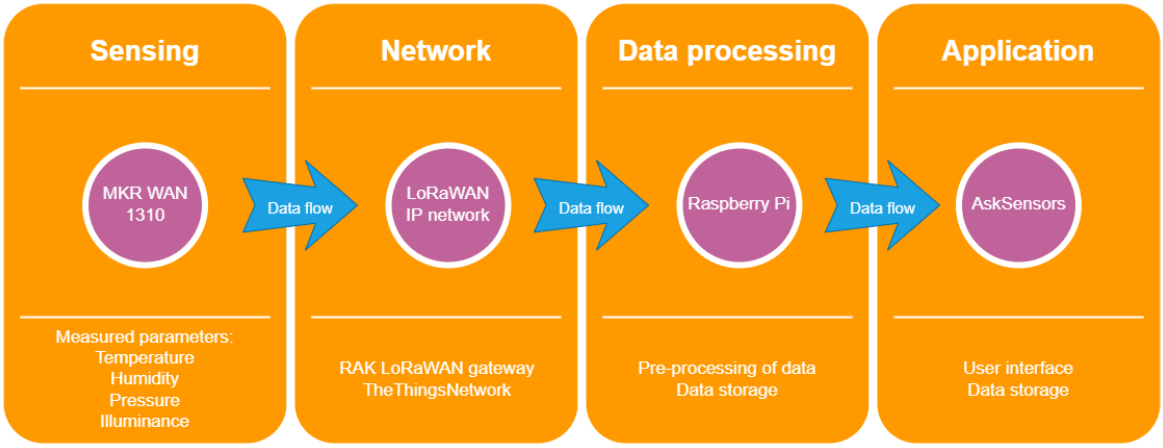

Figure 5.2: Four-stage architecture of the developed IoT application

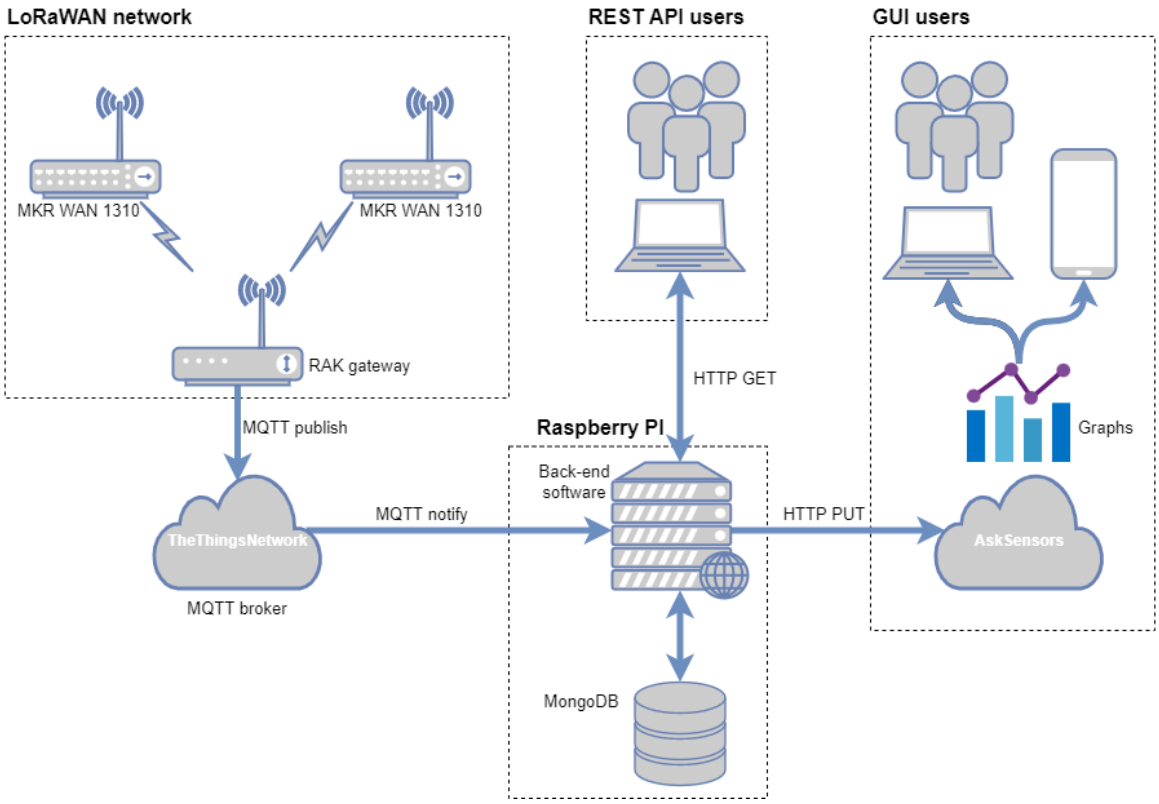

Figure 5.3: Overview of the developed IoT application

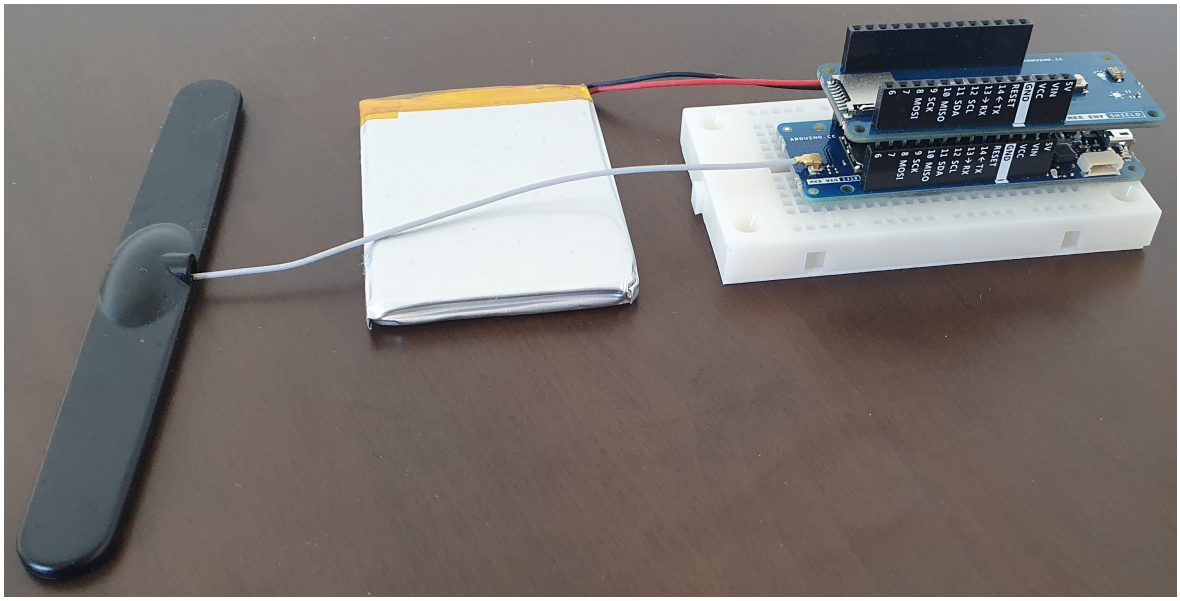

Figure 5.4: MKR WAN 1310 node with MKR ENV shield (right), antenna (left) and a battery (middle)

Relative humidity accuracy is $\pm$ 3.5 $\%rh$ between 20 $\%rh$ and 80 $\%rh$ and temperature accuracy is $\pm$ 0.5 $^{\circ}C$ between 15 $^{\circ}C$ and 40 $^{\circ}C$[44]. The defined accuracy does not include self-heating caused by operation of MKR WAN 1310 and drift of the sensors over time. Self-heating affects the measured temperature values and since relative humidity is calculated based on the temperature values, relative humidity values are also affected. Therefore, the actually accuracy of the temperature and relative humidity sensors is expected to be worse than defined in the datasheet.

Measurement accuracy of the pressure and illuminance sensors used in MKR ENV shield has not been defined [45][52]. Which means that the changes in the measurement values can be monitored but the absolute values are not accurate. Pressure values are given in kPa and illumance values as LUX.

The goal of this thesis is not to create measurement nodes with high measurement accuracy but to have a IoT platform that various kinds of nodes can be added to. Therefore, the measurement accuracy of MKR WAN 1310 with MKR ENV shield is sufficient.

Node activity diagram shown in Figure 5.5 describes how the nodes operates. When a node is powered up, it will send a join request to nearby LoRaWAN gateways and wait for a reply. If successful, measurement loop between active and sleep states is started and this continues until the battery runs out.

Measurements of sensors can fluctuate within a short time period which causes measurement errors. To minimize this effect, 5 measurements are taken in 1 second intervals and an average is calculated. The average is then sent forward to the network layer. The node goes to a deep sleep state for 15 minutes which after it wakes up and starts the measurements cycle again. Source code for the MKR WAN 1310 can be seen in Appendix C.

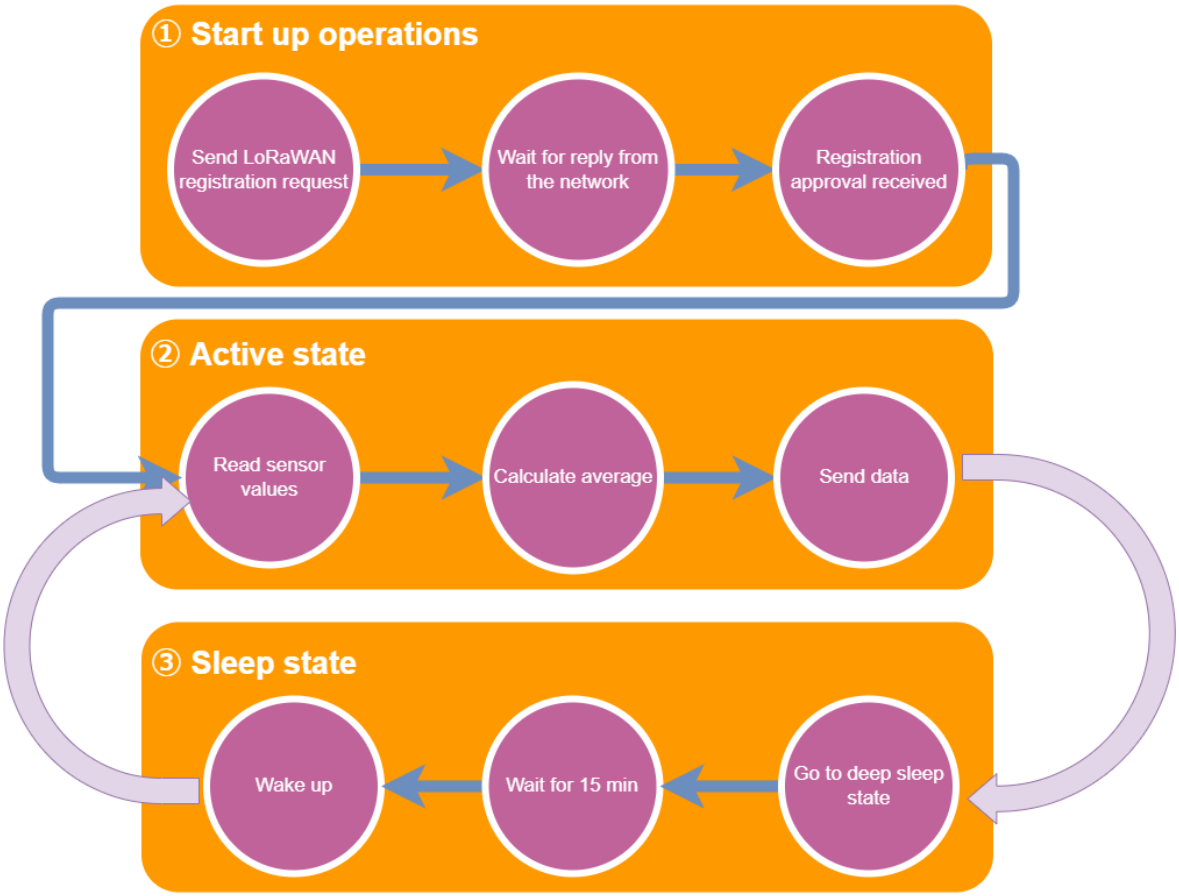

Figure 5.5: Node activity diagram

5.3.2 Network layer

Measurements values from nodes are transferred to the user interface via various communication protocols which are shown in Figure 5.6. The data transfer process starts with a measuring node sending data to RAK gateway with LoRaWAN protocol. RAK gateway, shown in Figure 5.7, operates between the LoRaWAN network and IP network which allows nodes to be connected to the internet. Between the gateway and the data processing Raspberry Pi server, publish-subscribe information flow is used, as described in Subsection 4.2.2. The gateway publishes a message to a MQTT broker run by TheThingsNetwork which then notifies the subscribers of the new message. Once a message is received by the data processing Raspberry Pi server, it is forwarded to the AskSensors service using REST API. Users can then check the measured data from the AskSensors' user interface.

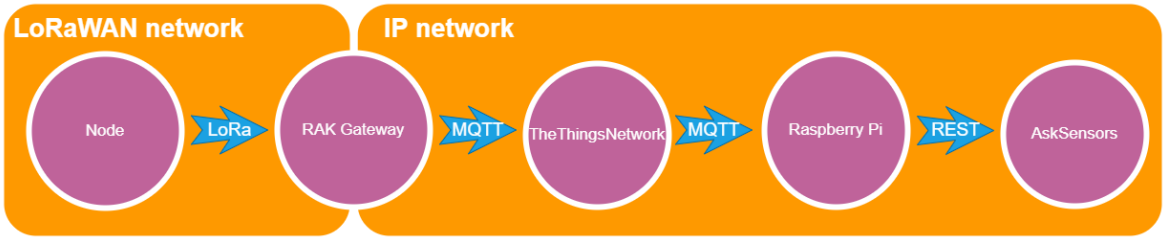

Figure 5.6: Network layer communication protocols

5.3.3 Data processing layer

Measured data is pre-processed in data processing layer before it is forwarded to the application layer. Data from nodes can be invalid, for example negative relative humidity values or temperature values lower than the absolute zero temperature of -275.15 $°C$. Invalid measurements are discarded as they could cause issues to any external application that makes automated decisions based on the data.

Raspberry Pi computer is used for the data processing and data storage and it is shown in Figure 5.8. It is small but powerful enough for this application. It can be easily swapped with a physical or cloud server with more processing power since the back-end software is developed using Java and it is running on Linux operating system.

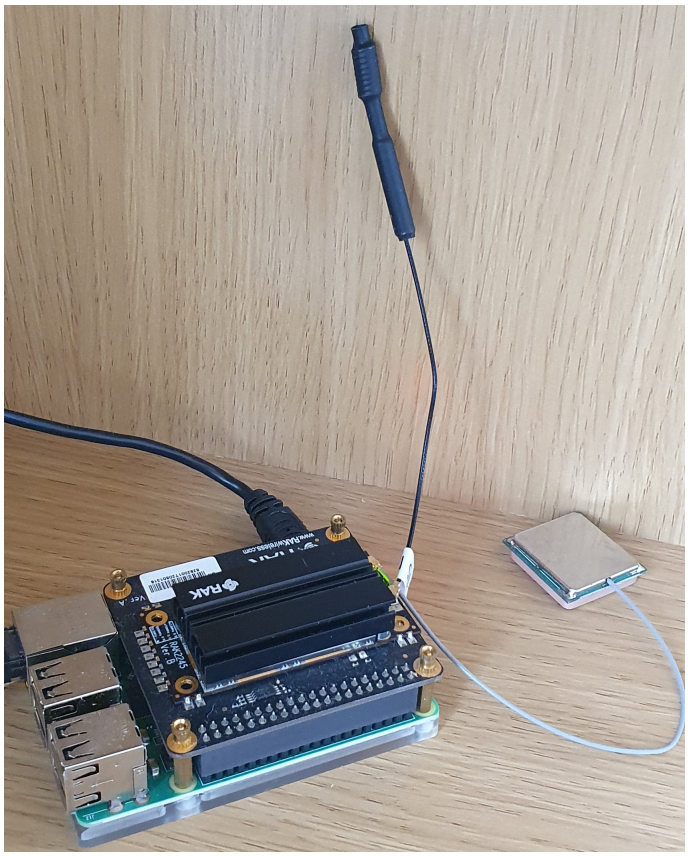

Figure 5.7: RAK LoRaWAN gateway

Figure 5.8: Raspberry PI server for data processing

Activity diagram of the back-end server software is shown in Figure 5.9. When the software is started, server settings are loaded including information on the registered nodes. Two threads are then started which are operating parallel and those are HTTP server which listens for REST API messages and MQTT client which subscribes to MQTT broker hosted by TheThingsNetwork.

There are two types of messages that the REST API can receive, data requests and xml files with measurement information from nodes. Data requests are HTTP GET messages from users or automated systems to retrieve measurements results. It is possible to retrieve either the latest measurement of a node or a range of measurements defined by the start time and the end time. Measurements are replied in JSON format. Example reply message when a range of measurements is requested is shown in Appendix D.

LoRa xml is a format used by a Finnish communication network company Digita to transfer measurement data. The ability to receive LoRa xml files is from the previous Indoor Air Quality project that the back-end software was also used in[34]. LoRa xml files are not needed in the thesis since LoRaWAN network provided by Digita was replaced with a local LoRaWAN gateway and MQTT broker hosted by TheThingsNetwork.

As described in Subsection 4.2.2 regarding to information description, classes and associations are presented with UML diagrams. Class diagram of the developed application is shown in Figure 5.10. ServerSettings class contains information about the used Mongo database and connection parameters to AskSensors service. MqttSettings class associated with ServerSettings class contains the connection parameters to the MQTT broker hosted by TheThingsNetwork. List of registered IoT devices (Device class) is associated with the ServerSettings class. Each Device can contain up to 5 channels and each channel can have one of the below measurement parameters.

- Temperature
- Humidity
- Pressure

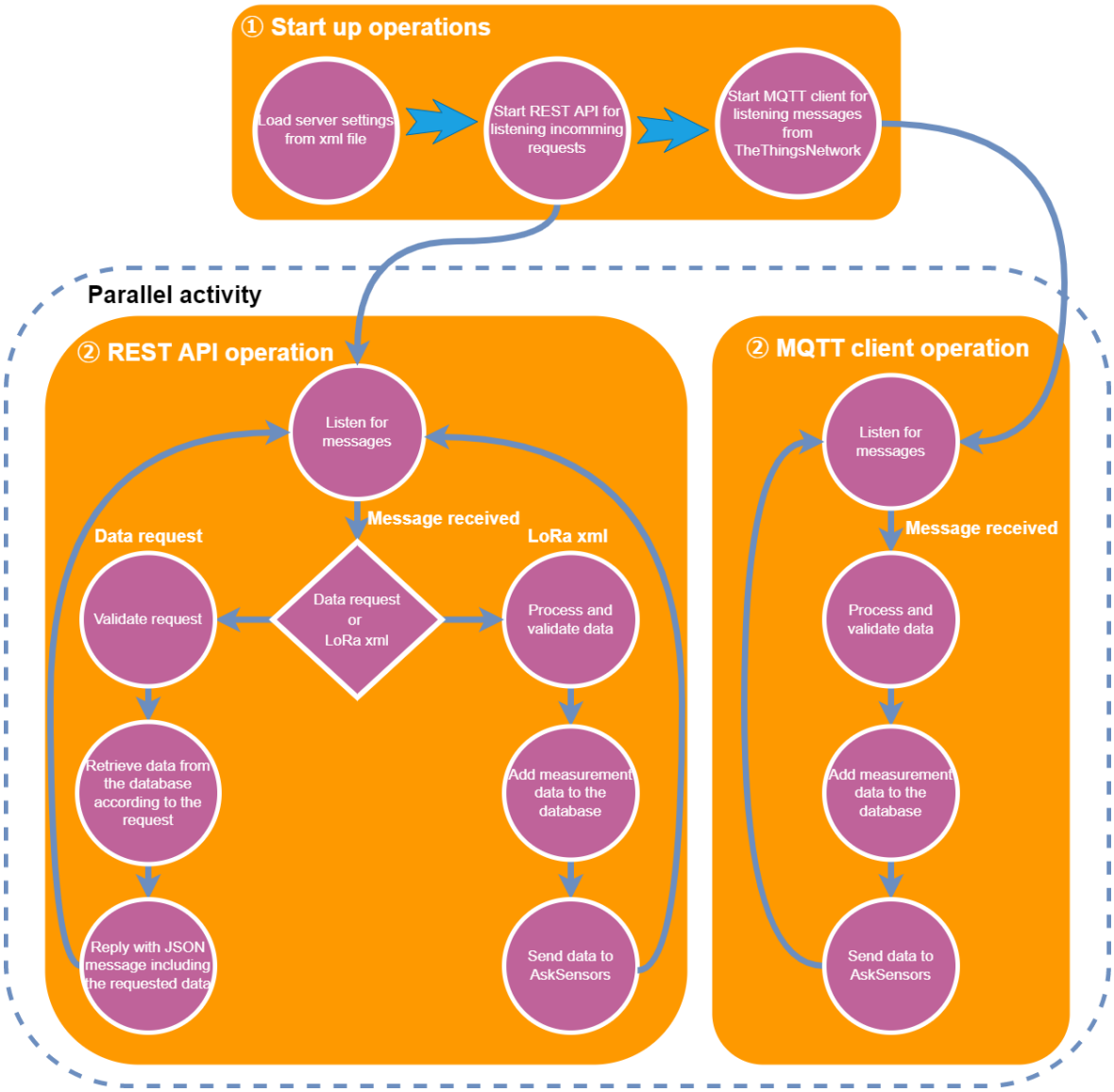

Figure 5.9: Activity diagram of the back-end server software

- Illuminance
- CO2

5.3.4 Application layer

The results of the developed application working from end-to-end can be seen in AskSensors cloud service. The service does not have data processing capabilities and is used only for showing the measurements results. Therefore, all data received by AskSensors needs to be valid. Measurements done by one of the MKR WAN 1310 nodes are shown in Figure 5.11 for relative humidity, Figure 5.12 for temperature, Figure 5.13 for illuminance and Figure 5.14 for pressure. Shown measurement period is from 2022-10-27 20:00 to 2022-10-28 08:00 and it contains the last 50 measurements. The measurements were done indoors in Tokyo, Japan. The nodes were placed in a room without heating and measuring with 15 minutes intervals.

Measured humidity values were relatively stable during the measurement period. Temperature can be seen constantly decreasing with the lowest value to be 14.3 °C. Illuminance measurement values clearly shows that lights were turned on and off between 2022-10-27 20:30 and 2022-10-27 22:30 and also that the room starts to receive gradually more light starting from 2022-10-28 06:00. Pressure values changes between 102.0 kPa and 103.0 kPa which is expected since absolute pressure does not usually change much within few hours. The resolution of the pressure measurements is 1 kPa which causes the graph to seem like the pressure is changing quickly.

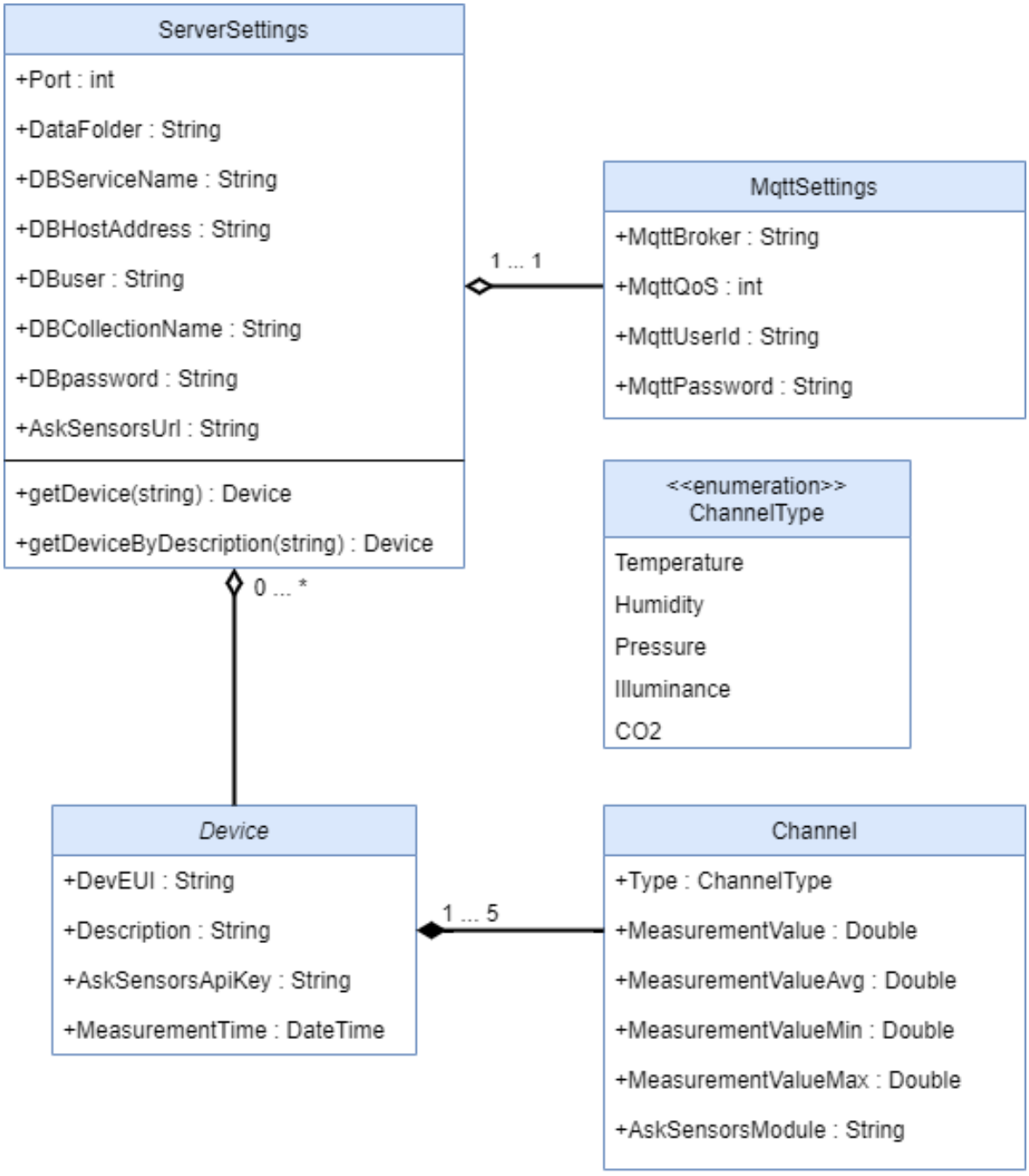

Figure 5.10: Class diagram of the developed application

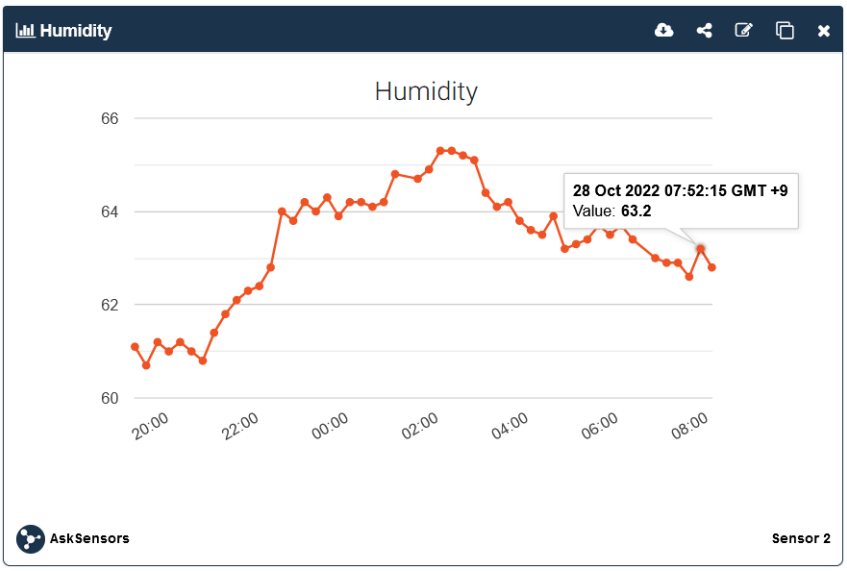

Figure 5.11: Measured humidity values shown in AskSensors.

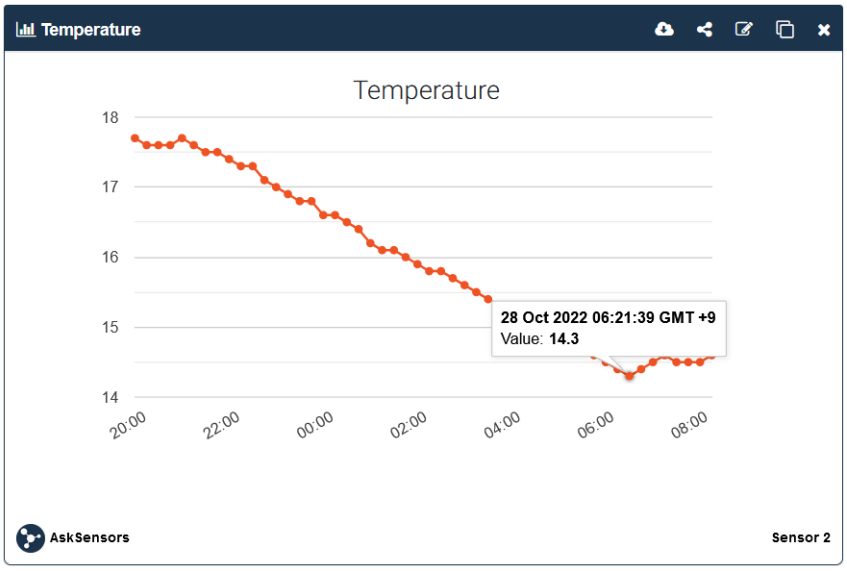

Figure 5.12: Measured temperature values shown in AskSensors.

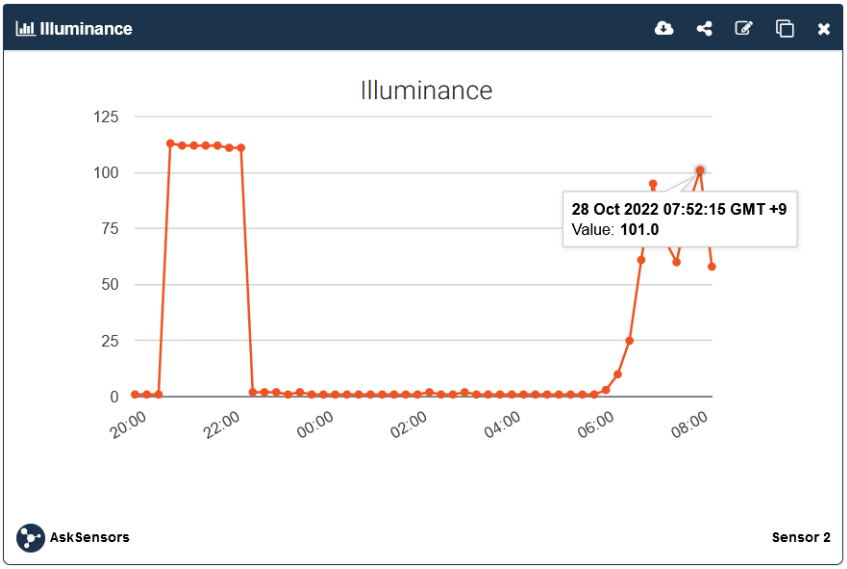

Figure 5.13: Measured illuminance values shown in AskSensors.

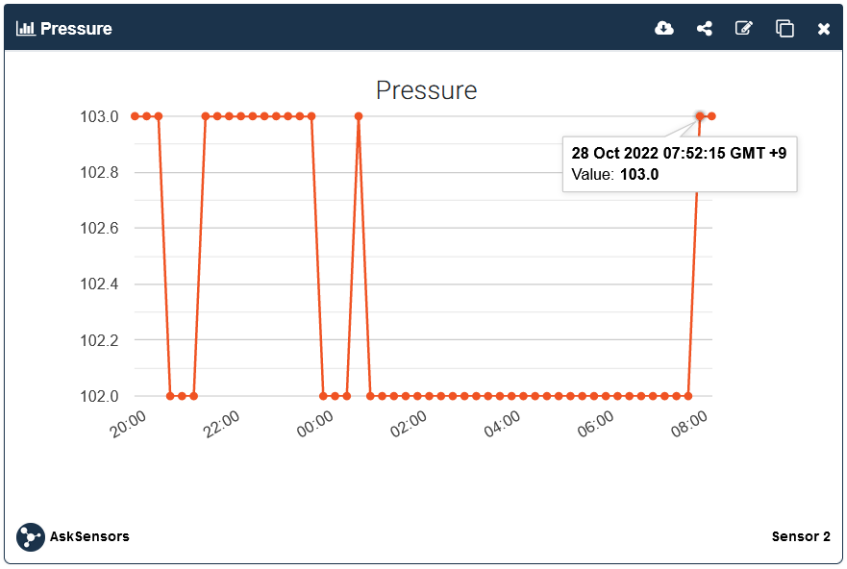

Figure 5.14: Measured pressure values shown in AskSensors.

5.4 Evaluation

In this section the developed application is evaluated and compared to the requirements listed in Section 5.1. The requirements for the application were:

- REQ1 - Can be connected to other services using standard APIs
- REQ2 - Scalability cost is low
- REQ3 - Maintenance cost is low
- REQ4 - Reliable communication with battery operated devices
- REQ5 - Autonomous operation of devices

Connecting to other services using standard APIs (REQ1) was the first requirement for the application. No new this application specific protocols were created and only standard APIs such as MQTT and REST were used. With the REST API described in Subsection 4.1.3, it is possible to connect the application to external applications to further analyze measurement data and expand the features of the application. Therefore it can be stated that REQ1 was fulfilled as well as the interoperability requirement by IoT ARM guidelines.

Second requirement was that the scaling cost is low (REQ2) since it was the number one reason for IoT proof of concept projects to fail according to the research presented in Section 2.1. When an application is scaled up, the maintenance cost also increases. Therefore, the second (REQ2) and the third (REQ3) requirements are closely related. The cost of scaling ca be divided into the cost of increasing measurement node fleet and the coverage of the wireless network. Generated data increases when the number of measurement nodes increases which then requires more from the application resources. As shown in Table 5.2, LoRaWAN technology does not bring additional cost per device when private LoRaWAN network is used when compared to sigfox and NB-IoT. However, maintaining a private LoRaWAN network can be more costly than using a service provider's network when an application is scaled up. Therefore, it can be stated that REQ2 and REQ3 are partly achieved.

Table 5.6: Reliability measurements done using two MKR WAN 1310 nodes

Distance	**Total measurements**	**Lost measurements**	**Loss rate**
10m	1812	13	0.72%
100m	606	7	1.16%

The reliability of communication with battery operated devices (REQ4) was tested using the two assembled measurement nodes shown in Figure 5.4. The testing was done indoors in Tokyo, Japan and the measurement interval was set to 15 minutes. Distance from the RAK LoRaWAN gateway was 10m for node 1 and 100m for node 2. Node 2 was placed in another building to test how well LoRa signal can pass through multiple buildings between the node and the gateway. The measurement results are shown in Table 5.6. Tokyo is known of its tall buildings but between the gateway and the node 2, there were only smaller houses.

The node placed 10m away from the gateway had only 0.72% loss rate out of 1812 measurements. The node placed 100m away had slightly higher loss rate at 1.16% out of 606 measurements. The observed maximum measurement loss in a row was 1. Since the measurement interval was 15 minutes, this means that there was 100% success rate to get measurement once in a 30 minutes. For the application developed for this thesis, this is acceptable loss rate. From communication technology perspective, it can be stated that the reliable communication requirement with battery operated devices has been achieved. Even better reliability can be achieved if the application software is improved to noticed any loss of measurements and retransmit if necessary.

Autonomous operation of devices (REQ5) was the last of the five requirements. Measurement nodes should be able to collect data periodically or when requested and deliver it to the next layer. This was achieved as demonstrated in Subsection 5.3.4. Temperature, humidity, pressure and illuminance measurements done by one of the nodes were shown in AskSensors service, which tests the whole application from end-to-end.

Autonomous operation also includes resiliency to recover from failures and for measurement nodes, this means failures related to hardware, software and connectivity.

As described in Table A.2, selecting fault-tolerant hardware is one of the tactics presented by IoT ARM framework to increase resiliency of an application. Measurements of nodes can be affected by the environment such has humidity or temperature changes. For example, if relative humidity reaches 100% it can break sensors due to water condensation. It is possible to prevent this from happening by heating up the sensors to keep sensor temperature over the ambient temperature. This increases the cost of a single node and for this application, keeping the costs low had higher priority than fault-tolerant hardware. Connectivity issues of measurement nodes can be reduced by fault-tolerant software, when the hardware is not broken.

It was noticed that the nodes got disconnected from the LoRaWAN network periodically. The 10m distance node stayed connected up to 12.1 days and the 100m node up to 2.3 days. The nodes do not wait for downlink confirmations after sending a measurement to conserve energy. This however causes that a node cannot verify if it still connected to the network or not. Need for autonomous operation outweights the increased battery lifetime received from not listening downlink messages. Therefore, application software should be improved to notice when a node disconnects from the network and automatically reconnect without need for an human operator. It can be stated that REQ5 was partially achieved, since the periodic measurements from the nodes were successfully delivered end-to-end but node software could not handle disconnection from the network.

6 Conclusion and future development

Cost-efficient environment measurement application for battery-operated IoT devices using IoT Architecture Reference Model (IoT ARM) was successfully developed in this thesis. The main goal for the application was to be able to monitor building conditions so that users can use the information to improve the energy efficiency of the building under monitoring. Before the start of development, requirements were collected from IoT ARM as well as the published market research commissioned by Microsoft. Development experience of the author also played a role in defining the requirements. The architecture was divided into four-stages with each stage explained in detail using IoT ARM guidelines.

Low-power wide area network technologies were researched to find out the best suitable communication technology for the application. The considered technologies were LoRaWAN, sigfox and NB-IoT. LoRaWAN was chosen due to low deployment costs and because the cost of expanding coverage is cheaper than for the other technologies. Once the communication technology was chosen, the hardware of the measurement nodes could be selected. Instead of developing the hardware for scratch, Arduino MKR WAN 1310 with built-in LoRaWAN connectivity was selected to reduce the development time. Arduino also offers environment measurement shield (MKR ENV) that can be attached to MKR WAN 1310. There are more accurate measurement sensors than what MKR ENV can offer but sensors with better accuracy would increase the cost of a single node. The developed application is flexible and more accurate sensors can easily be added, if necessary.

The developed application was evaluated with the requirements and needed improvements in mind. All the requirements were either found to be fully or partly achieved. As expected, LoRaWAN technology is not able to guarantee 100% success rate of transferring measurements. Around 1% of measurements send by the nodes were lost and the loss rate increased when a node was placed farther away from the gateway.

Based on the evaluation results below improvements are required. Improved node activity diagram is shown in Figure 6.1.

- Nodes: Listen for downlink messages from the server
- Nodes: Reconnect to LoRaWAN network if connection cannot be verified
- Nodes: Save measurement results to memory and re-transmit lost measurements
- Server: Send downlink message to confirm received measurements every 3 hours

Listening downlink messages from servers is needed for two reasons: To confirm whether the measurements have reached the server and whether the connection to LoRaWAN network is still working. If connection to the network cannot be confirmed, reconnecting process needs to be started. Arduino MKR ENV shield has a built-in micro SD card slot which can be used to save measurement data[6]. Measurement data can be then re-transmitted if measurements have been lost.

Server software must confirm all received measurements every 3 hours. Due to duty cycle restrictions, it is not possible to confirm every measurement separately. Instead, all measurements received between the confirmation messages must be confirmed with a single message. Nodes can then re-transmit only the lost measurements. To enable this, the LoRaWAN payload needs to be updated to include a sequence number.

IoT ARM framework proved to be very useful in designing and describing the application. In addition, activities and tactics listed in Appendix A gives IoT developers a great checklist what to consider during design and maintenance phases. IoT application requires continues development. New functions are needed when new technologies and use cases emerges. Information security brings its own challenges since battery-operated IoT devices usually have lower processing capabilities than fully powered devices. This means that encryption methods used with battery-operated devices cannot require excess amount of calculation which lowers the security provided. Nevertheless, the developed application is ready as minimum viable product (MVP) and the development will continue to keep it secure and relevant in the future.

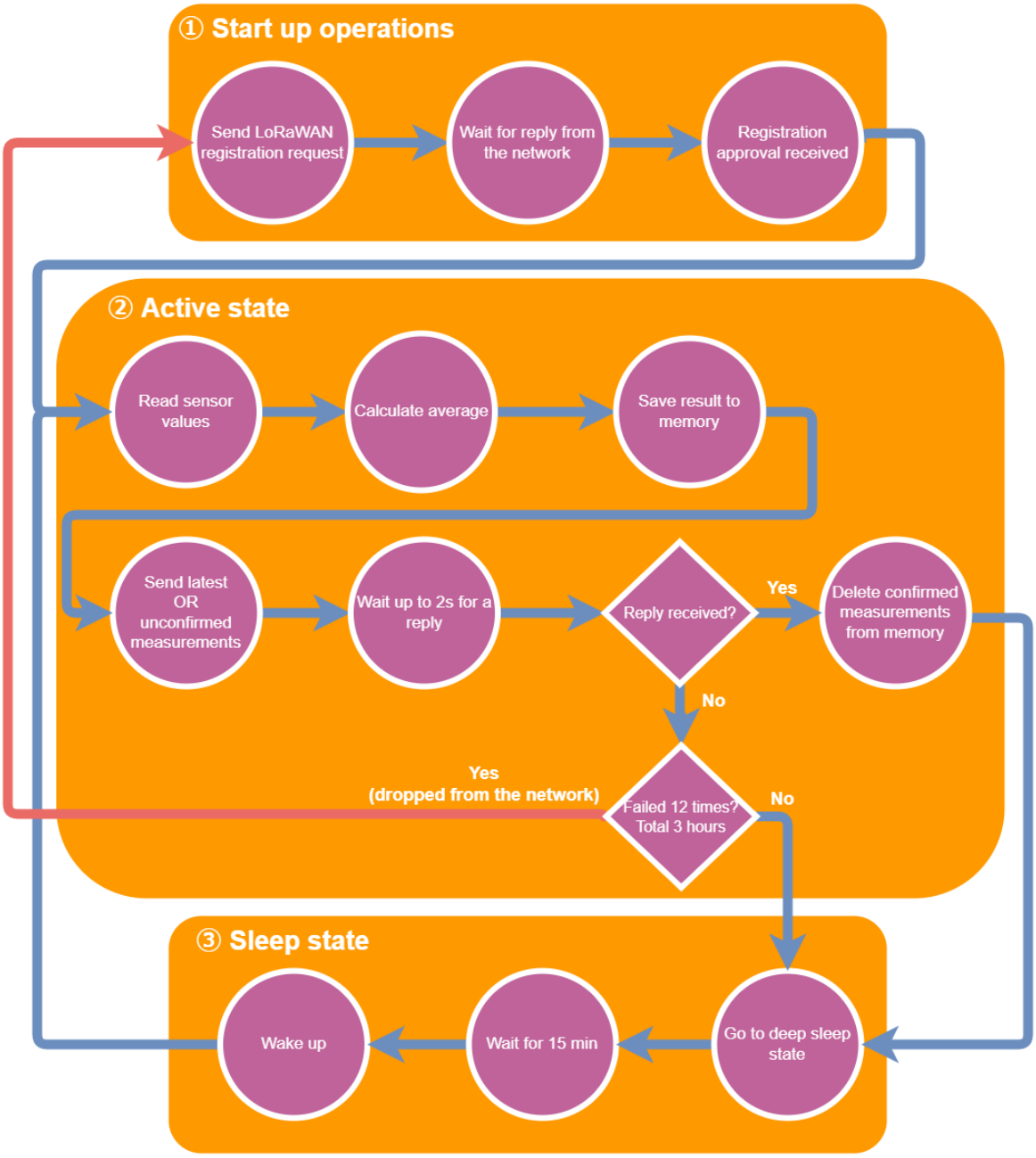

Figure 6.1: Improved node activity diagram

References

[1] 3RD GENERATION PARTNERSHIP PROJECT. About 3GPP Home. URL https://www.3gpp.org/about-3gpp/about-3gpp, accessed on 23 February 2022.

[2] ADELANTADO, F., VILAJOSANA, X., TUSET-PEIRO, P., MARTINEZ, B., MELIA-SEGUI, J., AND WATTEYNE, T. Understanding the limits of LoRaWAN. *IEEE Communications magazine 55*, 9 (2017), 34–40.

[3] AGARWAL, P., AND ALAM, M. Open service platforms for IoT. In *Internet of Things (IoT)*. Springer, 2020, pp. 43–59.

[4] AMAZON. AWS IoT. URL https://aws.amazon.com/iot/, accessed on 11 March 2022.

[5] AMAZON WEB SERVICES. Reliability Pillar - AWS Well-Architected Framework, July 2020.

[6] ARDUINO. Arduino MKR ENV shield. URL https://docs.arduino.cc/hardware/mkr-env-shield, accessed on 28 October 2022.

[7] ARDUINO. MKRWAN 1310. URL https://docs.arduino.cc/hardware/mkr-wan-1310, accessed on 28 October 2022.

[8] ASKSENSORS. AskSensors: Take full advantage of your IoT devices. URL https://asksensors.com/, accessed on 11 March 2022.

[9] ATZORI, L., IERA, A., AND MORABITO, G. The internet of things: A survey. *Computer networks 54*, 15 (2010), 2787–2805.

[10] BASSI, A., BAUER, M., FIEDLER, M., KRAMP, T., VAN KRANENBURG, R., LANGE, S., AND MEISSNER, S. *Enabling things to talk*. Springer Nature, 2013.

[11] CALLEBAUT, G., LEENDERS, G., VAN MULDERS, J., OTTOY, G., DE STRYCKER, L., AND VAN DER PERRE, L. The Art of Designing Remote IoT DevicesTechnologies and Strategies for a Long Battery Life. *Sensors 21*, 3 (2021), 913.

[12] DIGI-KEY ELECTRONICS. GATEWAY WIFI LORA ETHERNET EU. URL https://www.digikey.fi/fi/products/detail/laird-connectivity-inc/RG186/7356130, accessed on 3 March 2022.

[13] DIGI-KEY ELECTRONICS. MICRO BASE STATION SMBS-T4 PACK. URL https://www.digikey.com/en/products/detail/sigfox/10000619/10718492, accessed on 3 March 2022.

[14] FARRELL, S. Low-Power Wide Area Network (LPWAN) Overview. *Internet Engineering Task Force* (May 2018).

[15] GSMA. NB-IoT Commercialisation Case Study: How China Mobile, China Telecom China Unicom Enable Million More IoT Devices, 2019.

[16] HU, G., YI, Z., LU, L., HUANG, Y., ZHAI, Y., LIU, J., AND YANG, B. Self-powered 5G NB-IoT system for remote monitoring applications. *Nano Energy 87* (2021), 106140.

[17] INTERNATIONAL BAR CODING. Solution-as-a-service. URL https://ibcworld.net/news/explained-solution-as-a-service-solaas-is-it-right-for-you/, accessed on 11 March 2022.

[18] INTERNET OF THINGS - ARCHITECTURE. Requirements, 2022. URL https://www.iot-a.eu/public/requirements/, accessed on 24 May 2022.

[19] JAHNKE, A. The 4 Stages of IoT Architecture. URL https://www.digi.com/blog/post/the-4-stages-of-iot-architecture, accessed on 25 May 2022.

[20] KIWI TECHNOLOGIES. AIOT SaaS Cloud Platform. URL https://www.kiwi-tec.com/en/products/aiot/, accessed on 11 March 2022.

[21] KYOCERA. Sigfox Access Station Micro rental started, January 2019. URL https://www.kccs.co.jp/news/release/2019/0130/, accessed on 3 March 2022.

[22] LOCKL, J., SCHLATT, V., SCHWEIZER, A., URBACH, N., AND HARTH, N. Toward trust in Internet of Things ecosystems: Design principles for blockchain-based IoT applications. *IEEE Transactions on Engineering Management 67*, 4 (2020), 1256–1270.

[23] LoRa Alliance. LoRaWAN L2 1.0.4 Specification, October 2020.

[24] LoRa Alliance. 2021 Year End Report, February 2022.

[25] Malarski, K. M., Bardram, A., Larsen, M. D., Thrane, J., Petersen, M. N., Moeller, L., and Ruepp, S. Demonstration of NB-IoT for maritime use cases. In *2018 9th International Conference on the Network of the Future (NOF)* (2018), IEEE, pp. 106–108.

[26] Martinez, B., Adelantado, F., Bartoli, A., and Vilajosana, X. Exploring the performance boundaries of NB-IoT. *IEEE Internet of Things Journal 6*, 3 (2019), 5702–5712.

[27] Mekki, K., Bajic, E., Chaxel, F., and Meyer, F. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT express 5*, 1 (2019), 1–7.

[28] Meticulous Market Research Pvt. Ltd. Agriculture IoT Market Worth $32.75 Billion by 2027- Market Size, Share, Forecasts, Trends Analysis Report with COVID-19 Impact. *GlobeNewswire* (March 2021).

[29] Microsoft. Azure IoT: Quickly turn your vision into reality with secure, scalable, and open edge-to-cloud solutions from the Microsoft Cloud. URL `https://azure.microsoft.com/en-us/overview/iot/#overview`, accessed on 11 March 2022.

[30] Microsoft, hypothesis. IoT signals Edition 3, October 2021.

[31] Morrish, J. Global IoT market to grow to $1.5trn annual revenue by 2030. *IoTNow* (May 2020).

[32] Nauman, A., Qadri, Y. A., Amjad, M., Zikria, Y. B., Afzal, M. K., and Kim, S. W. Multimedia Internet of Things: A comprehensive survey. *IEEE Access 8* (2020), 8202–8250.

[33] Olivia. Vodafone Narrowband IoT. URL `https://www.oliviawireless.com/narrowband-iot`, accessed on 23 February 2022.

[34] Rauma, E., Laitala, M., Talasmo, T., Harju, J., and Chaheh, F. Indoor Air Quality Measurement. *University of Jyväskylä* (June 2021).

[35] RH WIRELESS WORLD. LoRa Sensitivity Calculator | LoRa Sensitivity Formula. URL https://www.rfwireless-world.com/calculators/LoRa-Sensitivity-Calculator.html, accessed on 21 February 2022.

[36] SIGFOX. Buy Sigfox connectivity. URL https://buy.sigfox.com/buy, accessed on 3 March 2022.

[37] SIGFOX. KS Technologies: Amusement park trash can level monitoring. URL https://www.sigfox.us/casestudies/kst/, accessed on 23 February 2022.

[38] SIGFOX. Laager: Residential Water, gas, and electric Meter Monitoring. URL https://www.sigfox.us/casestudies/laager/, accessed on 23 February 2022.

[39] SIGFOX. Sigfox Use Cases. URL https://www.sigfox.us/use-cases/, accessed on 24 February 2022.

[40] SIGFOX. Introducing 0G network, September 2021.

[41] SIGFOX. Sigfox connected objects: Radio specifications, March 2022.

[42] SINGH, R. K., PULUCKUL, P. P., BERKVENS, R., AND WEYN, M. Energy consumption analysis of LPWAN technologies and lifetime estimation for IoT application. *Sensors 20*, 17 (2020), 4794.

[43] SINHA, R. S., WEI, Y., AND HWANG, S.-H. A survey on LPWA technology: LoRa and NB-IoT. *Ict Express 3*, 1 (2017), 14–21.

[44] ST. HTS221 - Capacitive digital sensor for relative humidity and temperature. URL https://www.st.com/resource/en/datasheet/hts221.pdf, accessed on 30 October 2022.

[45] ST. LPS22HB - MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer. URL https://www.st.com/resource/en/datasheet/dm00140895.pdf, accessed on 30 October 2022.

[46] THE THINGS NETWORK. Community - Let's build this thing together. URL https://www.thethingsnetwork.org/community, accessed on 3 March 2022.

[47] THE THINGS NETWORK. Fair Use Policy explained. URL https://www.thethingsnetwork.org/forum/t/fair-use-policy-explained/1300, accessed on 17 March 2022.

[48] THE THINGS NETWORK. The Things Network - Overview. URL https://www.thethingsnetwork.org/docs/quick-start/, accessed on 3 March 2022.

[49] UNITED NATIONS. Sustainable Development Goals. URL https://sdgs.un.org/goals, accessed on 11 March 2022.

[50] VAISALA. Temperature Humidity validation/mapping services. URL https://www.vaisala.com/sites/default/files/documents/VIM-GLO-CMS-Mapping-Service-Brochure-B212431EN.pdf.

[51] VAISALA. viewLinc Monitoring, Alarming and Reporting Software. URL https://www.vaisala.com/en/products/software/viewlinc, accessed on 11 March 2022.

[52] VISHAY. TEMT6000X01 - Ambient Light Sensor. URL https://www.vishay.com/docs/81579/temt6000.pdf, accessed on 30 October 2022.

[53] ZORBAS, D., PAPADOPOULOS, G. Z., MAILLE, P., MONTAVONT, N., AND DOULIGERIS, C. Improving LoRa network capacity using multiple spreading factor configurations. In *2018 25th International Conference on Telecommunications (ICT)* (2018), IEEE, pp. 516–520.

A Activities and tactics of IoT-A

Table A.1: How changes in requirements and evolving technologies are handled?[10, p. 207]

Activities
Characterize the evolution needs
Assess the current ease of evolution
Consider the evolution trade-offs
Rework the architecture

Tactics
Contain change
Create extensible interfaces
Preserve development environments
Apply metamodel-based architectural styles
Build variation points into the software
Use standard extension points
Achieve reliable change
Apply design techniques that facilitate change

Table A.2: What can go wrong and how to deal with problems? [10, p. 211]

Activities
Produce the availability schedule
Capture the availability requirements
Estimate platform availability
Estimate functional availability
Assess against the requirements
Rework the architecture

Tactics
Select fault-tolerant hardware
Relax transactional consistency
Log transactions
Apply software availability solutions
Select or create fault-tolerant software
Design for failure
Allow for component replication
Use high-availability clustering and load balancing
Identify backup and disaster recovery solution

Table A.3: How should the application perform at launch and when data volume increases?[10, p. 208]

Activities
Capture the performance requirements
Create the performance models
Analyze the performance model
Conduct practical testing
Assess against the requirements
Rework the architecture

Tactics
Optimize repeated processing
Reduce contention via replication
Prioritize processing
Consolidate related workload
Distribute processing over time
Minimize the use of shared resources
Reuse resources and results
Partition and parallelize
Scale up or scale out
Degrade gracefully
Use asynchronous processing
Relax transactional consistency
Make design compromises

Table A.4: How to information security issues are handled with resource constrained devices?[10, p. 210]

Activities	**Tactics**
Use an Authorization component to enable interoperability with other applications	Use an extended Internet Threat Model for which takes into account specific IoT communication vulnerabilities
Define security impact on interaction model	Harden infrastructural functional components
Conduct risk analysis	Authenticate subjects
Validate against requirements	Define and enforce access policies
Consider performance/security trade-offs	Secure communication infrastructure (gateways, infrastructure services)
Capture the security requirements	Secure communication between subjects
Check interoperability requirements for impacts on security processes between heterogeneous peers	Secure peripheral networks (data link layer security, network entry, secure routing, mobility and handover)
Address all aspects of Service and Communication Security	Avoid wherever possible wireless communication
Integrate the trust model and support privacy features	Physically protect peripheral devices or consider peripheral devices as available to malicious users in the attacker model
Identify security hardware requirements	Avoid Over-The-Air device management; if necessary secure properly
Use infrastructural or federated Key Exchange Management to secure communication initiation and tunnelling between gateways for interoperability	
Use infrastructural Authentication components that support more Identity Frameworks for scalability and interoperability	

B IoT-Architecture Unified Requirements

UNI ID	Category	Description	Rationale
UNI.005	Autonomicity, Data handling & Communication	A system built using the ARM shall support event-based, periodic, and/or autonomous communication	"The remote monitoring device gathers patient measurements, data and or events. Data may be communicated each time the device gathers the data, accumulated measurements may be communicated periodically (e.g., hourly, daily), or data may be delivered upon request or upon certain events"
UNI.008	Interoperability	A system built using the ARM shall be able to run applications and services in an interoperable manner	"The problem is to provide a framework, a set of scenarios where these applications could be developed in harmony, in an interoperable way and in a way that responses to the real needs of organization and people"
UNI.012	Radio-awareness, Data handling & communication, Resilience	A system built using the ARM shall be able to handle interference between IoT devices (avoidance and detection)	"In order to achieve a reliable eHealth service the system must be interference-free"

UNI.014	Autonomicity	A system built using the ARM shall support devices to activate themselves into a collaboration	"The remote monitoring device is prepared for use and communication by the action of the patient or clinician. This may involve physically attaching or placing the device, registering the device, setting up the communications channels to M2M application entities, setting up the communications capabilities of the device and providing for secure communications."
UNI.015	Resource control	A system built using the RA shall be able to remotely control and configure devices	"The remote monitoring device may be configured by via the M2M network by the M2M application entities. The configuration capability could span simple parametric changes, such as, reporting rates, event or alarm trigger levels, and dosing levels to downloading and securely restarting new operating software"

UNI.018	Data handling & communication	A system built using the ARM shall support data processing (filtering, aggregation/fusion, ...) on different IoT-system levels (for instance device level)	"The remote monitoring device gathers patient measurements, data and or events. Data may be communicated each time the device gathers the data, accumulated measurements may be communicated periodically (e.g., hourly, daily), or data may be delivered upon request or upon certain events"
UNI.031	Service composition & programmability	A system built using the ARM shall enable centralized or decentralized automated activities (control loops)	"Today, due to sub-optimal processes, a lot of time and money is wasted. This situation could be improved a lot by tracking all the items/things, providing context data on them at any time and location, allowing for automated evaluation of the collected data and reacting immediately on a dangerous situation to protect against the break down of items."
UNI.040	Security, Resilience	A system built using the ARM shall provide ways to ensure security and resilience	"Road users and energy providers want to avoid shortages/ blackouts"

UNI.041	Data handling & communication	A system built using the ARM shall provide historical information about the physical entity	"A method for clarification whether the Cold/Hot Chain has been violated or not is required. To be able to do this, the continuous context information (e.g., temperature) of the things needs to be collected. This is for example of major importance to avoid any damage to the pharmaceutics during the transport and storage process."
UNI.043	Service composition & programmability	A system built using the ARM shall enable the composition of entity-related services	"The costs for complex logistics and healthcare processes need to be kept on a low level. A modular setup of the applications and services is one important ingredient to achieve this. Therefore it should be very easy to integrate things together with their atomic services into other services, and it should be easy for things to use services provided by others."

UNI.047	Interoperability	The system must ensure interoperability between objects or between applications	"As an example, CCTV system could inform traffic management of the length of the waiting queue at a crossroad. Having smart traffic lights receiving such input from the CCTV system could, could help changing the schedule of green/red light to optimize the traffic."
UNI.050	Discovery & lookup, Geolocation	A system built using the ARM shall support mobility of the physical entity	"The use of M2M Devices for monitoring health related information is not confined to the residence of the patient."
UNI.058	QoS, Availability, Resilience	The system shall provide high availability	"Communication blackouts are not accepted from client side and particularly if they are paying for premium services"

UNI.062	Security, Trust, Data handling & communication Availability, Integrity	A system built using the ARM shall provide trusted and secure communication and information management	"A method for clarification whether the Cold/Hot Chain has been violated or not is required. To be able to do this, the detailed context information (e.g., temperature) of the things, which have been collected in some database need to be easily made available. This is for example of major importance to avoid any damage to the pharmaceutics during the transport and storage process."
UNI.065	QoS, Reliability	A system built using the ARM shall provide reliable services	"In order to accommodate certain scenario, support of a certain degree of reliability might be necessary"
UNI.070	Self-description, Semantics	A system built using the ARM shall provide interoperability through the use of semantics	"I would like a way to create and exchange semantics between objects in order to design new applications"

UNI.071	Data handling & communication, Semantics, Interoperability	A system built using the ARM shall provide standardized and semantic communication between services	"Standard communications between objects, from a communication channel point of view but also from a semantic point of view. (Standardization of object semantic is somehow similar to the standardization of MIB (Management Information Base) of telecommunication equipments)."
UNI.087	Service composition & programmability	A system built using the ARM shall support service lifecycle management	"A service is something that originates at a certain time for accomplishing goalsthat lives for a certain reason, may endure for a time, and may or may not be retired"
UNI.093	Interoperability, Extensibility	A system built using the ARM shall be extensible for future technologies.	"The reference architecture shall provide an integral approach that combines legacy aspects as well as an imaginating vision on the Internet of Things."

UNI.095	Data handling & communication, Addressing	A system built using the ARM shall include an interface to IP communication protocols.	"The reference architecture shall consider that we have gateways to IP everywhere, so we must have a global addressing system with protocol and so on. That would be an evolution of IPv6. Or we need an integration package for existing addressing systems."
UNI.097	Data handling & communication	A system built using the ARM shall support information (data) lifecycle management.	"Deal with the lifecycle of information (how to distinguish, if information (tag) is temporary not available or not valid any more?)"
UNI.100	Resource control	A system built using the ARM should include means to wake-up sleepy devices.	"We must look out also for some way to wake up sleepy communications in order to manage energy consume."
UNI.101	Energy-awareness, Resource Control	A system built using the ARM shall include means to manage the energy consumption of devices.	"We must look out for a highly energy efficient system."
UNI.102	Interoperability, Enterprise Integration, Extensibility	A system built using the ARM shall take into account external computing resources, e.g. 'the cloud'.	"Maybe there should be some part of processing information in the cloud."

UNI.241	Interoperability, Resource Control	A system built using the ARM shall provide unified interfaces to access and query the observation and measurement data emerging from resources	This will enable integration of IoT data into business layer and high-level applications.
UNI.308	Data handling & communication, Resilience	A system built using the ARM shall support communication over Low-power and Lossy networks.	In some IoT scenarios, IoT-devices need to communicate across an non-reliable networks, like Low-power and Lossy networks. There is a wide scope of application areas for LLNs, including industrial monitoring, building automation (HVAC,lighting, access control,fire), connected homes, healthcare, environmental monitoring, urban sensor networks (e.g. Smart Grid), asset tracking.
UNI.310	Data handling & communication, Energy-awareness	A system built using the ARM shall support power-constrained devices (in particular the latter may be able to participate in communication).	Communication must be open also for devices with constrainend power, like battery-powered devices.

UNI.313	Data handling & communication, Interoperability	A system built using the ARM shall support mapping from different addressing schemes to IP v6 (Compatible-addressing-scheme support).	Communication with non-IPv6 networks must be possible (IPv4, others).
UNI.324	Data handling & communication, Availability	Devices must be able to switch between different networks when moving, e.g. roaming. (Mobility support)	To guaranty continuity of service, IoT devices may need to change between similar networks without losing ongoing connections
UNI.326	Data handling & communication, Interoperability	The system shall support interworking between different application protocols (what to communicate), for example through translation functionality in an IoT gateway.	Seamless communication between all kinds of applications,e.g. business, desktop and mobile applications, must be possible.

C MKR WAN 1310 source code

```
#include <MKRWAN_v2.h>
#include <ArduinoLowPower.h>
#include <Arduino_MKRENV.h>
#include <DeviceAPPKey.h>

  LoRaModem modem;
  int sleepTime = 900000;
  int averageCount = 5;
  String appEui = "0000000000000000";
  String devAddr;
  String nwkSKey;
  String appSKey;

  //Measurement values
  String temperature;
  String humidity;
  String pressure;
  String illuminance;
  String uva;
  String uvb;
  String uvIndex;

void setup() {
  delay(5000);
   pinMode(LED_BUILTIN, OUTPUT);
   digitalWrite(LED_BUILTIN, HIGH);
  Serial.begin(115200);
  Serial.println("Starting module");
  // Start LoRa Module and select frequency
  if (!modem.begin(AU915_TTN)) {
    Serial.println("Failed to start module");
    while (1) {}
```

```
  };

  //Start environmental module
  if (!ENV.begin()) {
    Serial.println("Failed to initialize MKR ENV shield!");
    while (1);
  }
  int connected;

    appKey.trim();
    appEui.trim();
    Serial.println("Joining to LoRa network");
    connected = modem.joinOTAA(appEui, appKey);
  if (!connected) {
    Serial.println("Failed to connect to LoRa network");
    while (1) {}
  }
  else
  {
    digitalWrite(LED_BUILTIN, LOW);
    Serial.println("Connected to LoRa network");
  }

  delay(5000);

  modem.setPort(3);
}

void loop() {
  triggerMeasurement(averageCount);
  Serial.println("Sending...");
  modem.beginPacket();
  modem.print(temperature + " " + humidity + " " + pressure  + " "
  + illuminance);
  modem.endPacket(false);
  Serial.println("Sent");
  LowPower.sleep(sleepTime);
```

```
}

//Trigger measurements and calculates the average
void triggerMeasurement(int count)
{
  double tmpTemperatureAVG;
  double tmpHumidityAVG;
  double tmpPressureAVG;
  double tmpIlluminanceAVG;
  for(int i = 0; i < count;i++)
  {
  delay(1000);
  //Read measurment values from the MRK ENV module
  double tmpTemperature = ENV.readTemperature();
  double tmpHumidity    = ENV.readHumidity();
  double tmpPressure    = ENV.readPressure();
  double tmpIlluminance = ENV.readIlluminance();
  Serial.println("Temperature: "+String(tmpTemperature)+
  ",Humidity: " + String(tmpHumidity) + ",Pressure: " +
  String(tmpPressure)  + ",Illuminance: " + String(tmpIlluminance));
  tmpTemperatureAVG += tmpTemperature;
  tmpHumidityAVG    += tmpHumidity;
  tmpPressureAVG    += tmpPressure;
  tmpIlluminanceAVG += tmpIlluminance;
  }

  //Calculate average measurement values
  temperature = AVGToString(tmpTemperatureAVG,1,count);
  delay(10);
  humidity = AVGToString(tmpHumidityAVG,1,count);
  delay(10);
  pressure = AVGToString(tmpPressureAVG,0,count);
  delay(10);
  illuminance = AVGToString(tmpIlluminanceAVG,0,count);
  Serial.println("--Averages--");
  Serial.println("Temperature: "+temperature+
  ",Humidity: " +humidity +
```

```
  ",Pressure: " + pressure  +
  ",Illuminance: " + illuminance);
  Serial.println("------------");

}

String AVGToString(double number, int rounding, int measurementCount)
{
  double tmpNumber = number / measurementCount;
  String returnString = String(tmpNumber,rounding);
  if (returnString.length() < 5)
  {
    //Calculation correct
    return returnString;
  }
  else
  {
    //Calculation incorrect
    return "nan";
  }
}
```

D REST API example reply


```
{
"Measurement1" : { "_id" : { "$oid" : "633e16f43eace935329e948d" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.9, 66.3, 102.0, 103.0],
"MeasurementTime" : "Oct 6, 2022, 5:44:51 PM" },

"Measurement2" : { "_id" : { "$oid" : "633e1a7e3eace935329e9491" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.4, 67.4, 102.0, 80.0],
"MeasurementTime" : "Oct 6, 2022, 5:59:57 PM" },

"Measurement3" : { "_id" : { "$oid" : "633e1e083eace935329e9495" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity","Pressure", "Illuminance"],
"MeasurementValues" : [19.4, 66.9, 102.0, 65.0],
"MeasurementTime" : "Oct 6, 2022, 6:15:03 PM" },

"Measurement4" : { "_id" : { "$oid" : "633e21923eace935329e9499" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.4, 67.3, 102.0, 95.0], "MeasurementTime" : "Oct 6,

"Measurement5" : { "_id" : { "$oid" : "633e251c3eace935329e949d" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
```

```
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.3, 68.5, 102.0, 97.0],
"MeasurementTime" : "Oct 6, 2022, 6:45:15 PM" },

"Measurement6" : { "_id" : { "$oid" : "633e28a63eace935329e94a1" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.3, 67.7, 102.0, 117.0],
"MeasurementTime" : "Oct 6, 2022, 7:00:21 PM" },

"Measurement7" : { "_id" : { "$oid" : "633e2c303eace935329e94a5" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.2, 67.9, 102.0, 92.0],
"MeasurementTime" : "Oct 6, 2022, 7:15:27 PM" },

"Measurement8" : { "_id" : { "$oid" : "633e2fba3eace935329e94a7" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.2, 68.5, 102.0, 93.0],
"MeasurementTime" : "Oct 6, 2022, 7:30:33 PM" },

"Measurement9" : { "_id" : { "$oid" : "633e33443eace935329e94a9" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature","Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.2, 68.7, 102.0, 111.0],
"MeasurementTime" : "Oct 6, 2022, 7:45:39 PM" },

"Measurement10" : { "_id" : { "$oid" : "633e36ce3eace935329e94ab" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.2, 69.2, 102.0, 52.0],
```

```
"MeasurementTime" : "Oct 6, 2022, 8:00:45 PM" },

"Measurement11" : { "_id" : { "$oid" : "633e3a583eace935329e94ad" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.1, 69.1, 102.0, 53.0],
"MeasurementTime" : "Oct 6, 2022, 8:15:51 PM" },

"Measurement12" : { "_id" : { "$oid" : "633e3de23eace935329e94af" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.0, 68.9, 102.0, 76.0],
"MeasurementTime" : "Oct 6, 2022, 8:30:57 PM" },

"Measurement13" : { "_id" : { "$oid" : "633e416c3eace935329e94b3" },
"DevEUI" : "0000000000000000",
"Description" : "mkr-wan-1310",
"ChannelTypes" : ["Temperature", "Humidity", "Pressure", "Illuminance"],
"MeasurementValues" : [19.0, 68.9, 102.0, 88.0],
"MeasurementTime" : "Oct 6, 2022, 8:46:03 PM" }
}
```