TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Indrek Guitor 182555IASM

DEVELOPMENT OF IOT DEVICE FOR HUMAN ACTIVITY DETECTION NEARBY SMART ELEVATOR AREA

Master's thesis

Supervisor: Uljana Reinsalu
Researcher, Ph.D

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Indrek Guitor 182555IASM

ASJADE INTERNETI SEADME ARENDAMINE INIMLIIKUMISE TUVASTAMISEKS TARGA LIFTI LÄHEDUSES

magistritöö

Juhendaja: Uljana Reinsalu
Ph.D

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Indrek Guitor

28.04.2020

Abstract

In this thesis, wireless Internet of Things (IoT) device is proposed for KONE smart elevator in the Tallinn University of Technology. The outcome of this work is an IoT device prototype, that can detect elevator passengers and, could be used to make automatic reservation calls for them.

This IoT device relies on a passive infrared (PIR) sensor to detect movement and it uses Narrowband Internet of Things (NB-IoT) technology to communicate with the elevator server. Moreover, additional solution based on thermal camera to detect only elevator passengers instead of every movement, is proposed.

Analysis of the total time delay and power consumption of the device have been made. The results show that prototyped IoT device, especially NB-IoT part, should be more investigated because: the power consumption is too high, and the long time delay would not make reservation calls reasonable for the elevator passengers.

This thesis is written in English and is 67 pages long, including 10 chapters, 30 figures and 16 tables.

Annotatsioon

ASJADE INTERNETI SEADME ARENDAMINE INIMLIIKUMISE TUVASTAMISEKS TARGA LIFTI LÄHEDUSES

Käesolevas magistritöös pakutakse välja juhtmevaba asjade interneti seade KONE targale liftile Tallinna Tehnikaülikoolis. Tehtud töö tulemusena loodi asjade interneti seadme prototüüp, mis suudab tuvastada lifti kasutajad ning teeks nende eest automaatse lifti kutse.

Asjade interneti seade kasutab PIR liikumisandurit, et tuvastada potentsiaalse lifti kasutaja liikumist, ja NB-IoT tehnoloogiat, et suhelda lifti serveriga. Lisaks pakuti välja termokaameral põhinev lahendus, mis tuvastaks ainult lifti kasutajaid, mitte möödakäijaid.

On tehtud analüüs seadme energiatarbest ja ooteajast. Tulemused näitavad, et asjade interneti seadme prototüüpi, spetsiifilisemalt NB-IoT tehnoloogiat, tuleks täpsemalt uurida. Selle põhjuseks on suur energiatarve ja pikk ooteaeg, mis teevad lifti automaatse kutsumise ebaefektiivseks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 67 leheküljel, 10 peatükki, 30 joonist, 16 tabelit.

List of abbreviations and terms

IoT	Internet of Things
LED	Light-Emitting Diode
IP	Internet Protocol
PIR	Passive Infrared
RFID	Radio Frequency Identification
RGB	Red, Green, Blue colour system
MEMS	Micro-Electro-Mechanical Systems
FOV	Field of View
3GPP	3rd Generation Partnership Project
NB-IoT	Narrowband Internet of Things
LPWAN	Low-Power Wide Area Network
LTE	Long-Term Evolution
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
PPP	Point-to-Point Protocol
MQTT	MQ Telemetry Transport
NIDD	Non-IP Data Delivery
HTTP	The Hypertext Transfer Protocol
FTP	File Transfer Protocol
SSL	Secure Sockets Layer
APN	Access Point Name
GSM	Global System for Mobile Communication
QoS	Quality of Service
Capex	Capital expenditure
OFDM	Orthogonal Frequency-Division Multiplexing
SC-FDMA	Single Carrier Frequency Division Multiple Access
PSM	Power Saving Mode
eDRX	Extended Discontinuous Reception

MCL	Maximum Coupling Loss
UART	Universal Asynchronous Receiver-Transmitter
ADC	Analog-to-Digital Converter
PLA	Polylactic Acid
RAT	Radio Access Technology
PDP	Packet Data Protocol
DNS	Domain Name System
NTP	Network Time Protocol
AES	Advanced Encryption Standard
SNR	Signal-to-Noise Ratio
ASIC	Application-Specific Integrated Circuit
I2C	Inter-Integrated Circuit
PEC	Packet Error Code
CRC	Cyclic Redundancy Check
PNG	Portable Network Graphics
CSV	Comma Separated Values
PDN	Packet Data Network
TAU	Tracking Area Update
RRC	Radio Resource Control
URC	Unsolicited Result Code
RF	Radio Frequency
MCU	Microcontroller Unit

Table of contents

List of figures

List of tables

1 Introduction

As the technology advances, new solutions are being sought to integrate technology into everyday life. For instance, elevators are being used every day to facilitate people's lives, especially in mid- and high-rise buildings. As a new solution, smart elevators have been introduced. Mostly "smart" elevator systems in buildings means that the system is using algorithms that should reduce peak-time congestion and overall passenger commuting times by grouping people with similar destinations into the same elevator. This improves the performance of the traditional elevators in terms of efficiency, flexibility, time, convenience and access control for the building owners and passengers [1].

Reduced waiting time and energy consumption are the main research areas for smart elevator systems. However, this is mostly done by modifying the elevator cabin driving algorithm: where to pick up passengers first, at what time the elevator should be at which floor etc. On the other hand, elevators are being developed smarter in many other ways as well - by attaching different sensors to the elevator system or by using artificial intelligence to provide better experience to end-users.

The objective of this master's thesis is to investigate and develop a wireless set of low power Internet of Things (IoT) devices for smart elevator systems. As a whole solution, each floor will have one IoT device placed above the elevator door, that detects elevator passengers and should make automatic reservation calls for them in the future. This paper analyses the adequacy of this IoT device above the smart elevator door to detect elevator passengers. This study is based on further development of a KONE smart elevator system set up in a TalTech ICT building.

This thesis is composed of 10 main chapters and one appendix. Chapter 1 includes a historical background, state of the art and overview of the thesis objectives. Chapter 2 analyses the available options for human detection and communication between IoT device and elevator server. Chapter 3 introduces the selected communication technology. Chapters 4 and 5 describe the selected hardware and security measures used for the application server. Then, in chapter 6, different NB-IoT technology power saving modes

were tested and analysed. Chapters 7 and 8 describe the analysis results for the proposed IoT device and for thermal camera solution. In chapter 9 possible suggestions for the future investigations are given. Finally, an overview of the results is given in the last chapter.

1.1 History of elevators

Elevator technology has been evolving since 236 BC, where Archimedes invented the first known elevator, which could be operated by humans or animals with pure muscle power. It was mostly used to lift heavy loads such as water or building materials. The first known elevators specifically for passengers were used in 18 century and they were called "the flying chair". The elevator system evolved further, first being powered by steam and then by hydraulic by using oil or water pressure. This was until 1887, when Alexander Miles was awarded with a patent for an electric elevator, which proved to be fast and practical since there were no height limits, compared to the previous elevator technologies [2].

The 20th century was an era of automation in the industry, which includes advancements in the elevator systems. For example, push buttons, advanced braking systems, magnetic inductor levelling, group controls, advanced design roller guides etc. were introduced to the elevator systems. In the later part of the century, "smart" elevator systems became available, as the technology advanced rapidly - integrated circuits, solid state door detector edges, light-emitting diodes (LED's), computer integrated controllers were introduced [2].

1.2 Related works

Various previous research on smart elevators have been proposed for improving the elevator scheduling efficiency, for instance, reducing the passengers waiting time, energy consumption etc. These studies have contributed primarily in optimizing the algorithm-based elevator's controller, e.g., fuzzy logic [3], genetic algorithm [4] and neural networks [5].

On the other hand, some examples of data gathering from sensors, mobile devices and user-interfaces can be brought out that focus on reducing passenger waiting time by placing the elevator call in advance.

Ge et al. [6] upgraded conventional elevator system into IoT based elevator system called Intellevator: A Context-Aware Elevator System for Assisting Passengers. Firstly, the agent server computational capability was increased and then end-user centred application was prototyped that provided the real-time context visualization and intelligence for assisting users, such as waiting time and advises. At the same time, another smart elevator system was proposed by Ge et al. [7] - PrecaElevator, that enables passengers to reduce their waiting time through pre-registering elevator calls in smart building. End-user centric system was developed where the elevator was callable if the passengers walking time to the elevator was shorter than the waiting time of the elevator to the departure floor. This intelligence of dynamic computing for the accessible range of elevator control allows the elevator to have high scheduling efficiency, as the excess calls without anybody riding the elevator decreases the scheduling efficiency of the smart elevator.

Mobile interaction with elevators is another great concept, which was explored by Turunen et al. [8]. In that study mobile application was developed that enables users to place elevator calls remotely inside an office building. The application handles corresponding elevator calls and it can be configured to provide quick access to common destinations, scheduled calls based on estimated walk time and continuous status information of the allocated elevators. This study proves that mobile interaction with elevators can be used to place the elevator calls in advance, thus shortening the perceived waiting time and expediting the movement of its users.

While the previous works allowed the users to place the elevator call in advance, it still needed human interaction. Kwon et al. [9] developed an efficient elevator scheduling system by making use of indoor sensor technologies, such as radio-frequency identification (RFID), video and floor sensors. In mentioned study, all three sensors are used together to detect elevator passengers automatically before they push the elevator call button. This information is sent to the elevator scheduling system through building networks.

Compared to previous works, this Master's thesis focuses on making the elevator calls automatic, without any effort required form the passenger, i.e., detecting elevator passengers and sending this information to the elevator server while using small, easily integrated wireless IoT devices.

1.3 Objectives

As the need of reducing passenger's waiting and moving time is growing, smart elevators are evolving further and further. Mostly elevator system algorithms are being researched, modified and developed to reduce passenger waiting time, but few researches are done to use sensors outside the elevator, to call the elevator beforehand, as mentioned in the previous paragraph.

This master's thesis consists of finding a solution to two major questions related to the smart elevator system. Firstly, how to detect humans, who want to enter the elevator, and secondly, how to send the analysed and processed information received from sensors to the elevator server, that is responsible for the elevator call system, taken into account difficult conditions for wireless signal sending in staircases.

Currently functioning KONE elevator, that is in the TalTech ICT building, was used for this Master's thesis. Since this elevator is being publicly used, the whole system has to be secured from threats, which means encrypting the sent out data, using identification keys for the sensors, opening only one specific elevator's server network port to the specified Internet Protocol (IP) address range to communicate with the IoT device etc.

It was chosen, that the solution should be a wireless IoT device, that could be easily added to the elevator system without changing elevators construction. However, this means that the solutions to the defined questions should also be energy efficient.

In order to solve these problems, following goals were set:

- Select wireless low power communication protocol
- Select sensors, that would detect elevator passengers
- Detect elevator passengers with selected sensors
- Select and design hardware for these IoT devices
- Implement software for these IoT devices

- Analyse power consumption of the IoT device by testing different power saving modes of the wireless communication module
- Analyse total time delay from detecting an elevator passenger to the elevator server receiving this packet from the IoT device
- Integrate IoT device with the elevator server (to collect data) wirelessly

2 Options available

There are many options available to find solutions to these two questions, however, every solution has its pros and cons, such as energy consumption, human recognition ability and simplicity to install it to the elevator system. These advantages and disadvantages of human detection options and sending data options are described below and selection of the most suitable technology to use is based on them.

2.1 Human detection options

Humans can be detected in many ways, starting from measuring carbon dioxide concentration in the air and using this data to estimate human presence to detecting smartphones with various on the market devices as most people have smartphones with them [10], [11], [12].

Cheaper and more widely used methods to detect humans are:

- Passive Infrared (PIR) motion sensor
- Thermal sensor
- Red Green Blue (RGB) camera
- Floor sensor
- RFID sensor
- Distance sensor
- Sound sensor

A **PIR motion sensor**, that consists of a pyroelectric sensor and lens can detect changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor. This means any kind of movement is detected with this sensor. On the downside, the measurement signal is lost during times of no movement in front of the sensor.

Using a Micro-Electro-Mechanical Systems (MEMS) **thermal sensor** that consists of a silicon lens and a thermopile sensor, the conventional pyroelectric sensor problem is eliminated, as the thermal sensors continue to generate a measurement signal during times of no movement. The principles this MEMS thermal sensor are following [13]:

- The silicon lens focuses radiant heat emitted from objects onto the thermopile sensor in the module.
- The thermopile sensor generates electromotive force in accordance with the radiant energy focused on it.
- The values of this electromotive force and the internal thermal sensor are measured.
- The device calculates the measured value (temperature of the object) via an interpolation calculation that compares the measured values with an internally stored lookup table.

RGB camera can be used to take the video feed of the elevator entrance. This video consists of images that are captured by the camera lens that focuses all the light rays to a single point to create a sharp image. This image should then be processed by convolution neural networks that are trained to detect humans in the pictures.

A completely different solution that is used to detect humans is using **floor sensors** [9]. This floor is similar to weight scales, as it detects weight change of about 50 grams and sends it to the location system to detect and track a human movement and even predict it. However, this solution is not cheap nor is it easily installable to everyday offices or buildings.

As mentioned in the related work chapter, **RFID sensors** could also be used to detect humans. RFID systems consist of three components: an RFID tag, an RFID reader, and an antenna. Integrated circuit and an antenna, which form an RFID tag, are used to transmit data to the RFID reader. The reader then converts the radio waves to digital data. Information collected from the tags is then transferred through a communications interface to a host computer system, where the data can be stored in a database and analysed. However, this only works in specific situations, where every person in the building has an RFID tag, for instance, security cards to get into the building and move between floors.

Another great way to detect humans and/or movement is to use a **distance sensor**, as it can detect when something is in front of the sensor. Distance sensors transmit a signal (ultrasonic waves/high frequency sounds) and wait for it to be reflected from the object and then receive the signals. Since the sound's velocity is known, it is possible to calculate

the distance to an object by using the time between the transmission and reception of the signal. Unfortunately, the high frequency sounds interference with microphones and other sound converting sensors such as hearing aids.

Possible, but very complicated solution to detect humans is to use **sound sensors**. Sound sensors consist of a capacitive microphone, peak detector and an amplifier, and with these components the sound is digitized. Sound waves cause diaphragm in the microphone to vibrate, resulting in capacitance change, which is then amplified and digitized for processing. However, this solution is only theoretical and very complicated, as it is difficult to distinguish background noise from elevator passengers talking and walking noise.

Comparison of these technologies is shown in Table 1 below and selection of the suitable technology to use is based on them.

Table 1. Comparison of human detection sensors.

Sensors	**Identifiable characteristics**	**Problems**	**Easily integratable in every office**	**Estimated power consumption**	**Detection distance**	**Detect stationary human**	**Field of view (FOV)**
PIR motion sensor (HC-SR505) [14]	Change components of infrared rays	Heat and light might trigger sensor	Yes	60 μA	3 m	No	100° in cone angle
Thermal sensor (D6T-32L-01A) [13]	Heat	Hard to detect passengers coming from outside, especially in the winter	Yes	19 mA	5-6 m	Yes	90° x 90°
RGB camera (Raspberry Pi Camera Module v2.1) [15]	Image	Very high power consumption, privacy invasion	Yes	Very high	10+ m	Yes	62.2° x 48.8°
Floor sensor	Weight	Not easily integratable	No	N/A	Direct contact	Yes	N/A
RFID sensor (MFRC522) [16]	RFID tag	Does not work in every building	No	6.5 mA	50 mm	No	N/A
Distance sensor (HC-SR04) [17]	Ultrasonic waves	Interfering with other sensors (microphone, hearing aid etc.), low FOV	Yes	15 mA	4 m	Yes	15°
Sound sensor (VMA309) [18]	Sound waves	Hard to distinguish background noise from passengers' noise	Yes	N/A	N/A	Yes	N/A

For this research, PIR motion sensor HC-SR505 and Thermal sensor D6T-32L-01A were selected to detect elevator passengers. The decision to use these sensors was based on the ease of the integration, as these modules are small and can be easily installed to the elevator system. Other factors that affected the decision include ensuring a high level of privacy and having satisfactory detection range. Besides that, this 32 x 32 matrix thermal sensor is a new promising solution to detect stationary and moving persons, which has not yet been used widely. This caught the author's interest to use the sensor in this solution.

2.2 Sending data options

When it comes to sending data from sensor device to elevator server, two possible categories are available: wired and wireless options. The first option would be to use wired communication, as it is more secured, and it can transmit data faster. For example, using the KONE elevator landing call button signal and its connections to the elevator server, one can add sensors to the KONE elevator system to trigger the elevator calling, without physically pressing the call button. However, it was chosen that the device should be wireless and easily added to the elevator system without changing its construction, which means this option was excluded.

On the other hand, there are many wireless solutions available to choose from, for instance, Wi-Fi, Bluetooth etc., but these consume a lot of power and have high frequency. High frequency does not spread far, nor does it penetrate through walls, which is why the comparison Table 2 for wireless IoT solutions was made.

Table 2. Comparison of low power wireless IoT technologies.

Technology	**LoRa**	**Sigfox**	**NB-IoT**	**LTE Cat M1**
Communication standards	LoRa Alliance	ETSI	3rd Generation Partnership Project (3GPP)	3GPP
Spectrum	Unlicensed	Unlicensed	Licensed	Licensed
Frequency band	868 MHz	868 MHz	832-862 MHz Transmit 791-821 MHz Receive (band 20)	880-915 MHz Transmit 925-960 MHz Receive (band 8)
Bandwidth	125, 250, 500 kHz	200 kHz	180 kHz	1.08 MHz
Transmit power	14 dBm	14 dBm	14/20/23 dBm	20/23 dBm
Data rate	50 Kbps FSK 11 Kbps LoRa	100/600 bps	250 Kbps (multi-tone) 20 Kbps (single tone)	1 Mbps
Maximum coupling loss (MCL)	155 dB	163.3 dB	164 dB	155.7 dB
Limitations	Separate gateway needed, 1% duty cycle	Slowest data rate, fixed data length, 140 messages per day maximum, only parts of Estonia covered with Sigfox	Theoretical latency up to 10 seconds	Not activated yet by Telia

Based on the Table 2, Narrowband Internet of Things (NB-IoT) was chosen, as there are less limitations and it outperforms all other Low-Power Wide-Area Networks (LPWAN) technologies in terms of coverage, energy efficiency, and cost by utilizing the existing cellular infrastructure.

3 NB-IoT

Narrowband Internet of Things (NB-IoT) is a wireless communication standard introduced by the Third-Generation Partnership Project (3GPP). NB-IoT is one of the licensed LPWAN cellular technologies based on Long-Term Evolution (LTE) [19]. Since LTE is globally known and used, supporting and driving IoT adoption through NB-IoT is a promising solution.

Like LTE, NB-IoT is based on orthogonal frequency-division multiple access with 180 kHz system bandwidth, which corresponds to one physical resource block in LTE transmission. NB-IoT allows long-range communications at low data rates and is most suitable for delay-tolerant applications. It can provide a data rate of 250 Kbps for multi-tone downlink communication and 20 Kbps for single-tone uplink communications.

However, many design changes such as retransmission for coverage extensions, synchronization sequences, random access preamble, scheduling delay for reducing computational complexity and power boosting for downlink transmissions have been introduced to ensure its best coexistence with the existing LTE infrastructure and to fulfil the need of IoT applications [20], [21], [22]. The detailed design changes with a key insight on the technology can be found in the standard [23].

3.1 Architecture

NB-IoT system setup consists of three functionality layers as shown in the Figure 1. The three layers of NB-IoT system are [21]:

- End node layer (IoT device is installed in the TalTech ICT building at the entrance of the elevator)
- Network layer (Telia's commercial Base-station is used, which is installed at TalTech campus)
- Application layer (KONE elevator server is set up as User Datagram Protocol (UDP)/Transmission Control Protocol (TCP) server that currently only stores the data end nodes transmit)

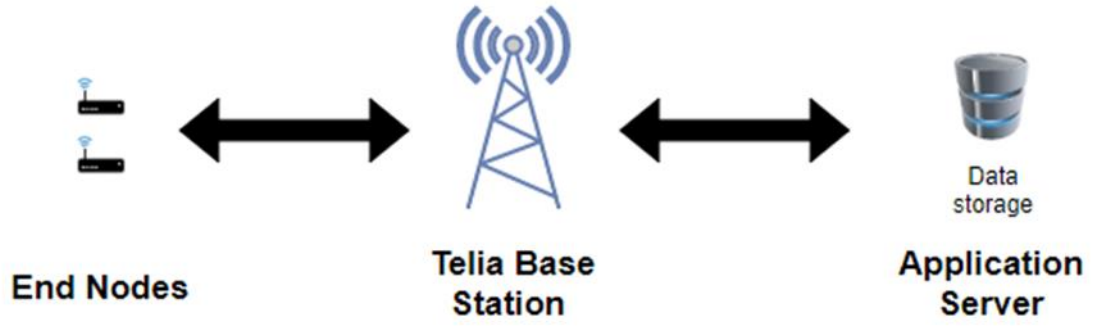

Figure 1. Three layers of NB-IoT system.

IoT devices are installed at the entrance of KONE elevator in the TalTech ICT building. This device detects humans with corresponding sensors and transmits the collected data to the next higher layer - Network layer. At the Network layer, a commercial Base-station operated by Telia, supports LTE Cat-NB1 network to collect all the transmitted data from the IoT devices and sends it further to the highest layer of the NB-IoT system, i-e., Application layer. In this Application layer received data is stored and it could be used in the future to make reservation calls for elevator passengers. Each of the NB-IoT system layer used is explained in detail as follows:

- The end node layer in chapter 4
- The network layer in chapter 3.2
- The application layer in chapter 5

3.2 Network layer

For the base station, Telia's commercially enabled NB-IoT base station with in-band deployment mode was used to test power saving modes, which are described in chapter 6. This base station also supports 5G and is installed at the premises of TalTech main building. The antenna of the base station installed at TalTech is mounted approximately at a height of 20 m from the sea-level [20], [21]. However, all of Telia's base stations over Estonia can be used to transmit data over NB-IoT. For example, power saving modes were tested in Tallinn and transmission delay, that is described in chapter 7.2, was tested in Tartu.

3.3 Data transfer protocols

There are many options to choose from when it comes to transferring data through NB-IoT systems. The available options for BG96 NB-IoT module are following [1]:

- Point-to-Point Protocol (PPP)
- MQ Telemetry Transport (MQTT)
- UDP
- UDP service
- TCP
- Non-IP Data Delivery (NIDD)
- Hypertext Transfer Protocol (HTTP)
- File Transfer Protocol (FTP)
- Secure Sockets Layer (SSL)

Unfortunately, some of these protocols cannot be used, as they are not available in the Telia network layer, for example, NIDD; or it is not possible to use in this scenario (PPP). As the UDP and UDP service consume the least amount of time and energy to transfer the data, these were the first choices. Sending UDP packets worked on a private network on randomly selected ports, however, transferring data from IoT device to the KONE elevator server did not work, as there are many security concerns - ports closed, access denied for outside network etc. TalTech network administrators granted access to specific IP-range, that the IoT device would get from Telia's Access Point Name (APN) and also opened a prearranged port for communication, however, UDP packets did still not go through and elevator server did not receive any UDP packets. On the other hand, the same network configuration worked for TCP packets and it was applied in this Master's thesis.

[1] NB-IoT module BG96 data transferring options are shown in the documentations available in https://gitlab.pld.ttu.ee/inguit1/Indrek_Guitor_Masters_Thesis/tree/master/BG96_AT_COMMANDS/Software

3.4 Mode of operation

To offer flexible deployment possibilities to network operators, NB-IoT can be deployed in three different modes, as shown in Figure 2 with the limited bandwidth requirement [24]:

- In-band mode
- Guard band mode
- Stand alone mode

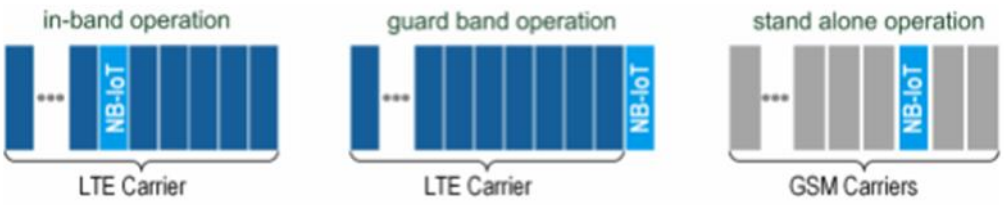

Figure 2. NB-IoT three different deployment modes.

In in-band and guard band modes, NB-IoT occupies one physical resource block of 180 KHz in LTE spectrum both in the downlink and uplink. However, in the stand alone mode it occupies the 200 KHz bandwidth by "reframing" the Global System for Mobile Communication (GSM) spectrum. These possibilities enable fast integration and coexistence with legacy LTE and GSM systems.

The choice of mode selection is critical and has an impact on the overall network coverage, quality of service (QoS) as well as capital expenditure (Capex). To support the flexibility and switching between these modes, NB-IoT extensively uses the LTE design such as OFDM in downlink and single carrier frequency division multiple access (SC-FDMA) in the uplink [20], [21].

3.5 Power reduction

NB-IoT devices are intended to have a 10 years battery life to support massive deployment with limited human intervention. To achieve this, two features i.e., Power Saving Mode (PSM) from 3GPP Release 12, and extended Discontinuous Reception (eDRX), new feature from 3GPP Release 13, were supported [20]. These features are used to extend the end device's battery longevity and they are more deeply explained in chapter 6.3.

3.6 Coverage enhancement method

For applications that are in hard-to-reach areas, such as undergrounds and indoors, NB-IoT technology for cellular IoT services is designed to enhance its coverage. Additional coverage of 20 dB is acquired as compared to the legacy LTE systems, which corresponds to 164 dB of maximum coupling loss (MCL). To enhance its coverage, NB-IoT uses up to 128 and 2048 retransmissions in uplink and downlink, respectively. This makes NB-IoT suitable for use cases that are latency insensitive as it can tolerate up to 10 seconds transmission delay. NB-IoT narrow bandwidth and its support for repetition are the key features to enable the enhanced coverage [20].

4 Selected Hardware

As two different methods were selected to detect humans, two different IoT solutions were proposed. First one is using PIR to detect humans and NB-IoT technology to send the data to the elevator server, and the second one is using Thermal camera to detect humans. Second device is made to see if it is possible to distinguish elevator passengers from passers-by on floors where more people walk past the elevator entrance. This solution ensures that the elevator is called for only the elevator passengers, not for passers-by.

4.1 PIR based IoT device

The first prototype for this PIR based IoT device is shown in the Figure 3, with its schematics in Figure 4. It consists of the following parts:

- 9V alkaline battery
- HC-SR505 PIR motion sensor
- P-Channel SQD19P06-60L mosfet
- Arduino Uno
- Avnet Silica NB-IoT Sensor Shield based on Quectel BG96
- Telia SIM card

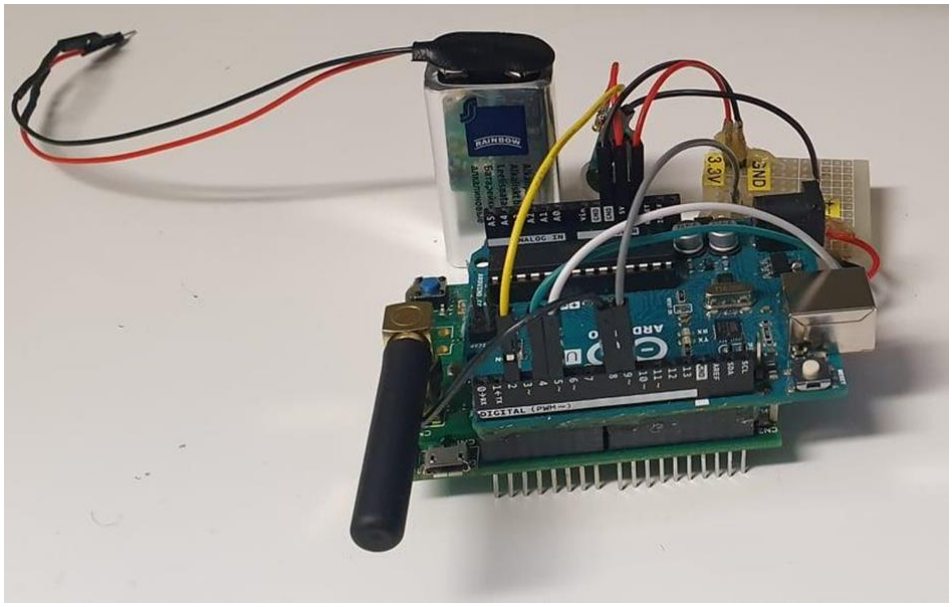

Figure 3. First prototype of the PIR based IoT device.

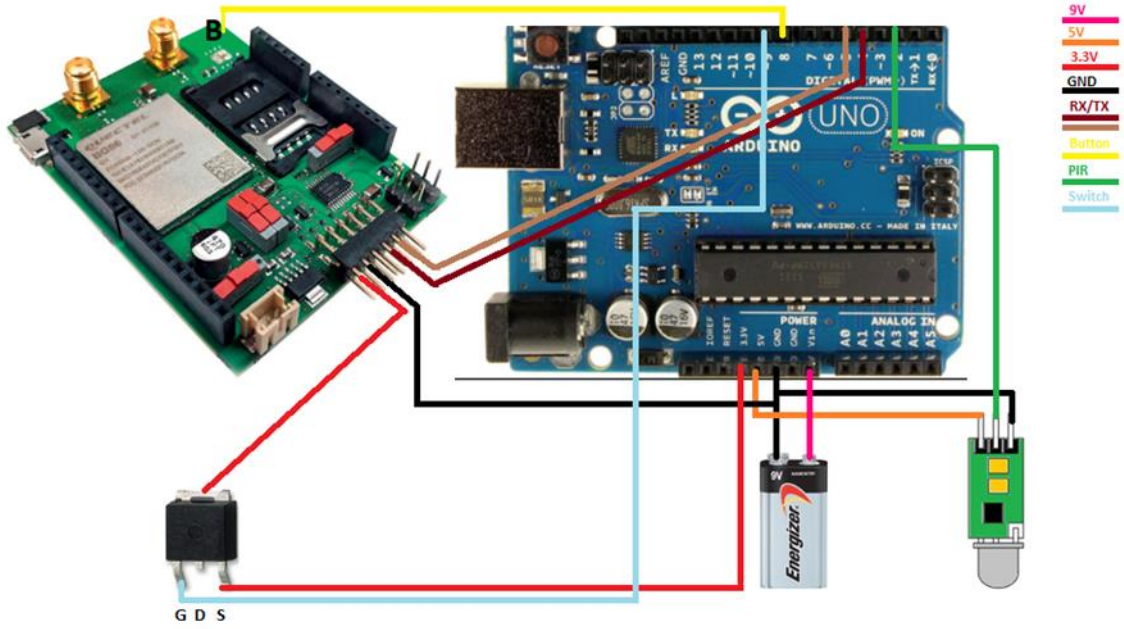

Figure 4. First PIR based IoT device schematics.

This IoT device detects movement with a PIR sensor and is communicating with the elevator server using NB-IoT technology. Unfortunately, this Avnet Silica NB-IoT sensor shield has no documentation about the pinout of the BG96 module to make it compatible with Arduino, as it was described it would be, which is why universal asynchronous receiver-transmitter (UART) communication had to be made using 4A type Pmod interface (expanded UART) [25]. The pinout of NB-IoT shield is also required to make additional connections between Arduino and NB-IoT sensor shield to test some of the NB-IoT low power modes, for instance, sleep mode or airplane mode.

This prototype uses a mosfet as a switch to control the status of the NB-IoT sensor shield: whether it needs to be powered on or powered off. In Figure 4, the B indicates a push button, which needs to be pressed for at least 100 ms to turn the module on. To make the push button press automatic, the button connection needs to be driven low (0V) by Arduino for at least 100 ms, which is done with a separate connection. This prototype is powered by a 9V battery. Since this Avnet silica NB-IoT sensor shield does not have the pinout defined and it is also obsolete, the second prototype was built.

The second prototype for this IoT device is shown in the Figure 5, with schematics in Figure 6 and it consists of the following parts:

- 2 Li-Ion 18650 3.6V batteries (in series)
- HC-SR505 PIR motion sensor
- N-Channel AUIRF540ZS mosfet
- Arduino Uno
- Dragino NB-IoT shield for Arduino based on Quectel BG96
- Telia SIM card
- 2 1.13k ohm resistors

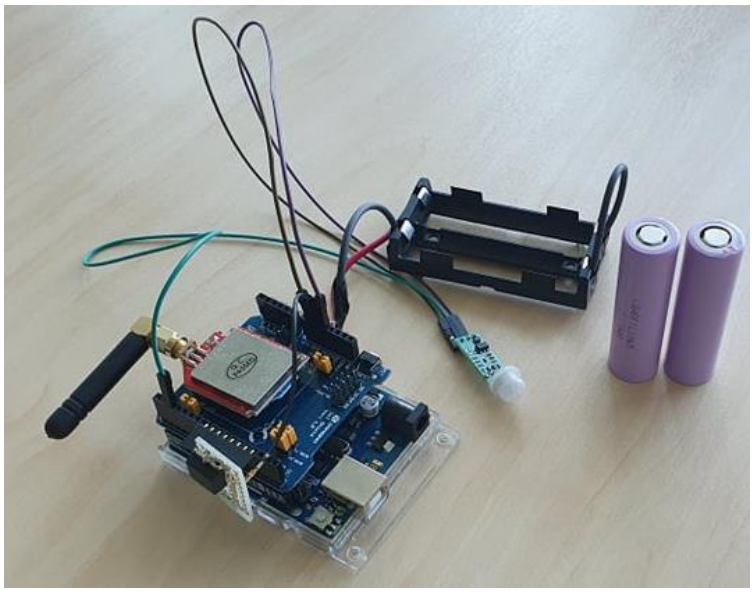

Figure 5. Second prototype of the PIR based IoT device.

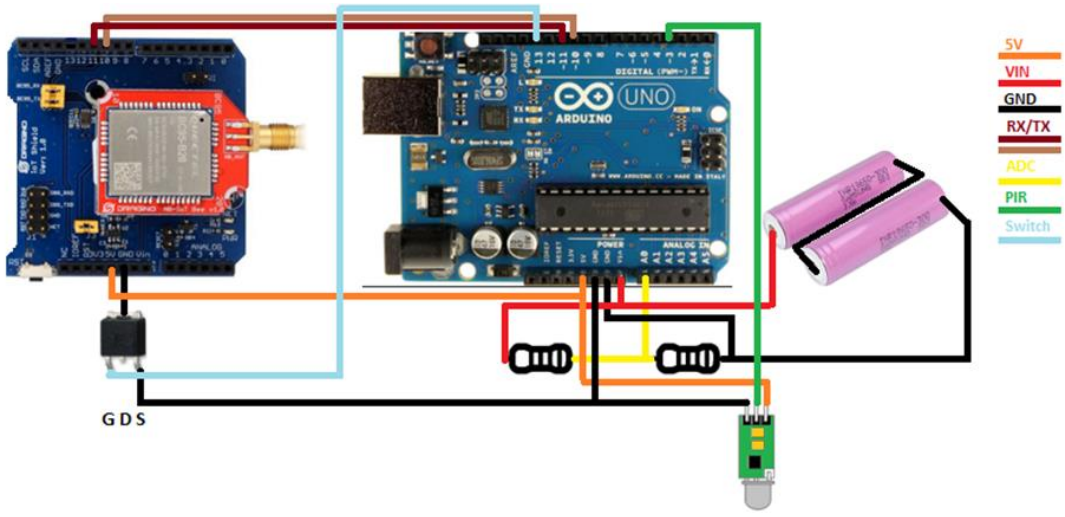

Figure 6. Second prototype of PIR based IoT device.

Second prototype has a bigger battery, new NB-IoT sensor shield and different mosfet, but the working principle is the same: humans are detected with PIR and the information is sent to the elevator server using NB-IoT technology. In case of NB-IoT sensor shield error, mosfet is used to reset the NB-IoT module Quectel BG96. In addition, 2 resistors are used to create a voltage divider to measure the battery level and replace and recharge the batteries in a timely manner.

This Dragino sensor shield has UART pins defined, so that Arduino and NB-IoT module can communicate through UART interface, however it still does not have all the required pins defined to test all power saving modes of the NB-IoT module [26].

4.1.1 Voltage divider

As this device is battery powered, the battery level should be monitored with analog-to-digital converter (ADC). Since the Arduino Uno ADC maximum input voltage is only 5V, voltage divider had to be made to lower the input voltage. For this, 2 resistors are used to create a voltage divider circuit as shown in Figure 7.

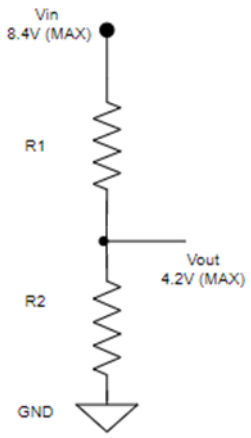

Figure 7. Voltage divider schematic for the second PIR based IoT device.

The formula for this voltage divider circuit is given in the equation below.

$$Vout = \frac{Vin * R2}{R1 + R2}$$

The Vin maximum value is 8.4V, as the two 3.6V (4.2V when fully charged) Li-Ion 18650 batteries [27] are connected in series to increase the input voltage for Arduino. To get the

Vout lower than 5V, it is best to use same value resistors, as then the Vout is reduced to half of the Vin value, changing the maximum Vout value from 8.4V to 4.2V.

4.1.2 Casing and placement

This PIR based IoT device is made compact and its casing (Figure 8) is 3D drawn to attach and place all the hardware components inside. Size of the custom casing is 146 x 96 x 68 mm and it is 3D printed in the TalTech office using Polylactic Acid (PLA).

Figure 8. 3D printed casing for the IoT device.

Since PIR (HC-SR505) detects everything that moves in 3 m distance from a 100 degree cone angle, it had to be limited to only detect elevator passengers that are in front of the elevator. For this, tape measurements were made at the entrance of the KONE elevator in the ICT office building in TalTech. Area, where to detect elevator passengers is shown in Figure 9, marked with green colour (on the ground) and is measured to be 230 cm x 180 cm. The device is placed above the elevator at a height of 2.3 m, marked with red cross.

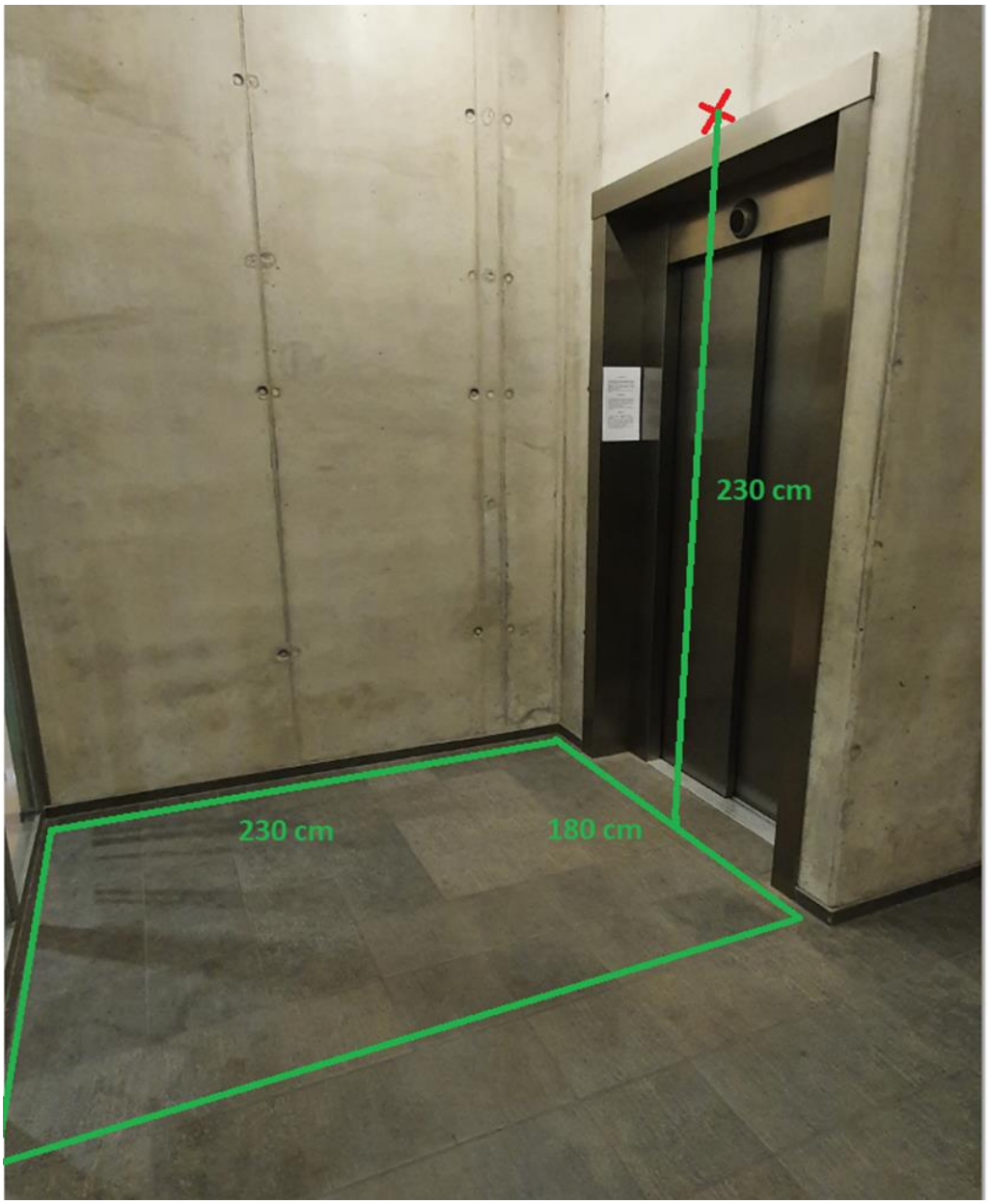

Figure 9. Area of Interest for the PIR sensor.

To detect elevator passengers and not everything that moves in the PIR detection range, the device had to be placed on top of the elevator entrance door in a certain angle with a restricted field of view. As the FOV of the PIR is 100 degrees cone angle, the optimal placement of the PIR inside the 3D printed box would be -5 to 50 degrees vertical face down. If the PIR is placed at a smaller angle, movement would not be detected at the far end of the specified area, and if the PIR is placed at a greater angle, movement in front of the elevator entrance would not be detected. The simplest and the most reliable placement of the PIR was selected: 0 degrees, PIR facing down.

Since the height of the elevator is 230 cm and the distance across from the elevator door is 230 cm, the angle α, which should limit the FOV from the front, can be calculated.

$$\alpha = \tan^{-1}\left(\frac{230}{230}\right) = 45°$$

FOV from the front should be limited at an angle of 45° and at the sides from 42.74° (Beta) to 30.93° (Gamma), these FOV restriction angles (Beta and Gamma) are calculated and more deeply explained in chapter 8.2.

4.1.3 Software for the PIR based IoT device

Software for the PIR based IoT device was written on Arduino Uno, which communicates with the Dragino NB-IoT sensor shield through the UART interface. This software for the PIR based IoT device can be found in the corresponding software repository (Appendix 1 – Software repository). Since NB-IoT module BG96 has a default baud rate 115200 for Quectel, it had to be changed to 57600, as the Arduino UNO does not support 115200 baud rate well. When using baud rate 115200, incorrect and unreadable characters were received.

On the first boot up, NB-IoT configuration had to be done with AT commands [1], which consists of:

- Configuring Radio Access Technology (RAT) searching sequence
- Network category to be searched under LTE RAT
- Band configuration
- Enabling network registration
- Configuring parameters of TCP/IP context
- Activating packet data protocol (PDP) context
- Configuring Domain Name System (DNS) server
- Syncing local time with Network Time Protocol (NTP) server

[1] NB-IoT module BG96 configuration documentation is available on https://gitlab.pld.ttu.ee/inguit1/Indrek_Guitor_Masters_Thesis/tree/master/BG96_AT_COMMANDS/Software

Depending on the power saving mode used (more info in chapter 7.1) Arduino is woken up by the interrupt caused by PIR, which then itself wakes up NB-IoT. After which the next steps are taken:

- Battery level is checked
- PDP context is activated
- TCP socket service is opened
- Current time is acquired from NB-IoT module
- Message is encrypted with Advanced Encryption Standard (AES) -128
- Message is sent to the elevator server

Message sent to the elevator server (Table 3) consists of:

- ID - Device ID (depending on the floor)
- hh:mm:ss - Current time
- xx - Battery level (01-99)

Table 3. Message sent to the elevator server.

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Data	I	D	,	h	h	:	m	m	:	s	s	,	x	x

After these steps, both Arduino and NB-IoT are put back to selected power saving modes and the cycle repeats.

4.2 Thermal camera based solution

To better analyse the human movement: to distinguish elevator passengers from passers-by, thermal camera is used. Additional visualizing device had been made in order to analyse and visualize the thermal camera output and compare it with the real scenarios. This solution is the advancement of the PIR based solution, however it is only a solution, not a fully functioning IoT device, as more focus was used on analysing and determining the elevator passengers and passers-by.

Thermal camera based solution, shown in Figure 10, is a simple device, that consists of only 2 components:

- OMRON D6T-32L-01A Thermal camera
- STM32L432KCU6 microcontroller

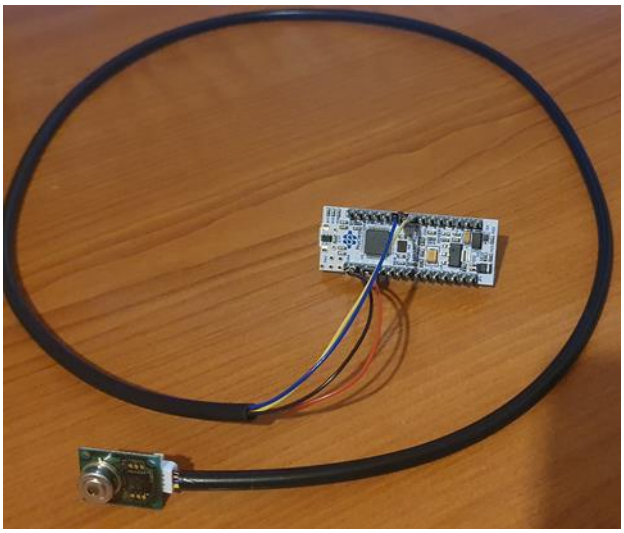

Figure 10. Thermal camera based solution.

MEMS Thermal camera (D6T-32L-01A) high sensitivity enables detection of moving or stationary human presence. It features high Signal-to-noise ratio (SNR), which is achieved by OMRON's unique MEMS and application-specific integrated circuit (ASIC) technology, superior noise immunity with a digital output and high-precision area temperature detection with low crosstalk field of view characteristics. It has incredibly high FOV: 90 degrees horizontal and 90 degrees vertical with object detection ranging from 0 to 200 degrees Celsius. The sensor uses Inter-Integrated Circuit (I2C) communication protocol and it updates the temperature data every 300 ms. However, the power consumption of this sensor compared to PIR is higher, at around 19 mA, as it has 1024 temperature elements [28]. Moreover, the price of this sensor is very high, especially in comparison with PIR sensor.

4.2.1 Thermal camera based solution software

This thermal camera based solution software can be found in the corresponding software repository (Appendix 1 – Software repository). Since this D6T thermal camera outputs large amounts of data through the I2C interface, the STM32 clock speed had to be

modified to be 1000 kHz (Fast-Mode Plus). The D6T sensor updates temperature data every 300 ms and the data outputted is in the following format (Figure 11) [28]:

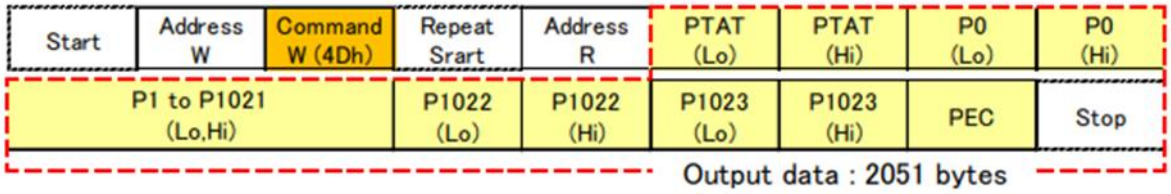

Figure 11. D6T-32L-01A output data format.

MEMS Thermal Sensor D6T examples were used to read and store these values [29]. The PTAT and Pn temperature data represents values equal to temperature values (°C) multiplied by a factor of 10 as signed 16-bit integers. Packet error code (PEC) value, which represents the cyclic redundancy check (CRC-8), is used to detect communication errors and improve the data reliability. After verifying the correctness of the data, it had to be unpacked and put into another more readable format (Table 4) for the visualizing device (chapter 4.2.2), so that the values could be forwarded to it through the UART interface. The new data packet consists of 1024 temperature measurements with 1 decimal point, separated by commas. The packet ends with a newline to be distinguishable from other packets and an example of the packet is shown in the table below.

Table 4. Temperature data format for the visualizing device.

Byte	0	1	2	3	4	5	6	7	8	9	10-5109
Data	2	2	.	5	,	2	2	.	6	,	...
Byte	5110	5111	5112	5113	5114	5115	5116	5117	5118	5119	5120
Data	,	2	3	.	2	,	2	4	.	6	\n

4.2.2 Visualizing device for thermal camera

The visualizing device, that is shown in Figure 12, is made to collect and analyse the thermal camera output data and compare it with RGB camera output. The software for this device can be found in the corresponding software repository (Appendix 1 – Software repository). It consists of the following hardware:

- Raspberry Pi 3 model B+
- Raspberry Pi camera V2.1
- Thermal camera D6T-32L-01A
- STM32L432KCU6
- XD Xclusive 5000mAh power bank

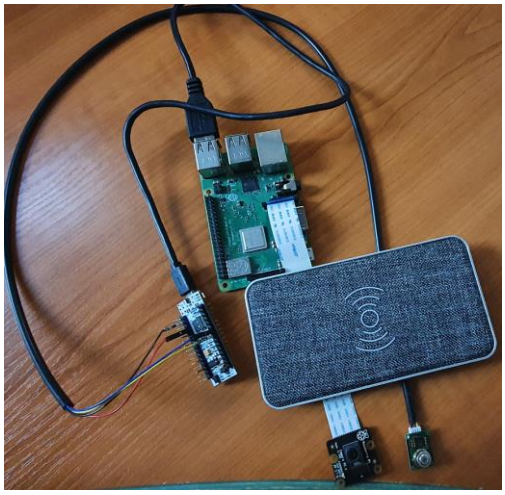

Figure 12. The visualizing device.

Due to worldwide pandemic, this hardware is placed inside an available box found in the dorm and then placed approximately at the same height (2.3 m) as the original IoT device should be in the ICT office building (Figure 13, marked with red box). The different elevator passenger behaviour scenarios are also played out in the dorm because of this pandemic.

Figure 13. Visualizing device placed in the dorm.

The Raspberry, which is the main processing unit for this device, receives temperature data from STM32 through the UART interface. Since the UART is not the most reliable communication interface, the packet length and last byte are checked. These checks are used, as most bytes are lost, not corrupted, especially if a large amount of data is transferred every 300 ms. This data loss in packets also appeared while testing elevator passenger behaviour scenarios, which is described in more details in chapter 8.1.

However, if the thermal camera output data is correct, the Raspberry immediately takes a snapshot with the Raspberry Pi camera and saves the image on the SD card's specific folder along with the temperature data received. This folder can then be later opened with visualizing and analyzing software, to compare and analyse the thermal camera output and Raspberry Pi camera output side by side.

4.2.3 Visualizing and analyzing software

Visualizing and analyzing software, which can be found in the corresponding software repository (Appendix 1 – Software repository), is a python program made to display thermal camera output and Raspberry Pi camera output side by side. Example image of the software is shown in the Figure 14. It allows the user to open the specified folder,

where the thermal camera and RGB camera outputs are saved. Analysis of different human behaviour scenarios in front of the elevator is described in chapter 8.2.

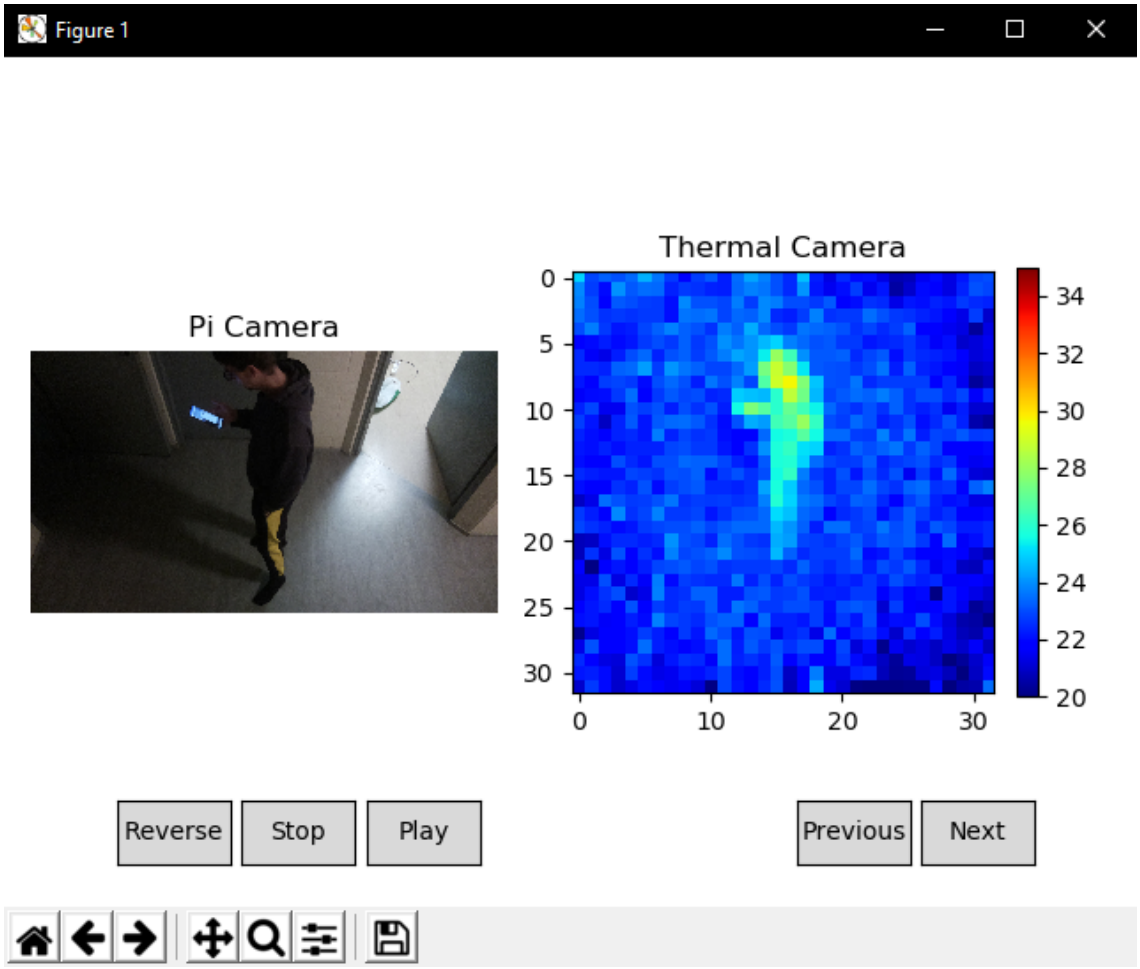

Figure 14. Example image of the visualizing and analyzing software.

As the thermal camera outputs the 32 x 32 matrix temperature data in a string, this program displays the values as a matrix, just as the image was taken, where the temperature values are also visualized with cool and warm colours. The following Figure 15 shows how the thermal camera stores each pixel value and the visualizing and analyzing software displays the pixels the same way they were captured by the thermal camera.

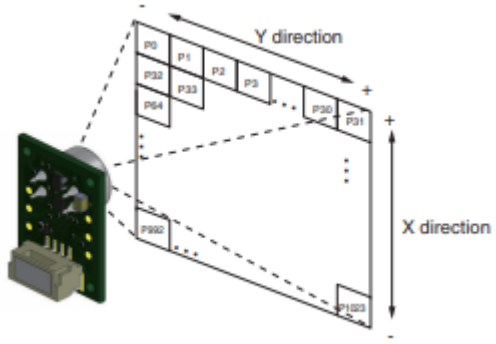

Figure 15. Thermal camera D6T detection area for each pixel.

The RGB image is next to the thermal camera matrix visualization and the user can view these outputs side by side. User also has an option to play these outputs as a video, forward or backward, with a possibility to stop it or even scroll it forward and backward one by one. The only problem with this program is that even though the RGB camera and thermal camera are attached side by side on the visualizing device, the field of view angles are not the same, as thermal camera FOV is 90 degrees horizontal and 90 degrees vertical, RGB camera FOV is 62.2 degrees horizontal and 48.8 degrees vertical. This causes the edges of the images not to match perfectly.

5 Security measures used for the application server

Security is a very important aspect when it comes to advancing this elevator system as hackers could potentially crash this publicly used elevator. This is why many different security measures were used. Firstly, as NB-IoT is using cellular networks, it benefits from all the security and privacy features of mobile networks, such as support for user identity confidentiality, entity authentication, confidentiality, data integrity, and mobile equipment identification. This makes NB-IoT more secure than other wireless IoT communication technologies [30].

To make the message sent from the IoT device to the elevator system more secure, the message is encrypted with AES with a key size of 128 bits. Available AES libraries for Arduino were used, message was divided into blocks of 16 bytes, each block of data was encrypted separately, and the data blocks were put together and sent out as a one single message. This makes the message accessible only to the authorized parties, such as the elevator server. However, it does not prevent interference to the elevator server, but it denies the intelligible content to a would-be interceptor.

To prevent the interference, TalTech network opened the network communication to the outside world for only specific port, with a specific IP range. The IP NB-IoT device could get, is specific and based on the APN used in the network. In this master's thesis, the cellular network provider is Telia, who provided the APN and its specific IP range based on it, as they are responsible for assigning IP addresses for the end devices based on the APN-s they use.

Furthermore, the software written by the author for the elevator server that is receiving UDP and TCP packets sent from IoT devices, is always checking for the incoming message port. This software also has other security measures, such as checking that the encrypted message contains only hexadecimal characters and has specific length, it also checks the ID inside the message, which determines on which floor the elevator is needed. This software can be found in the corresponding software repository (Appendix 1 – Software repository).

6 NB-IoT power saving modes

NB-IoT module BG96 has a variety of power saving modes available to use. To know which power saving mode is the most effective for this use case, each one of them was reviewed and tested. Testing different power saving modes includes measuring current consumption and time from detecting a human movement to actually sending the data to the server. It should be noted that at that time TalTech had not granted network access to the IoT device through the specified port, the power saving modes were tested using UDP connection to the private network server. Brief description of the available power saving modes is given in the Table 5.

Table 5. Different NB-IoT power saving modes.

NB-IoT power saving mode	**Description**	**Disabled functionalities**
Powered down	Least amount of current consumed, high wake up time	All functionalities disabled
PSM	Similar to “powered down” mode, very low current consumption, lower wake up time, as the module remains registered to the network	All functionalities disabled, but module remains registered to the network
Sleep	Current consumption reduced to a lower level, device can still receive data	Unknown
Airplane	Medium current consumption, low wake up time	Radio frequency functions disabled
Minimum functionality	Similar to airplane mode, medium current consumption, low wake up time	Radio frequency and (U)SIM functions disabled
Idle	Highest amount of current consumed, no wake up time, device can send and receive data at all times	-
eDRX	Similar to idle, high power consumption, no wake up time	Receiving disabled for a period of time

While the device is mostly in power saving mode, it goes through many other stages before actually sending the data and going back to power saving mode. This includes waking up from power saving mode (interrupt caused by the PIR), activating PDP

context, opening UDP socket, processing data (getting hardware clock time, encrypting the message), sending UDP packet and going back to power saving mode for at least 8 seconds (as this is the time PIR's output is high after detecting movement, which cannot be changed). For this reason, oscilloscope was used to measure and analyse the current consumption of each power saving mode stage.

6.1 Test setup

To get an accurate measurement of current consumption, the current must be measured with respect to time. Therefore, oscilloscope is required, as a basic multimeter is not sufficient. Since the oscilloscope available to use is with voltage probes, and not with current probes, a small resistor is used in line with the power supply input to the system. Standard oscilloscope voltage probe is used to measure the voltage across the resistor, which is used to effectively measure the current by dividing the voltage by the resistance. Resistor with a small value of 1.5 ohm is used, as this value is small enough that it shouldn't affect the existing circuitry, and large enough to provide a voltage that can be measured with decent precision [31].

When performing measurements, regulated direct current power supply was used instead of batteries, as batteries can be defective and run out, which can affect the measurement results. Figure 16 shows the test setup used for current measurement.

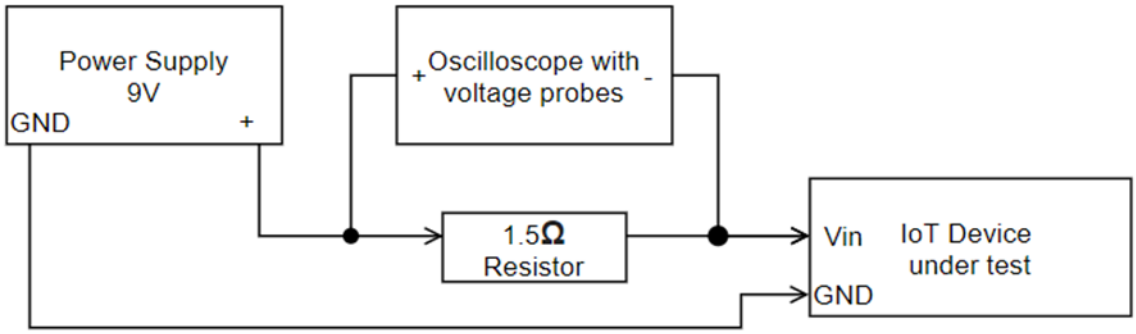

Figure 16. Current measurement setup for the IoT device.

Components used to measure current consumption:

- Power Supply: Manson NSP-2050 Switching mode power supply
- Oscilloscope with voltage probes: Agilient Technologies InfiniiVision DSO-X 3034A
- 1.5-ohm Resistor: FM0207 1.5 Ohm resistance, 0.6W power, 1% tolerance
- IoT Device under test (second prototype of PIR based IoT device)

6.2 Embedded software

The NB-IoT power saving modes are controlled by Arduino. Besides reducing power consumption on NB-IoT module, Arduino should also be using low power mode. Two different low power mode libraries for Arduino were tested: LowPower.h [1] and avr/sleep.h with avr/power.h [2], and the results were similar - current was reduced to about 20mA.

It should also be noted, that the Arduino sleeping time is not consistent in the test cases. The main reason for this delay time fluctuation is related to the communication between the NB-IoT and Arduino. Arduino gets "OK" answer from NB-IoT and starts its delay but the NB-IoT may not have finished its task and is continuing to consume power. Also, the open source software library LowPower.h, which is used on Arduino, might use the timers incorrectly, which is why the 8 second delay in Arduino almost always goes over 8 seconds and 2 second delay mostly ends before actual 2 seconds. This anomaly can be seen in the 6.3 sub-chapters.

The software cycles start with an 8 second sleep, as this is the time PIR's output is high after detecting movement, which cannot be changed. Therefore, the Arduino is sleeping for the same amount of time as the PIR is high. Ideally the cycle from waking up to sending the data to going back to power saving mode would be without delays: to make it faster and consume less power, but it is not possible in this case (more details in chapter 6.2.1). The problems occur, if the delay between each AT command is removed or too

[1] Library is available in https://github.com/rocketscream/Low-Power

[2] Documentation for these power saving modes is available in https://www.nongnu.org/avr-libc/user-manual/modules.html

short, as the NB-IoT module would produce more errors, causing the Arduino to retry certain AT commands, which takes additional time and energy.

6.2.1 Optimal delay

To find out how long the optimal delay should be between AT commands, it was best to ask from TalTech PhD students in Thomas Johann Seebeck Department of Electronics, who have also been working with NB-IoT. They suggested to use 4 to 6 second delay between each AT command. Since this huge delay between each AT command is not acceptable, tests were performed to find the optimal delay:

- That should be used after the NB-IoT device wakes up from power saving mode and before the first AT command
- Between each AT command

To perform tests, Arduino software timer was used to measure the time between waking up from power saving mode to sending the data. Each one of these tests include following stages: waking up from power saving mode, activating PDP context, opening UDP socket, data processing, sending data using UDP socket, along with the delay after waking up from power saving mode and delay between each AT command. Since optimal delay between each AT command does not depend on the power saving mode used, minimum functionality power saving mode was used. Each delay combination was selected based on the previous knowledge. Each combination was tested 10 times to find the average test time with the given delay combination, which is then used to find the best delay combination: delay between each AT command and delay after waking up from the power saving mode. Table 6 [1] represents the given test results of using different delay times.

[1] First two rows of the table represent the combination of delays.

Table 6. Optimal delay test results for AT commands.

Delay after waking up from power saving mode (sec)	3	4	2	0
Delay between each AT command (sec)	1	1.5	2	2
Test 1 (ms)	12577	8963	10495	10784
Test 2 (ms)	90432	136754	8463	6804
Test 3 (ms)	6844	29275	10495	11465
Test 4 (ms)	8377	8962	11097	6663
Test 5 (ms)	7625	101631	8463	6613
Test 6 (ms)	7597	8962	10496	7023
Test 7 (ms)	7044	31152	11437	6594
Test 8 (ms)	8643	8963	8463	6713
Test 9 (ms)	8007	10005	10495	6603
Test 10 (ms)	86619	8962	11068	6744
Average time in ms	24376.5	35362.9	10097.2	7600.6

As it can be seen from the Table 6, increasing the delay after the module has woken up only increases the time for the data to be sent out. It can also be seen that decreasing the delay between each AT command to under 2 seconds, led to more errors being produced, which in turn led to the average test time to be higher. For example, the time for the module to complete the test went over a minute for 4 times over 20 test cases. According to data from the table, at around 2 second delay between each AT command, everything worked out smoothly: almost no errors were produced and the average time to complete the test case were similar. This leads to a conclusion, that increasing the delay between each AT command to over 2 seconds would only increase the overall time to complete the task, which would not be practical.

Around the same time, meeting with Quectel (the NB-IoT module manufacturer) was taken place, where the author had an opportunity to discuss NB-IoT module problems: huge delay between AT commands and PSM not working (chapter 6.3.2.1). It was mentioned that the NB-IoT module should work correctly (without producing errors and without crashing) with about 500 ms delay after each AT command. This, however, turned out to be working only for certain AT commands, such as general commands (requesting the firmware version, current configuration, manufacturer identification etc.), not for network related AT commands.

6.3 Measurement and analysis

The best results are obtained using multiple work cycles of NB-IoT power saving modes, which consist of waking up from NB-IoT power saving mode to sending data to elevator server and going back to power saving mode. These results are captured from oscilloscope as Portable Network Graphics (PNG) and Comma Separated Values (CSV) files to analyse them.

Since this oscilloscope outputs 2000 measuring points for the CSV file, current consumption calculations are not 100% accurate, as the length of one cycle (waking up from power saving mode, activating PDP context, opening UDP socket, sending data to elevator server, going back to power saving mode etc.) varies between different power saving modes, meaning the time differs between two measuring points. For example, in power down mode, one measuring point is 0.0145 seconds, however in minimum functionality mode it is 0.0110 seconds, although both power saving modes have 2000 measuring points all together.

6.3.1 Powered down

Least amount of power is consumed if the device is not powered on when there is no data to be transmitted. On the other hand, turning back on the NB-IoT module takes a lot of time and energy as the module needs to register to the network each time it is powered on. The power consumed in relation to time for this mode is shown in the Figure 17 and each state in the mentioned graph is described in Table 7 below.

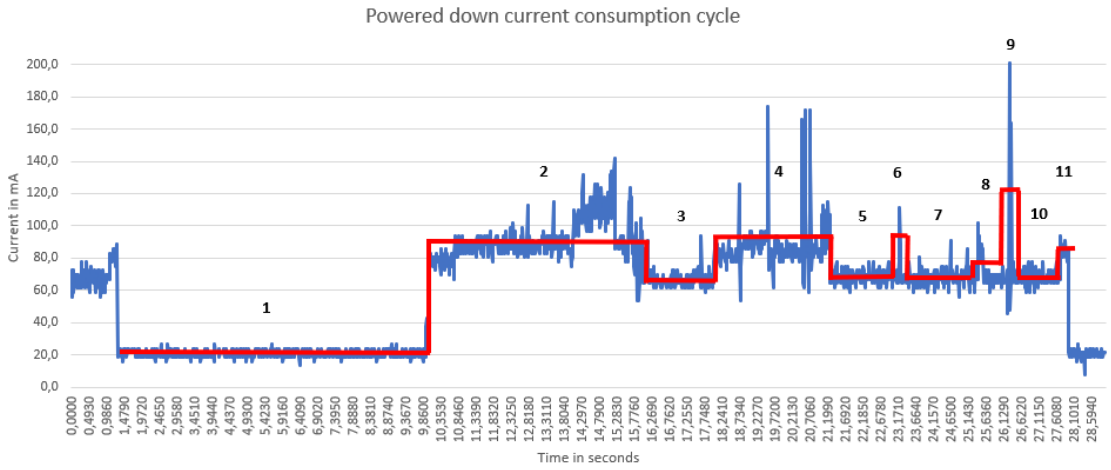

Figure 17. Powered down current consumption cycle.

Table 7. Description of the powered down current consumption cycle stages.

Stage in graph	Stage description	Time (s)	Current (mA)
1	Arduino sleeping and NB-IOT powered down	8.6275	20.9324
2	NB-IoT Waking up	6.2060	90.1914
3	Arduino sleeping	1.8415	66.5831
4	Activating PDP context	3.2335	87.2936
5	Arduino sleeping	1.9140	68.2512
6	Opening UDP socket	0.0435	91.5157
7	Arduino sleeping	2.1460	67.9732
8	Data processing	0.8700	70.0773
9	Sending data using UDP socket	0.0435	119.2630
10	Arduino sleeping	1.3630	68.1585
11	Powering down NB-IoT and Arduino sleeping	0.2030	84.5114

As it can be seen from the Figure 17, the NB-IoT waking time is over 6 seconds, after what Arduino can start communicating with this module. The time from detecting a human to sending the data to elevator server takes the longest in this powered down mode compared to other power saving modes. It is also important to notice that the first PDP

context activation always fails in the first tries, which is why it is in this stage for longer period of time compared to PDP context activation stages in other power saving modes. To fix this issue, delay after waking up from powered down mode needs to be increased, as this issue does not exist in other power saving modes. After the PDP activation, other stages do not differ a lot in other power saving modes, but they do take time. For instance, data processing takes about 0,8 seconds, which consists of acquiring current hardware time from Quectel BG96, forming a data packet and then encrypting it.

6.3.2 Power Saving Mode (PSM)

Best way to reduce power consumption of the BG96 module is to put it in Power Saving Mode (PSM) after each data transmission, as it is similar to powered down mode. According to 3GPP Release 13, the device can be in PSM mode for approximately up to about 413 days. The advantage of this mode over powered down, is that the module remains registered on the network and there is no need to reattach or re-establish packet data network (PDN) connections, which saves time and energy [32].

This mode is only possible to use, if the network operator supports PSM. In this thesis, the network operator was Telia, who confirmed that the PSM was supported in the network.

This module requests two timers on every attach for this mode:

- Active Timer T3324
- Tracking Area Update (TAU) T3412

Active timer is used to keep the device powered on, for example, receive, process and send data, and when the Active timer expires, the device goes to Power Saving mode. TAU is used to periodically notify the availability of the IoT device to the network.

PSM time is the difference between these two timers (T3412 - T3324) as shown in the Figure 18. The network may accept these values or set different ones. The network then retains state information and the device remains registered with the network. If a device awakes and sends data before the expiration of the time interval it agreed with the network, a reattach procedure is not required [33], [34].

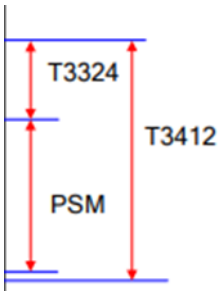

Figure 18. The interdependence of timers T3324, T3412 and PSM.

6.3.2.1 PSM problems

Unfortunately, this mode did not work as planned and it caused a lot of problems. Firstly, every AT command (test and reading) that was related to PSM, gave error. After further investigation it turned out that the firmware was too old for this mode, so it was required to contact Quectel and request a new firmware, as the firmwares are not online and they are not available for everybody. Contacting them via e-mail, new firmware access was granted (specifically for Europe), along with the tools and documentations to be used for the upgrade.

After this, PSM test and read commands gave correct answers, but the module still did not work as planned. Developers from Telia and PhD students from TalTech had same problems: the module is not going into PSM as planned. A few configurations worked, but only once, as when trying to use the same configuration again, everything failed.

Using PSM AT command to configure these two timers, the NB-IoT module responded with “OK”, as if everything is correct and there are no errors. However, when checking timer values that were saved, NB-IoT returned incorrect values, as it can be seen from the Figure 19. Different PSM test cases are displayed in the figure, in the test case A, the first “AT+CPMS=1” command [1] is used to enable PSM with T3412 value “01101010”, which corresponds to 20 seconds, however, when reading the stored value with the “AT+CPMS?” command, the T3412 value is changed into “01001000”, which corresponds to 80 hours. In the test case B, both of the stored T3324 and T3412 timer values are read with “AT+CPMS?” command with 2 seconds apart, yet the stored values

[1] Exact description of the mentioned AT commands is described in the document: https://gitlab.pld.ttu.ee/inguit1/Indrek_Guitor_Masters_Thesis/blob/master/BG96_AT_COMMANDS/Software/Quectel_BG96_AT_Commands_Manual_V2.3.pdf

are not the same. Test case C is similar to test case A, where the written and stored timer T3324 and T3412 values are not the same.

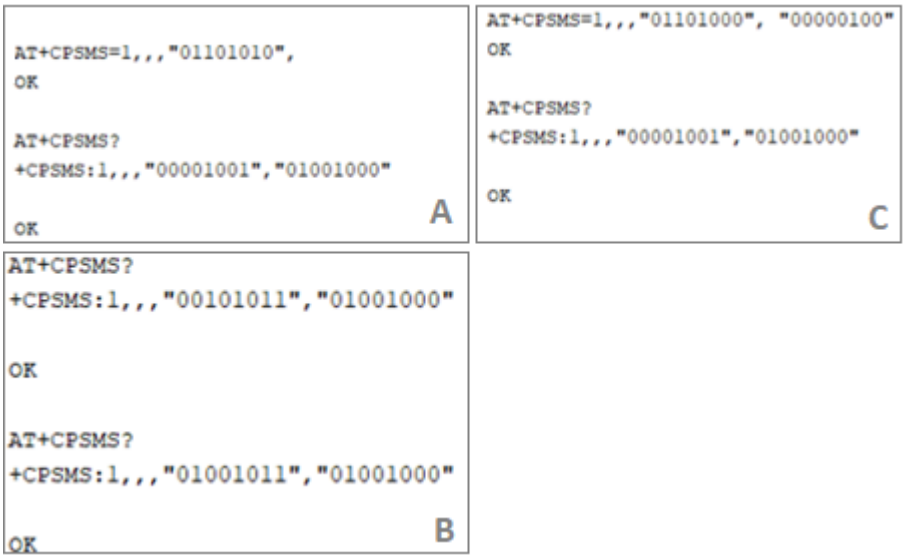

Figure 19. Examples (A, B, C) of PSM not saving the given timer values correctly.

However, there are options for this module to go to PSM immediately, with certain AT commands, for instance:

- AT+QCFG="psm/enter",1
- AT+CPSMS=1,,,10000001,00000000

The first AT command is usually used to trigger the module enter into PSM mode immediately after the Radio Resource Control (RRC) is released, but this did not work, even after waiting for hours.

On the other hand, second AT command controls the settings of the power saving mode (PMS) parameters, which in this case should enable entering into PSM. Setting TAU value (T3412) to 30 seconds and Active Time value (T3324) to 0. This worked partially with trial and error method, as this setting is used for the module to go into PSM immediately. The module should stay in PSM mode forever, regardless of the TAU timer. Unfortunately, this method has its own problems, for instance, when waking up the module with interrupt, it almost immediately goes back to PSM, with very limited time to send AT commands for this module to stop entering PSM or to change active timer values to complete its task: send UDP packets to elevator server. Sometimes these AT commands are not even executed, as the module goes into PSM too fast, meaning it is

unreliable to use this method (setting the Active Timer value to 0) for the module to go into PSM mode.

6.3.3 Sleep

To reduce power, one option is to put the NB-IoT module to sleep. This can only be done, if the host microcontroller communicates with the NB-IoT module via UART interface and the connection between the host and the module is following (Figure 20):

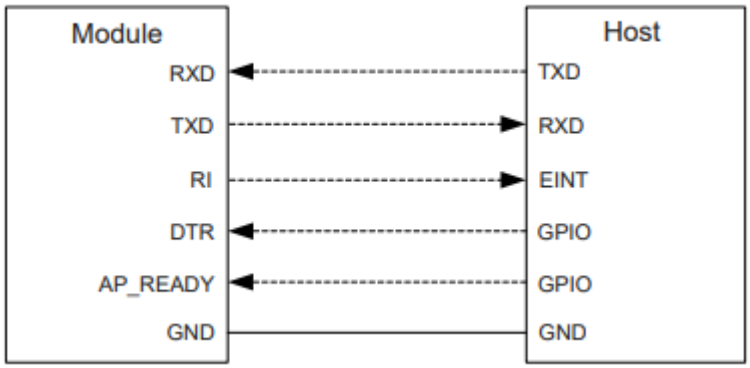

Figure 20. Required connection between Host and Module for the sleep mode.

Firstly, the sleep mode has to be enabled with the following AT command:

- AT+QSCLK=1.

After executing this command, the DTR pin can be driven high for the module to enter sleep mode. The BG96 can be woken up from sleep by driving the DTR pin to low level. AP_READY pin can be used to detect the sleep state of the host and RI pin is used to wake up the host, if the BG96 has Unsolicited Result Code (URC) to report. For more details and behaviours about these pins please refer to the Quectel documentations [1].

In this mode, the current consumption of the module will be reduced to a lower level, but the module can still receive paging messages, SMS and TCP/UDP data from the network normally. Unfortunately, the DTR, RI and AP_READY pins were not defined and the

[1] https://gitlab.pld.ttu.ee/inguit1/Indrek_Guitor_Masters_Thesis/tree/master/BG96_AT_COMMANDS/Software

documentation is nowhere to be found for Avnet silica nor Dragino expansion shield for this mode to be tested.

6.3.4 Airplane mode

Airplane mode is used to reduce power consumption by disabling Radio frequency (RF) functions. When the module enters airplane mode, the RF functions do not work and all AT commands correlative with RF function will be inaccessible.

This mode can be set via the following ways.

- Hardware - W_DISABLE pin is pulled up by default. Driving it to low level will let the module enter airplane mode.
- Software - Following AT commands are used to put the device into airplane mode back to full functionality mode:
    - AT+CFUN=4
    - AT+CFUN=1

As mentioned before, the hardware pins on the NB-IoT expansion shields are not documented, meaning this option cannot be used to put the device into airplane mode. On the other hand, using AT command to put the NB-IoT module to airplane mode worked perfectly. The power consumed in relation to time for this mode is shown in the Figure 21 and each state in the mentioned graph is described in Table 8.

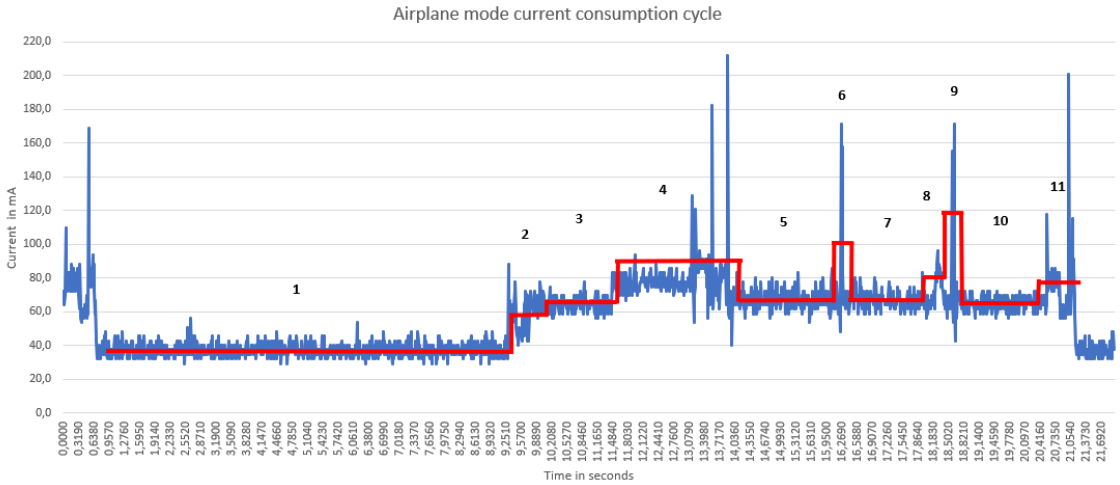

Figure 21. Airplane mode current consumption cycle.

Table 8. Description of the airplane mode current consumption cycle stages.

Stage in graph	Stage	Time (s)	Current (mA)
1	Arduino sleeping and NB-IoT in airplane mode	8.6130	37.3500
2	Activating NB-IoT full functionality	0.4290	57.3535
3	Arduino sleeping	1.7160	64.7997
4	Activating PDP context	2.5080	79.2551
5	Arduino sleeping	2.2330	67.6193
6	Opening UDP socket	0.0440	103.4506
7	Arduino sleeping	1.6610	67.1958
8	Data processing	0.6050	70.1126
9	Sending data using UDP socket	0.0440	103.4507
10	Arduino sleeping	1.9140	65.1028
11	Activating NB-IoT airplane mode and Arduino sleeping	0.5940	76.2114

As it can be seen from this mode, activating PDP context requires much less time compared to power down mode: about 0,8 seconds less. Other important stages look similar to other modes both in time and current consumption wise.

6.3.5 Minimum functionality

Minimum functionality mode is used to reduce power consumption by disabling RF functions along with (U)SIM card. When the module enters minimum functionality mode, the RF and (U)SIM functions do not work and all AT commands correlative with RF and (U)SIM function will be inaccessible.

Following AT commands are used to put the device into minimum functionality mode and back to full functionality mode:

- AT+CFUN=0

- AT+CFUN=1

This mode should consume slightly less power compared to airplane mode, because in addition to disabling RF functions, (U)SIM is also disabled. On the other hand, it should take a little bit more time to wake up from minimum functionality mode, as the (U)SIM functions should power up first, compared to airplane mode, where (U)SIM functions are already enabled. The power consumed in relation to time for this mode is shown in the Figure 22 and each state in the mentioned graph is described in Table 9 below.

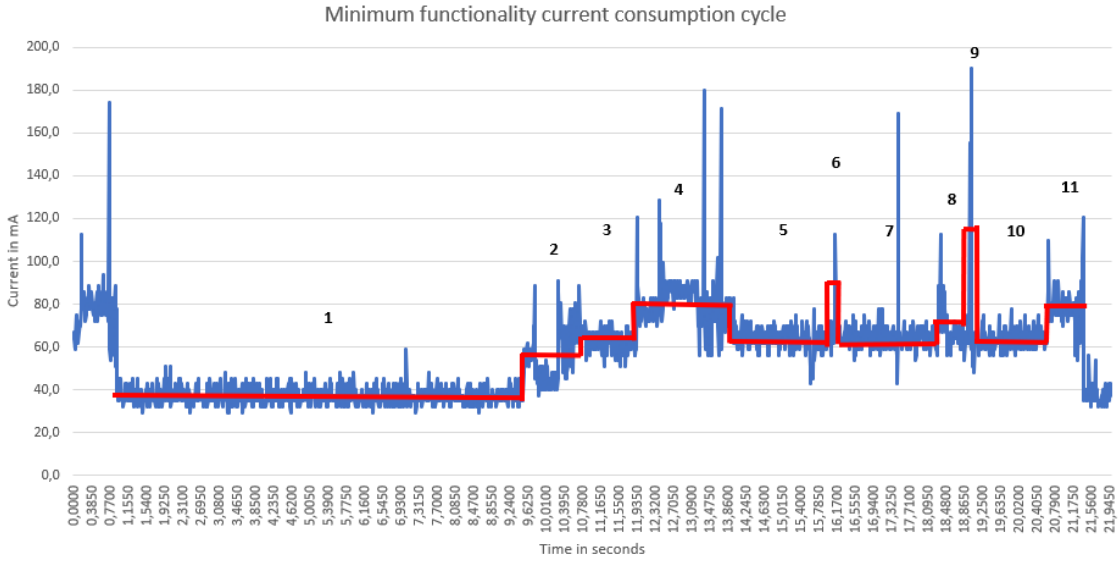

Figure 22. Minimum functionality mode current consumption cycle.

Table 9. Description of the minimum functionality mode current consumption cycle stages.

Stage in graph	Stage	Time (seconds)	Current (mA)
1	Arduino sleeping and NB-IoT in minimum functionality	8.6130	37.2167
2	Activating NB-IoT full functionality	1.1770	55.8844
3	Arduino sleeping	1.1880	62.8464
4	Activating PDP context	2.2440	80.3757
5	Arduino sleeping	1.9360	63.9583
6	Opening UDP socket	0.0330	89.0870
7	Arduino sleeping	1.9800	65.2694
8	Data processing	0.8140	67.1089

Stage in graph	Stage	Time (seconds)	Current (mA)
9	Sending data using UDP socket	0.0330	117.2529
10	Arduino sleeping	1.5840	63.5084
11	Activating NB-IoT minimum functionality and Arduino sleeping	0.7590	78.2197

It can be seen that the minimum functionality mode indeed takes more time to wake up - more than 2.5 times longer, compared to airplane mode. However, the energy consumption in the minimum functionality mode is only about 0.13 mA lower than in the airplane mode.

Also, it should be noted that sometimes when the NB-IoT module is active and PDP context is activated, random energy consumption spikes can appear. This does not happen in every cycle, but it can happen, and it will consume additional power, as it can be seen in stage 7.

6.3.6 Idle

For the system to have the lowest delay, the NB-IoT should be in idle mode. In this mode, the PDP context is always activated and UDP socket is opened, which means the device can send data immediately and it does not have to waste time on AT commands and delays to set up the connection with the elevator server. On the other hand, keeping the system idle for the most of the time would lose the meaning of low power IoT, as it requires a lot of power.

As it was mentioned before, when PDP context is activated and UDP socket opened, random energy consumption spikes can appear - this is clearly visible in this mode, as it can be seen in the red circles in the Figure 23.

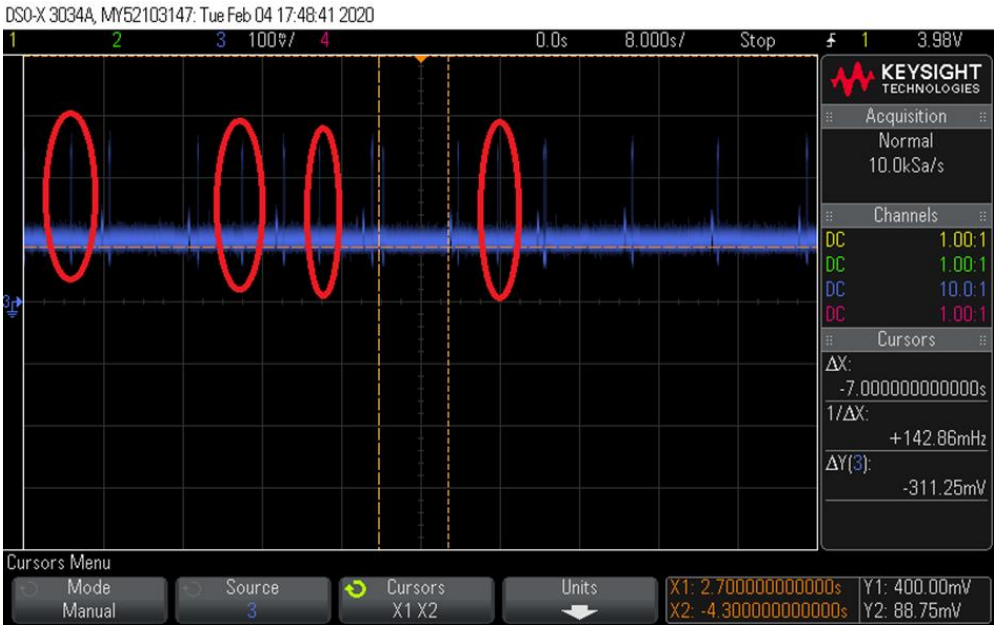

Figure 23. Energy consumption spikes in Idle mode.

The power consumed in relation to time for this mode is shown in the Figure 24 and each state in the mentioned graph is described in Table 10.

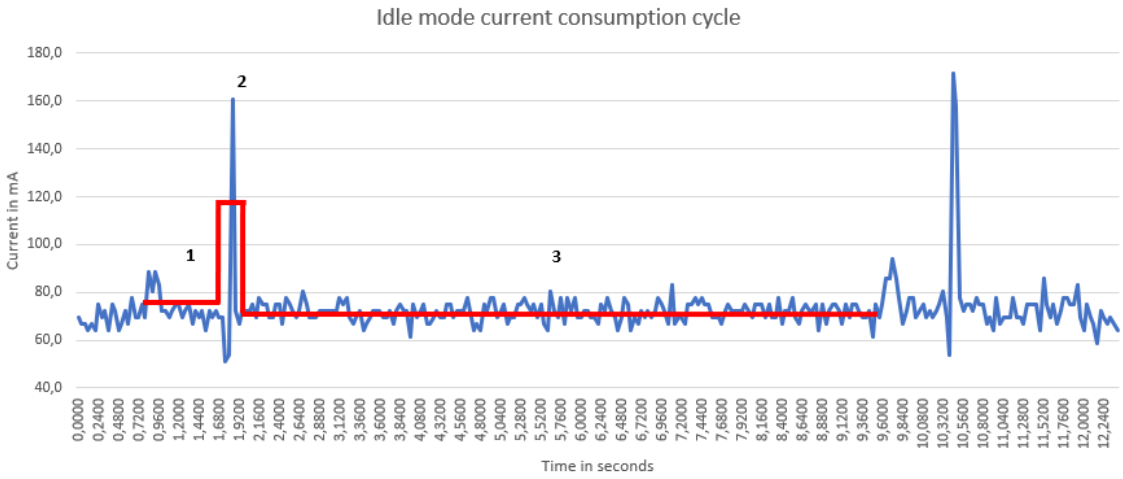

Figure 24. Idle mode current consumption cycle.

Table 10. Description of the idle mode current consumption cycle stages.

Stage in graph	Stage	Time (S)	Current (mA)
1	Processing data	0.8533	70.4851
2	Sending data	0.0600	116.6573
3	Arduino sleeping and NB-IoT in Idle	7.7440	68.8121

Since in this Idle mode the IoT device is always connected to the elevator server, the data is sent out after 2 stages, not after 8 stages, as it is described in the previous power saving modes. This makes the overall delay for this mode to be the lowest, however, the sleeping current consumption is the highest.

6.3.7 Extended Idle Mode DRX (e-I-DRX)

Extended Discontinuous Reception (eDRX) is an extension of an existing LTE feature which can be used by IoT devices to reduce power consumption. eDRX can be used without PSM or in conjunction with PSM to obtain additional power savings [33]. In the eDRX mode, the device can be in an inactive mode from only a few minutes to a few hours at a time.

The principle of e-I-DRX is to switch off the receive section of the radio module for a period of time. The IoT device cannot be contacted by the network while it is not listening, and this means the IoT device might receive data a few seconds later. This e-I-DRX decreases the device reachability, but it also decreases overall power consumption of the device.

The average power consumption in this mode, counting only the consistent cycles, is shown in Table 11, but it should be noted that using this mode is not recommended, as it is not working correctly (more details in chapter 6.3.7.1).

Table 11. Description of the eDRX mode current consumption cycle.

Stage	**Time (seconds)**	**Current (mA)**
eDRX (HIGH)	10.6740	81.1338
eDRX (LOW)	10.3005	55.2465

6.3.7.1 e-I-DRX problems

To test this method, AT command AT+CEDRXS was used to enable the use of e-I-DRX with the LTE Cat NB1 access technology. Firstly, e-I-DRX cycle length duration of 5.12 seconds (minimum length supported by BG96 module) was selected and no errors were present, however it did not work as supposed to. While researching and testing, it came out that the BG96 module supports 5.12 and 10.24 seconds e-I-DRX cycle length duration values, but in reality these values are not allowed according to GSMA whitepapers [33],

which is why reading eDRX dynamic parameters and measuring the current consumption of this mode with these values did not give promised results. It can be seen in the Figure 25 that when reading e-I-DRX settings with corresponding AT command, the response was “OK” with correct access technology (5 = LTE Cat NB1) and correct e-I-DRX cycle length duration (“0000” = 5.12 seconds), however, when reading e-I-DRX dynamic parameters, the response was also “OK”, but with “0” answer, meaning no eDRX was selected.

```
AT+CEDRXS? +CEDRXS: 5,"0000"OK

AT+CEDRXRDP+CEDRXRDP: 0OK
```

Figure 25. eDRX settings and dynamic parameters not matching.

Further testing with longer e-I-DRX cycle length durations somewhat worked and the graph from oscilloscope is displayed in the Figure 26.

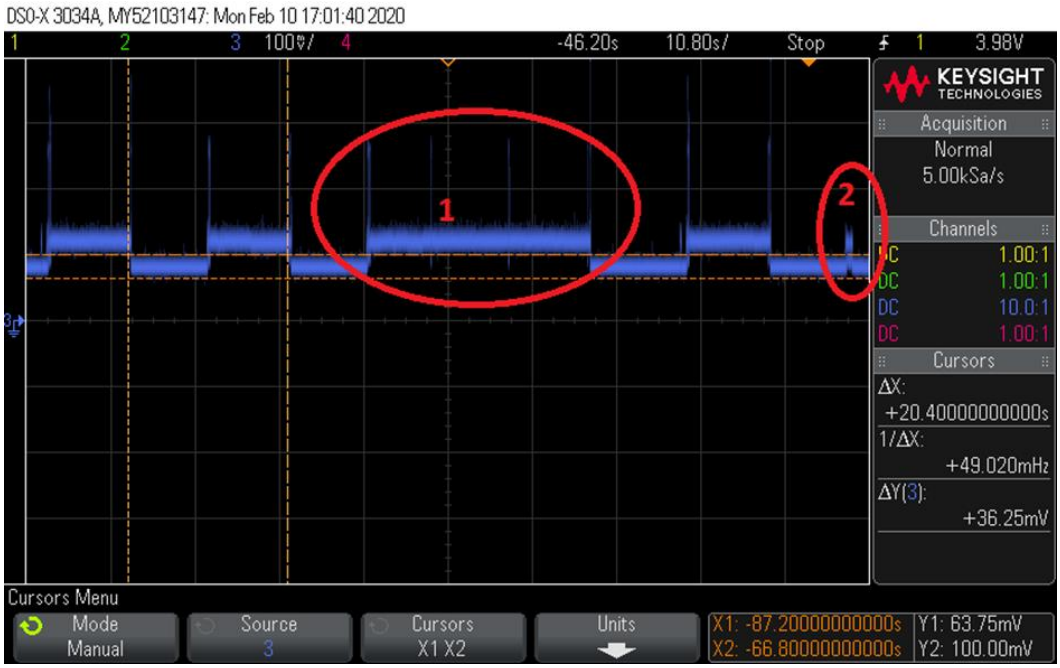

Figure 26. eDRX with a cycle length duration of 20.48 seconds.

From tests it came out that this mode is not consistent, and this is also visible in the figure as the test for this case should send data every 22 seconds (software delay added), with eDRX 20.48 seconds. Firstly, there are power consumption spikes at about every 10 seconds, instead of 22 seconds for data transmission. Secondly, the eDRX is not consistent, as sometimes it is disabled for the longer period (Stage 1), or it is enabled for

longer period, as can be seen at the start of Stage 2. This unpredictable power consumption fluctuations and data transmission errors make this power saving mode unreliable.

When testing various cases with this module, it was seen that when the IoT device wants to send data before eDRX ends (before the 20,48 seconds), the data will not be sent out, which should not happen, as eDRX disables the receiving of NB-IoT, not transmitting. This also causes a problem for this smart elevator system, as the data should be sent out after a human is detected, not after the eDRX timer runs out.

This is the only power saving mode, where Arduino delay was used instead of sleeping, since this library has only 1, 2, 4, 8 second(s) and forever sleep mode and making interrupts to wake the Arduino to send data about every 20 seconds would make the software and hardware more complicated to compare with other power saving modes. That is why the Arduino power consumption is higher, compared to other power saving modes.

6.4 Formulas and calculations

Before the average current of the IoT device can be calculated, the average current draw during connection event has to be calculated with the following formula:

$$Avg.current\ during\ connection\ event = \frac{\begin{array}{c}(Stage\ 1\ time) * (Stage\ 1\ current) + \\ (Stage\ 2\ time) * (Stage\ 2\ current) + \\ \ldots + (Stage\ n\ time * Stage\ n\ current\ )\end{array}}{\begin{array}{c}Total\ awake\ time\ during \\ one\ connection\ event\end{array}}$$

As long as the units of time and current consumption are used consistently in the formula, it provides accurate results of current consumption during connection event. The tables from the Measurement and analysis sub-chapters are used to calculate the average current consumption during connection event for power saving modes that are reliable and working correctly.

The next step is to calculate the average current consumption for the entire connection interval, which takes into account the time during which the device is sleeping. For this, the next formula can be used:

$$Total\ average\ current =$$

$$= \frac{\left[\begin{array}{c}(\text{Connection Interval} - \text{Total awake time}) * (\text{Average sleep current}) + \\ (\text{Total awake time}) * (\text{Average current during connection event})\end{array}\right]}{(Connection\ interval)}$$

The connection interval is one hour (3600 seconds) and total awake time is calculated based on this cycle. Total awake time depends on 2 variables: awake time for one cycle and average number of cycles in one connection interval. Awake time for one cycle is calculated using the data in the Measurement and analysis sub-chapters and the average number of cycles in one connection interval is calculated based on the elevator usage in the TalTech ICT office building (Figure 27[1]). In the figure, vertical axis represents average usage per hour, and the horizontal axis represents the time in hours, floors are displayed with different colours. For instance, 5th floor average elevator usage per 24-hour cycle is 2.9 times an hour.

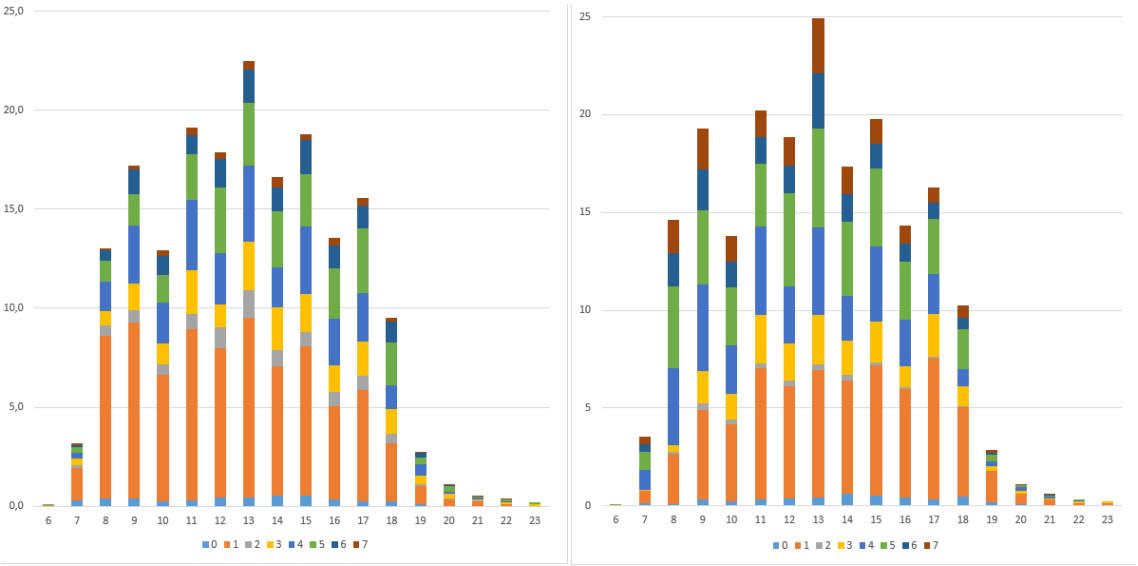

Figure 27. Average usage of TalTech ICT elevator per hour, left figure shows start floor and right figure destination floor.

And lastly, the following formula can be used to calculate the expected battery life of this IoT device in different power saving modes. As the IoT device uses 2 Li-Ion 18650 3.6V batteries in series, the voltage is increased, however the capacity stays the same - 3350 mAh.

[1] These data are provided by the group working on the KONE smart elevator project.

$$Expected\ battery\ life = \frac{Battery\ capacity}{Total\ average\ current}$$

Average current consumption and expected battery life of each reliable power saving mode is shown in the Table 12, based on the previous formulas [31].

Table 12. Average current consumption and expected battery life of each reliable power saving mode.

Power saving mode	**Awake time during one connection event (s)**	**Total awake time (s)**	**Current consumption during connection event (mA)**	**Total average current (mA)**	**Expected battery life (h)**
Power down mode	17.8640	51.8056	79.5622	21.7761	153.8384 (~ 6.4 days)
Airplane mode	11.7480	34.0692	69.6780	37.6560	88.9632 (~ 3.7 days)
Minimum functionality mode	11.7480	34.0692	67.6932	37.5051	89.3212 (~3.7 days)
IDLE	0.9133	2.6487	73.5184	68.8488	48.6573 (~2 days)

Based on the data shown in table above, it can be seen that the highest expected battery life is achieved by using powered down mode on NB-IoT. However, the awake time during one connection event is also highest, meaning it takes the most time to send the data out, which means this mode is not suitable to use, if the system has time constraints. Idle mode is the opposite of powered down mode: the expected battery life and awake time during one connection event are the lowest. This mode is best suitable to use, if the system has time constraints and the data needs to be sent out quickly after the module is triggered. Minimum functionality and airplane mode are very similar in terms of power consumption and awake time during one connection event. These modes are used in cases, where awake time during one connection event and expected battery life are not critical.

7 PIR based IoT device analysis

During this master's thesis IoT device for smart elevator system was designed and programmed to detect humans with a PIR sensor and notify the elevator server autonomously using wireless NB-IoT technology. Developed compact and working NB-IoT prototype device can be placed on top of the KONE smart elevator entrance in TalTech ICT office building. This smart solution could make the elevators even more autonomous, as in the future it could make reservation calls for elevator passengers without human interaction.

PIR based IoT device prototype uses an Arduino microcontroller as it was meant to be compatible with both NB-IoT sensor shields (Avnet silica NB-IoT sensor shield and Dragino NB-IoT sensor shield). Firstly, the NB-IoT had to be tested and configured, to see if and how this technology works. Low power microcontroller unit (MCU) was not required for this part. After getting the conformation through testing, that the NB-IoT communication works with the private network, the software code for the Arduino was further developed for the smart elevator system, to know if the proposed solution would work. Lastly, different NB-IoT power saving modes were tested with the already written software for the Arduino. This allowed the author faster prototyping and testing without focusing on finding the low power MCU and reprogramming the existing software into another language for the MCU. Although, low power MCU should be used with NB-IoT modules to make these smart elevator IoT devices have a long lasting battery life. However, the NB-IoT technology should be more deeply developed and researched beforehand, as the time to make a connection with users selected application server takes too much time.

7.1 Power saving modes

One of the key aspects of IoT devices is power reduction, which should be done to achieve long battery life on these devices. For this IoT device, power consumption is primarily measured for NB-IoT power saving modes, as this is new and underused technology. The IoT device uses Arduino as the main controller, however, this MCU can easily be swapped with a low power MCU to save more power. The only criterion for the MCU

would be to have an UART interface, as this Dragino NB-IoT sensor shield (BG96) uses it to communicate with the host.

Chosen IoT device has quite large energy consumption, as it can be seen from the chapter 6.4. Firstly, the most energy consuming hardware for this IoT device is Arduino, which consumes about 21 mA in sleep mode and even more in the full operating mode to process data and communicate with NB-IoT over UART interface. However, as mentioned before, this problem can easily be solved by swapping out the Arduino with low power MCU. Secondly, the intended and most promising power saving mode for NB-IoT, PSM, did not work, as explained in details in chapter 6.3.2.1.

Analysis of these power saving modes revealed that the most suitable mode of operation for this IoT device would be idle mode, as with using other power saving modes, the power consumption would be reduced about 2-3 times, but the delay from detecting an elevator passenger to the IoT device sending this data out would be increased about 10-17 times, which would not be acceptable.

7.2 Time delay

This IoT device for smart elevator is not time critical, meaning the connection with an elevator server does not need to be made within ms after detecting elevator passengers - it can handle small delays. Time from detecting an elevator passenger to elevator server getting the information can be divided into two parts:

- Time from detecting an elevator passenger to IoT device sending the data through NB-IoT
- NB-IoT transmission time over the air

The first part of the time delay depends on the NB-IoT power saving mode used and is shown in Table 13. The data is based on the results gathered in 6.3 sub-chapter.

Table 13. Time delay from detecting an elevator passenger to IoT device sending the data through NB-IoT.

Power saving mode	**Time (s)**
Power down mode	16.2980
Airplane mode	9.2400
Minimum functionality mode	9.4050
IDLE	0.9133

As it can be seen, the first part of the time delay is already quite huge when using power saving modes that require re-establishing a connection with the elevator server. This time delay is mainly caused by the NB-IoT module, as it needs 2 second delay between each AT command to successfully complete its tasks, as described in chapter 6.2.1.

The second part of the time depends on Telia's network usage and it can theoretically be up to 9-10 seconds. There are two ways to find out the NB-IoT transmission time, first option would be to ping the elevator server to get the round trip time, second option would be to use a script on the elevator server to receive the TCP packets with a timestamp from the IoT device.

To get more precise ping results, three different test cases were conducted for the period of 24 hours per test:

- Pinging cycle 8 sec (PIR maximum detection interval)
- Pinging cycle 20.69 min (Overall average TalTech ICT office elevator usage per hour)
- Pinging cycle 7.23 min (Maximum TalTech ICT office elevator usage per hour)

The ping results are shown in the Table 14, however it should be noted that ping returns round trip time, not exact transmission time, and this value cannot be divided in half to get one way transmission delay time. Moreover, the pinging cycle means that the time delay between each ping command from Arduino is started after the NB-IoT has replied the previous ping results to Arduino, as the NB-IoT sensor shield cannot communicate with Arduino through UART interface before the NB-IoT module has finished its previous task and replied with “OK”.

Table 14. Ping results of 3 different test cases.

Pinging cycle	**Packets sent**	**Packets lost**	**Minimum delay time (ms)**	**Maximum delay time (ms)**	**Average transmission delay time (ms)**	**Packet loss (%)**
8 sec	9702	155	91	6227	1036.023	1.6
20.69 min	71	0	102	1944	905.479	0
7.23 min	202	1	361	2253	1056.214	0.5

Performed tests show that the minimum (91 ms) and maximum (6227 ms) delay time was acquired from the first test, where pinging cycle was 8 seconds. It was found that the more packets sent at a shorter interval, the greater probability of losing packets. The average overall transmission delay time based on these tests stayed close to 0.9 to 1 second.

For the second option to find out the transmission time, the IoT device had to sync time with NTP server in Estonia, after which the current time could be acquired from NB-IoT module's hardware. This time is sent out to the elevator server over TCP, which then is saved to a server log file along with server's current time. These time values can be used to calculate the NB-IoT transmission time over the air.

Since TCP guarantees packet delivery and no packets are lost, only one 24-hour test case was conducted (Table 15). However, the data sending cycle is increased by 2 seconds. The possible reason could be the following: NB-IoT sensor shield does not return a current timestamp from hardware, if the delay time is less than 10 seconds before the next UDP/TCP related AT command. It should also be noted that the inaccuracy of these results could be influenced by the following factors:

- Transmission delay includes AES-128 encryption by the Arduino, which takes about 100 ms
- The NB-IoT sensor shield does not return the current time with millisecond precision, but to the nearest second
- The elevator server and NB-IoT sensor shield module are probably not synchronized to the same NTP server

Table 15. Average transmission delay time using TCP.

Data sending cycle	**Packets sent**	**Minimum delay time (s)**	**Maximum delay time (s)**	**Average transmission delay time (s)**
10 sec	8793	-0.980	29.912	2.531

As mentioned before, the results are not 100% accurate, which is why the minimum delay time is negative. However, the maximum delay time compared to the ping results is almost 5 times greater, as the TCP guarantees packet delivery and does not allow the packets to be lost, therefore we assume retrasmition of the packet many times here, which increases the delay. The average transmission delay time is also greater than the average ping result time. When comparing the results of ping delay and TCP transmission delay, it also came out that the delay in the TCP option was significantly higher during the daytime, from about 5:45 to 21:40, however this did not appear to happen with ping results.

The overall time delay for this IoT device using different power saving modes is shown in the Table 16 below, taking into account the TCP average transmission time, as the packets are sent out exactly the same way the TCP average transmission delay time was measured.

Table 16. Overall delay for this IoT device using different power saving modes.

Power saving mode	**Total time delay**
Power down mode	18.829
Airplane mode	11.771
Minimum functionality mode	11.936
IDLE	3.4443

As mentioned in the previous chapter 7.1, IDLE mode was selected for this smart elevator project. This means the average delay time from detecting an elevator passenger to the elevator server getting the information is about 3.4 seconds, which is about 3.5 to 5.5 times faster compared to other power saving modes.

8 Passenger detection with thermal camera

Passenger detection solution with a new small 1024 (32 x 32) channel thermal camera is more reliable than the PIR based solution, as this thermal camera does not trigger when there are passers-by. However, it takes more time to detect the elevator passenger, as the elevator passenger is distinguished from passers-by by the time spent in the area of interest. This distinguishing process is more deeply discussed in the chapters 8.1 and 8.2.

8.1 Scenarios

The passenger different behaviours can be categorized as follows:

- Elevator passenger is coming from the elevator
- Elevator passenger is waiting for the elevator
- Passer-by is walking past the elevator

As mentioned in chapter 4.2.2, all the scenarios were captured with a visualizing device, which saved the different scenarios with RGB and thermal camera to the SD card. The device was placed in the dorm room approximately at the same height (2.3 m) and at an angle of 45 degrees, so that the current scenario would be visible to both cameras, as they have different FOV (62.2° x 48.8° for the RGB camera, and 90° x 90° for the thermal camera). Due to the worldwide pandemic, all the scenarios were played out in the dorm room with the following limitations:

- Only one person conducted experiments
- Persons size did not change
- Persons body temperature did not change
- Persons clothes were the same
- Room temperature did not change

The visualizing device captured the scenarios with 300 ms intervals, as this is the thermal camera output data refresh rate. However, many temperature packets were lost between the thermal camera, SMT32 microcontroller and Raspberry Pi, as the UART interface is slow and not the most reliable. Instead, I2C or SPI could be used in the future, to play the scenarios without the mentioned limitations, as they are faster than UART. For instance, D6T thermal camera also uses I2C, as it outputs large amount of data every 300 ms and

it has minimal loss of data packets. This UART is not a confounding factor, as it is only used to test and compare the scenarios. The proposed thermal camera solution has I2C connection between STM32 microcontroller and D6T thermal camera, where packet loss is very unlikely.

These scenarios were played out many times in a row, in order not to affect the research results because of the loss of packets. All together 165 RGB pictures and thermal camera outputs were saved to analyse the scenarios (chapter 8.2) and to trigger the device only if the elevator passenger is detected.

8.2 Analysis of the scenarios

Analysis of the previously mentioned scenarios was done with the help of visualizing and analyzing software. The analysing consists of 3 steps:

- Where to detect humans
- Detect humans in that area
- Distinguish between an elevator passenger and a passer-by

Firstly, the area of interest had to be calculated and converted into thermal camera output pixels, which could later be used to detect elevator passengers. As mentioned in chapter 4.1.2, the area of interest in front of the elevator is 230 x 180 cm. This value must be converted into pixel values that D6T-32L-01A thermal camera uses as seen in the Figure 28. The thermal camera FOV is 32 x 32 pixels in x- and y-direction, and this corresponds to 90° in x- and 90° in y-direction, which means, each pixel will be 2.8125° in x- and y-direction.

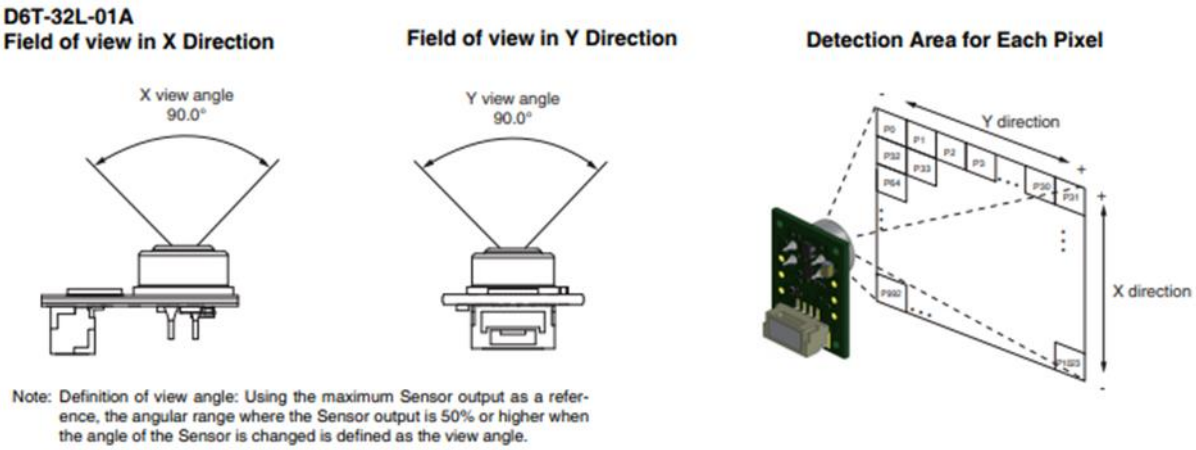

Figure 28. D6T-32l-01A FOV and detection area for each pixel [29].

Since the camera optimal placement is at 2.3 m and at a 45° angle, basic trigonometry algorithms can be used to calculate the area of interest into pixels. In Figure 29, green colour area indicates the 3D area of interest. Based on the height of the camera and the length from the furthest side of the elevator passenger detection area to the elevator, the number of pixels in x-direction in the area of interest can be calculated:

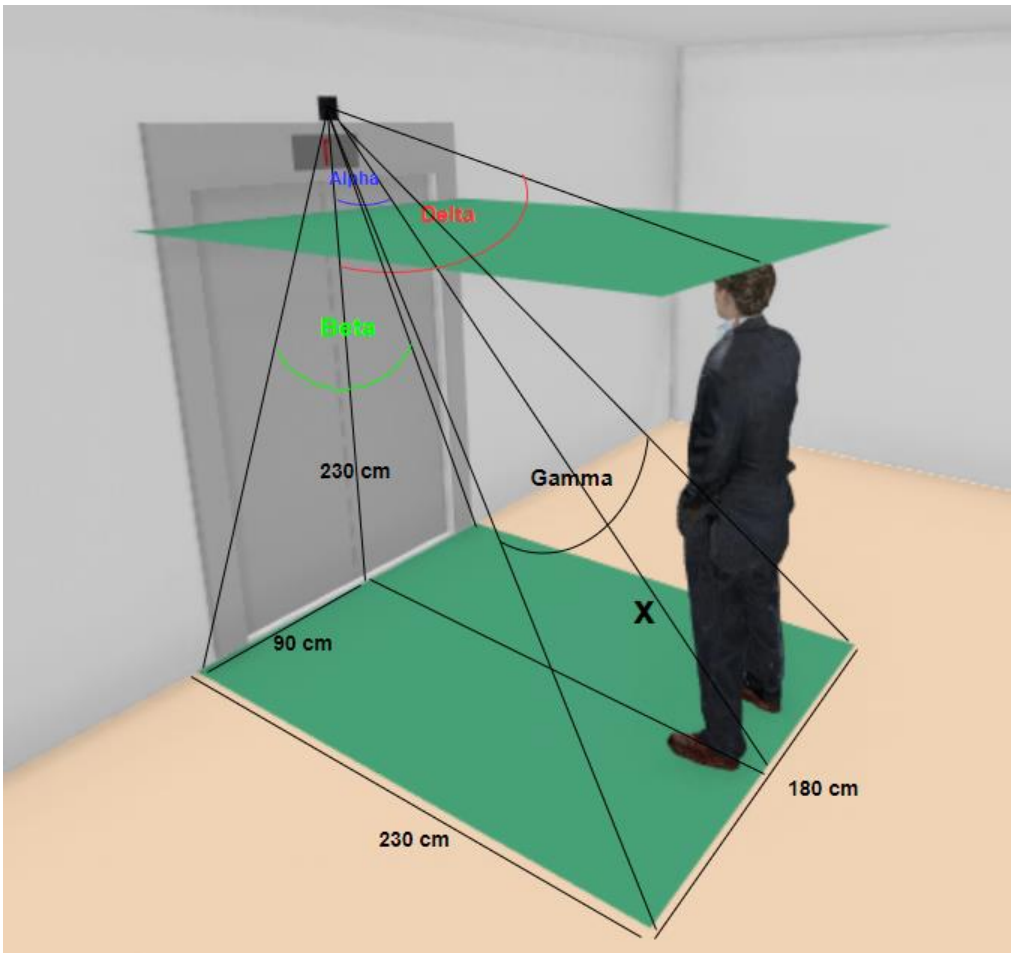

Figure 29. Area of interest and IoT device placement.

$$\alpha = \tan^{-1}(\frac{230}{230}) = 45°$$

This alpha can be used to calculate the pixels, which detect the area of interest vertically:

$$\frac{45°}{2.8125°} = 16\,pixels$$

This corresponds to 16 bottom rows of the thermal camera output to be in the area of interest vertically.

Since the camera is not perpendicular to the ground, 3D image will be displayed in 2D, meaning y-direction pixels in the area of interest depend on the x-direction pixels. With the width of the area of the interest and height of the camera y-direction area of interest, pixels can be calculated for the closest y-direction row to the elevator.

$$\beta = 2 * \tan^{-1}\left(\frac{90}{230}\right) \approx 42.74°$$

$$\frac{42.74°}{2.8125°} \approx 16\ pixels\ (Rounded\ up\ to\ an\ even\ number)$$

Since the y-direction pixels that depend on the x-direction pixels change linearly, furthest y-direction row from the elevator can be calculated next. Firstly, distance from the camera to the furthest area of interest side has to be calculated:

$$X = \sqrt{230^2 + 230^2} \approx 325.27\ cm$$

Now the y-direction pixels can be calculated for the furthest area of interest side.

$$\gamma = 2 * \tan^{-1}\left(\frac{90}{325.27}\right) \approx 30.93°$$

$$\frac{30.96°}{2.8125°} \approx 12\ pixels\ (Rounded\ up\ to\ an\ even\ number)$$

Finally, the area of interest has to be made into 3D to identify whole human body in that area - which means average human height (160-170 cm) has to be taken account. For the camera to detect taller people almost completely, 190 cm sketch was used. To calculate Delta, length of the area of interest and length difference between the sketch elevator passenger's height and the height of the camera must be used. Now the number of pixels in x-direction can be calculated that represents the area in 3D.

$$\delta = \tan^{-1}\left(\frac{230}{230 - 190}\right) \approx 80.13°$$

$$\frac{80.13°}{2.8125°} \approx 28\ pixels$$

All in all, the 3D area of interest converted into red pixels is presented in Figure 30 in the 32 x 32 matrix output.

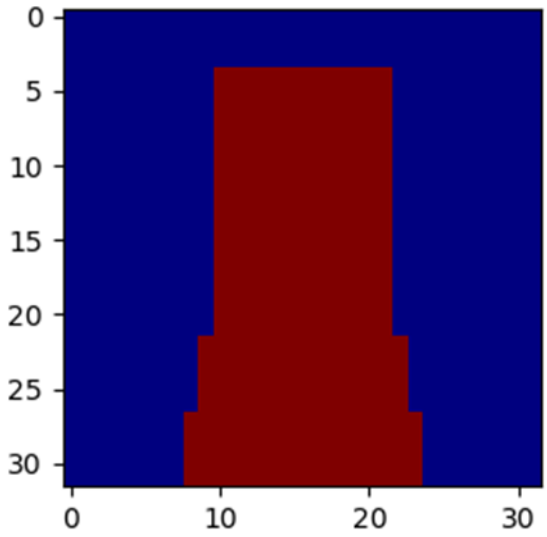

Figure 30. Area of interest in the thermal camera output.

The second step is to detect humans in this area, however, average human body temperature cannot be used, as the thermal camera output temperature also depends on the distance of the object, surrounding air temperature and other factors, meaning the average room temperature, lowest human body temperature and number of detected human pixels should be calculated with the previously gathered data. For these scenarios, the average room temperature was 22 degrees, the human had a body temperature over 25 degrees regardless of the distance and position, with an average of about 40 pixels for person while standing still, and about 20 pixels while moving.

Finally, after the human is detected in the area of interest, it must be determined whether he or she is an elevator passenger or a passer-by. To do this, average time for passer-by to be in the area of interest can be used, which depends on the maximum distance in the area of interest and the average human walking speed (1.4 m/s). As the longest distance in the area of interest is diagonally, simple Pythagoras theorem can be used:

$$\sqrt{230^2 + 180^2} \approx 2.92\ m$$

And the average time to walk this distance is:

$$\frac{2.92}{1.4} \approx 2.09\ s$$

This is the average maximum time for the passer-by to be in the area of interest, which can be calculated into frames, taking into account the data output update rate (300 ms): $\frac{2.09}{0.3} \approx 7\ frames$

Meaning, if the human is detected in the area of interest for longer than 7 frames, the MCU should trigger. However, this only happens, if no thermal camera outputs were lost and the passer-by walks diagonally past the elevator, which is unlikely.

On the other hand, mostly the passer-by walks past the elevator horizontally, meaning the average time to walk this distance will be:

$$\frac{1.8}{1.4} \approx 1.29\ s$$

$$\frac{1.29}{0.3} \approx 4\ frames$$

For most of the cases, the elevator passenger is distinguished, if the cluster of over 20 pixels is detected in the area of interest for more than 4 consecutive frames. Detection of the same cluster but for less than 4 frames is considered as a passer-by or as an elevator passenger, who just left the elevator.

9 Future work

Smart elevator system project can be developed further in many ways, especially when improving elevator passenger detection to make the reservation calls automatic. The current proposed IoT device uses PIR to detect movement and sends the information to the elevator server using NB-IoT technology. The second, improved method to detect only elevator passengers can be used in conjunction with the proposed IoT device to additionally reduce the power consumption of the second method by only activating thermal camera when movement is detected.

Moreover, to improve the reliability of the elevator passenger detection, the scenarios in chapter 8.1 should be played out without the mentioned limitations, as these factors affect the thermal camera output. This means the scenarios should be done with different people with different clothing in different room temperatures, including scenarios, where people come from the cold environment (winter).

Hypothetical solution to faster distinguish elevator passengers from passers-by with thermal camera is to take into consideration the change in human walking speed: if the human is slowing down in the area of interest, there is a high probability that he/she is an elevator passenger, otherwise passer-by.

Another suggestion would be to place sensors that detect elevator passengers further away from the elevator, to detect passengers earlier and make a reservation call for them before they even get to the elevator entrance. However, this requires the office buildings to be with a certain architecture, to enable the elevator passenger identification much earlier, just as it was developed by Kwon et al. (described in the chapter 1.2).

Last but not least, different NB-IoT communication technology modules should be looked into to find ways to remove huge and probably unnecessary delays between AT commands. For example, SARA-N3 series, multi-band NB-IoT (LTE Cat NB2) modules [35].

10 Summary

The main purpose of this thesis was to prototype a wireless IoT device that could be used in the future to make automatic elevator reservation calls for elevator passengers. This consists of finding a solution to two questions related to the smart elevator system:

- How to detect elevator passengers?
- How to send the analysed and processed information received from sensors to the elevator server wirelessly from staircase?

During this research, different passenger detection sensors suitable for elevator were compared. Based on the available information and specification, two elevator passenger detection sensors, which were suitable for this IoT device under the given conditions, were selected and tested. Furthermore, low power communication technologies were also compared and NB-IoT was introduced, as it outperforms all other LPWAN technologies in terms of coverage, energy efficiency, and cost by utilizing the existing cellular infrastructure.

Along with the wireless PIR based IoT device prototype, thermal camera based solution was proposed to detect only elevator passengers, not everything that moves. To test its reliability, visualizing device was made to compare and analyse the thermal camera output with the real life scenarios. This device saves the thermal camera and RGB camera output to the SD card, which can be visualized and analysed with a corresponding python software made by the author. Finally, the elevator server software was created to collect the wireless PIR based IoT devices data. This TalTech smart elevator server system is made secure from most of the threats by author and inaccessible to third parties in cooperation with TalTech.

Different NB-IoT power saving modes were tested: delay and current consumption were measured and analysed to know which mode would be the best suitable for this smart elevator system. Unfortunately, not all the power saving modes could be tested, as one of them is not yet properly developed (PSM) and for the other (sleep mode), there is no documentation of the connection pins. Besides that, this module requires huge delays between AT commands, which, however, makes the use of power saving modes unusable for this given smart elevator project, as the connection with the elevator server needs to

be made every time the module wakes up from power saving mode. The only suitable mode for this smart elevator system is idle, although it consumes 2-3 times more power when compared to other power saving modes, it does not need to make a new connection to the elevator server every time passenger is detected, which means the elevator server gets the right information 3.5-5.5 times faster.

All together, two prototypes of PIR based IoT devices, thermal camera based solution and visualizing device were made, along with the elevator server and visualizing and analysing software. Therefore, all the goals set for this thesis were met, however, the adequacy of this IoT device is questionable, as the power consumption is too high, and the long delay time would not make reservation calls reasonable for the elevator passengers.

References

[1] N. Saloio, “Elevators are Getting Smarter,” Building Owners and Managers Association (BOMA), [Online]. Available: https://fmlink.com/articles/elevators-are-getting-smarter/. [Accessed 28 04 2020].

[2] Landmark Elevator, 15 02 2016. [Online]. Available: https://landmarkelevator.com/history-of-elevator-technology/. [Accessed 28 04 2020].

[3] J. Fernandez, P. Cortés, J. Muñuzuri and J. Guardix, “Dynamic fuzzy logic elevator group control system with relative waiting time consideration,” IEEE, 2014.

[4] E. O. Tartan, H. Erdem and A. Berkol, “Optimizing the intelligent elevator group control system by using genetic algorithm,” IEEE, 2014.

[5] J. Liu and Y. Liu, “Ant colony algorithm and fuzzy neural network-based intelligent dispatching algorithm of an elevator group control system,” IEEE, 2007.

[6] H. Ge, T. Hamada, T. Sumitomo and N. Koshizuka, “Intellevator: A Context-Aware Elevator System for Assisting Passengers,” IEEE, 2018.

[7] H. Ge, T. Hamada, T. Sumitomo and N. Koshizuka, “PrecaElevator : Towards Zero-Waiting Time on Calling Elevator By Utilizing Context Aware Platform in Smart Building,” IEEE, 2018.

[8] M. Turunen, H. Kuoppala, S. Kangas, J. Hella, T. Miettinen, T. Heimonen, T. Keskinen, J. Hakulinen and R. Raisamo, “Mobile Interaction with Elevators: Improving People Flow in Complex Buildings,” in *ResearchGate*, 2013.

[9] O. Kwon, E. Lee and H. Bahn, “Sensor-aware elevator scheduling for smart building environments,” in *Elsevier*, 2014.

[10] C. Jiang, M. K. Masood, Y. C. Soh and H. Li, “Indoor occupancy estimation from carbon dioxide concentration,” in *Arxiv*, 2016.

[11] “Libelium,” Libelium, [Online]. Available: http://www.libelium.com/products/meshlium/smartphone-detection/. [Accessed 28 04 2020].

[12] Berkeley Varitronics Systems, [Online]. Available: https://www.bvsystems.com/industries-2/cell-phone-detection/. [Accessed 28 04 2020].

[13] OMRON, [Online]. Available: https://omronfs.omron.com/ko_KR/ecb/products/pdf/en_D6T_users_manual.pdf. [Accessed 28 04 2020].

[14] “RapidOnline,” TruSens, [Online]. Available: https://components101.com/sites/default/files/component_datasheet/HC-SR505%20Datasheet.pdf. [Accessed 28 04 2020].

[15] Raspberry Pi, [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/. [Accessed 28 04 2020].

[16] “NXP Semiconductors,” [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf. [Accessed 28 04 2020].

[17] “Sparkfun,” [Online]. Available: https://www.sparkfun.com/products/15569. [Accessed 28 04 2020].

[18] “Velleman,” [Online]. Available: https://www.velleman.eu/products/view/?id=435532. [Accessed 28 04 2020].

[19] “NB-IoT Webpage,” [Online]. Available: https://www.ttu.ee/institutes/thomas-johann-seebeck-department-of-electronics/nb-iot/. [Accessed 29 04 2020].

[20] C. B. Mwakwata, H. Malik, M. M. Alam, Y. L. Moullec, S. Parand and S. Mumtaz, “Narrowband Internet of Things (NB-IoT): From Physical (PHY) and Media Access Control (MAC) Layers Perspectives,” Sensors, 2019.

[21] S. Z. Khan, H. Malik, J. L. R. Sarmiento, M. M. Alam and Y. L. Moullec, “DORM: Narrowband IoT Development Platform and Indoor Deployment Coverage Analysis,” in *Elsevier*, 2019.

[22] H. Malik, H. Pervaiz, M. M. Alam, Y. L. Moullec, A. Kuusik and M. A. Imran, “Radio Resource Management Scheme in NB-IoT Systems,” in *IEEE*, 2018.

[23] “[SPEC] 3GPP TR 45.820 – Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT),” iTecTec, [Online]. Available: https://itectec.com/archive/3gpp-specification-tr-45-820/. [Accessed 28 04 2020].

[24] I. Guitor, “NUTIMAJA EDASIARENDUS LORA JA NBIOT RAADIOVÕRGU NÄITEL,” TalTech, 2018.

[25] “Avnet-Silica BAENBIOTBG96SHIELD-E Board’s documentation,” Avnet Silica, 30 05 2019. [Online]. Available: https://avtech.emea.avnet.com/cont/ . [Accessed 29 04 2020].

[26] “NB-IoT Shield,” Dragino, 29 07 2018. [Online]. Available: https://wiki.dragino.com/index.php?title=NB-IoT_Shield. [Accessed 29 04 2020].

[27] Y. S. Lee and D. M. Kim, “PRODUCT SPECIFICATION Rechargeable Lithium Ion Battery Model : INR18650 F1L,” LG Chem, 04 03 2015. [Online]. Available: http://keeppower.com.ua/download/2015-11/150304_PS_LGC_INR18650_F1L_v1.pdf. [Accessed 29 04 2020].

[28] “MEMS Thermal Sensors D6T,” OMRON, [Online]. Available: http://components.omron.eu/getattachment/b02cde93-fa0a-4d46-9806-0da0c7a093f2/A274-E1-02.pdf.aspx. [Accessed 29 04 2020].

[29] “MEMS Thermal Sensors D6T User’s Manual,” OMRON, [Online]. Available: https://omronfs.omron.com/ko_KR/ecb/products/pdf/en_D6T_users_manual.pdf. [Accessed 29 04 2020].

[30] “Narrowband – Internet of Things (NB-IoT),” GSMA, [Online]. Available: https://www.gsma.com/iot/narrow-band-internet-of-things-nb-iot/. [Accessed 29 04 2020].

[31] S. Kamath and J. Lindh, “Measuring Bluetooth® Low Energy Power Consumption,” Texas Instruments, [Online]. Available: https://www.ti.com/lit/an/swra347a/swra347a.pdf. [Accessed 29 04 2020].

[32] Quectel, “BG96 Hardware Design,” Quectel, 2019.

[33] “NB-IoT Deployment Guide to Basic Feature set Requirements Version 1.0,” GSMA, 02 08 2017. [Online]. Available: https://www.gsma.com/newsroom/wp-content/uploads/CLP.28v1.0.pdf. [Accessed 29 04 2020].

[34] “Ultra IoT C-SGN Guide, StarOS Release 21.3 Chapter: Power Saving Mode (PSM) in UEs,” CISCO, 06 09 2017. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-3_N5-5/Ultra_IoT_CSGN/21-3-Ultra-IoT-CSGN-Guide/21-3-Ultra-IoT-CSGN-Guide_chapter_0110.html. [Accessed 29 04 2020].

[35] “SARA-N3 Series,” U-Blox, [Online]. Available: https://www.u-blox.com/en/product/sara-n3-series. [Accessed 02 05 2020].

Appendix 1 – Software repository

The software codes, BG96 documentations and analyzing results for this Master's thesis are stored in https://gitlab.pld.ttu.ee/inguit1/Indrek_Guitor_Masters_Thesis. This repository has internal access level limited only to people with TalTech Uni-ID as it contains sensitive data.