Access Control System Based on UHF RFID Technology

Smart Button for Nursing Homes

Bennebroek. A.Q. 4701267
Ramezani. S. 5052025

Department of Electrical Engineering
Delft University of Technology

December 26, 2022

TU Delft

TU Delft

Access Control System Based on UHF RFID Technology

BSc Thesis

By
Arthur Bennebroek and Shayan Ramezani

Assigned by
Momo Medical

December 26, 2022

Student number:	Arthur Bennebroek	4701267
	Shayan Ramezani	5052025
Project duration:	October 17, 2022 - December 23, 2022	
Thesis committee:	Dr. P.J. French	TU Delft, supervisor
	D. Eldering	Momo Medical, supervisor
	J. van Rijn	Momo Medical, supervisor
	Dr. ir. I.E. Lager	TU Delft
	Ing. J. Bastemeijer	TU Delft

Delft University of Technology

Faculty of Electrical Engineering, Mathematics
and Computer Science

Electrical Engineering Programme

Abstract

To make open door policies easier in nursing homes, it is important to have a smart system that can check whether access is allowed for the person trying to enter some place or not. This thesis documents how an access control system has been built, for Momo Medical, to be used at a nursing home. It sheds light on hardware considerations made and the software written for the final product. All the knowledge gathered and the time spent has led to a final product that with minor tweaks can be used in the existing infrastructure of Momo Medical but also without the presence of it. There are however still improvements left to be made for the final product to be commercially appealing.

Preface

This bachelor graduation project has been an interesting and challenging experience. As a team, we wanted to work on a project that would have tangible results and not only did we find such a project, we also became part of a team that was really welcoming and open.

We have learned a few important things in this project. One of them is that when it comes to finding the right hardware, experience is really important. It is crucial to have access to different kinds of hardware and have some process of trial and error for finding the best hardware. Another lesson, which comes from seeing the Momo Medical team, is how important team dynamic can be and that we do not need to always follow conventional hierarchies in a company. The team also taught us about sprint meetings according to the SCRUM method, retrospective meetings to reflect on the way of working, and the legendary VrijMiBo, to name but a very few!

We hope that this thesis will help future students to have a feeling of how the bachelor graduation project can go at TUDelft. For readers only interested in getting a overview of the project and the results, the abstract, introduction, conclusions in different chapters in the body of the thesis and the discussion and conclusion can be the most useful. For readers interested in the hardware possibilities for comparable systems, chapter 3 can be most useful.

We would like to first hand show our gratitude for the support and coaching we got from the Momo Medical team, especially Danny Eldering and Joey van Rijn for being always available for questions and taking part in our meetings. We would also like to thank Dr. Paddy French for his time supervising us and Dr. Ioan Lager and Menno Gravemaker for giving us this opportunity. And last (but not least), we want to thank subgroup two (Marijn Boringa, Job van Erp, Corné Ploumen) for their passion and all the joyful moments we have gone through together.

Delft, The Netherlands
December 26, 2022
Arthur Bennebroek and Shayan Ramezani

Contents

List of Abbreviations

ACS Access Control System. 3–6, 9, 10, 12, 13, 21, 23, 29–35, 50

BLE Bluetooth Low Energy. 3, 7, 8

CRC Cyclic Redundancy Check. 24–28

EPC Electronic Product Code. 27, 32, 33

ERP Effective Radiated Power. 17

GPIO General Purpose Input/Output. 10–12, 24, 32

HF High Frequency, the frequency band between 3 and 30 MHz. iv, 9

I2C Inter-Integrated Circuit communication protocol. 11

IoT Internet of Things. 3, 7

IR Infrared. 7

KaM Keep-alive Message. 26

NFC Near Field Communication. 7–9

OOK On-Off Keying. 14

PIE Pulse-Interval Encoding. 14

PMO Pulse Mode Operation. 29

PR-ASK Phase Reversal - Amplitude Shift Keying. 14

RF Radiofrequency. 7, 10

RFID Radio Frequency Identification. iv, 3, 7–15, 17, 20, 21, 32, 36

RSSI Received Signal Strength Indicator. v, 9, 11, 12, 20, 21, 23, 26, 27, 31–33

SPI Serial Peripheral Interface. 11, 16

SRAM static random access memory. 10, 11

TFM Tag Found Message. 27

TLM Temperature Log Message. 26

TTL Transistor–transistor logic. 13

UART Universal Asynchronous Receiver-Transmitter. iv, v, 10, 11, 13, 15, 23–25, 29, 30

UHF Ultra High Frequency, the frequency band between 300 and 3000 MHz. iv, 9–12, 14–17, 21, 32, 36

1 Introduction

Momo Medical[1], a startup company, is developing new technologies so that care givers can work more efficiently in the nursing homes and therefore are able to give more specific and better care to their residents. At the moment, their main product is a sensorplate, the BedSense, which is placed underneath the mattress of a bed and together with the corresponding app gives a better insight into the behavior and the needs of the resident while he/she is sleeping. It notifies the care giver when a resident is about to get out of bed and gives insight in the state of a residents wellbeing. Also they designed a smart incontinence pad which lets the care giver know if the incontinence pad is full and needs to be changes. They are always looking for more ways to help the care givers work more efficiently.

1.1. Problem Definition

Currently most nursing homes are leaving the concept of closed departments behind. In this old system, residents were locked up in their own department and were not allowed to move freely through the nursing home. In this case, locked up literally means locking the door between the different departments so that the residents were not able to leave their own department. As this is ethically questionable, more and more nursing homes are switching to a system with open departments in which the residents are allowed to move more freely between different departments. However, this introduces new challenges, such as residents being able to wander to places they are not allowed to go and that whenever a resident needs help, they are much more difficult to find.

One problem, regarding the change to open departments, is that some but not all residents are allowed to go to other departments. Psychogeriatric residents are not allowed to leave their department while the other residents are allowed to leave. To solve this, a system needs to be designed that is able to recognize which resident is trying to pass through a door to another department. Depending on whether or not they are allowed to go there, the system locks or unlocks the door. This system can also notify care givers when someone is trying to pass a door, while they are not allowed to.

In most of these nursing homes, residents wear a panic button, which they can press if they need help. Pressing this button will send a signal to the care giver that the resident needs attention. However, as the resident is no longer confined to a single department, it can be very challenging to find the resident in need. This search can take up a lot of the care givers' valuable time. To prevent this loss of time and also decrease the workload of the care givers, the panic button needs to be localized as this often is the only trackable device the residents always have with them. By localizing the panic button, care givers will now be able to find the resident faster when they are in need of help.

[1] `https://momomedical.com/`

1.2. Project Overview

For this project, two systems will be designed for Momo Medical. The first system, which this thesis will dive into, is an access control system which will allow or not allow a resident access to other parts of a nursing home. In addition, the system setup can be used for other goals, e.g. notifying a care giver when a resident is trying to exit his/her room at night.

The second system is a real-time localization system which will determine the location of a resident inside a nursing home. The care givers can either request the location of the resident or, whenever the panic button is pressed, get a message indicating that a resident needs help. In addition, this system could be set up in a way that it can also localize utilities such as lifting aids and keys. Furthermore, the system can also be used by care givers whenever they are in need of help themselves.

A schematic overview of the complete project is shown in Figure 1.1. In this figure, a clear division can be seen between the two different systems of the project, which will work together on the localization of a resident. These two parts, which are different systems, will not communicate directly but only via the Momo Medical database. For both systems, data is obtained from the database. For the localization system, this data will also contain the output of the access control system. The messages for the care givers will come from the server and will be visible in the Momo Medical app.

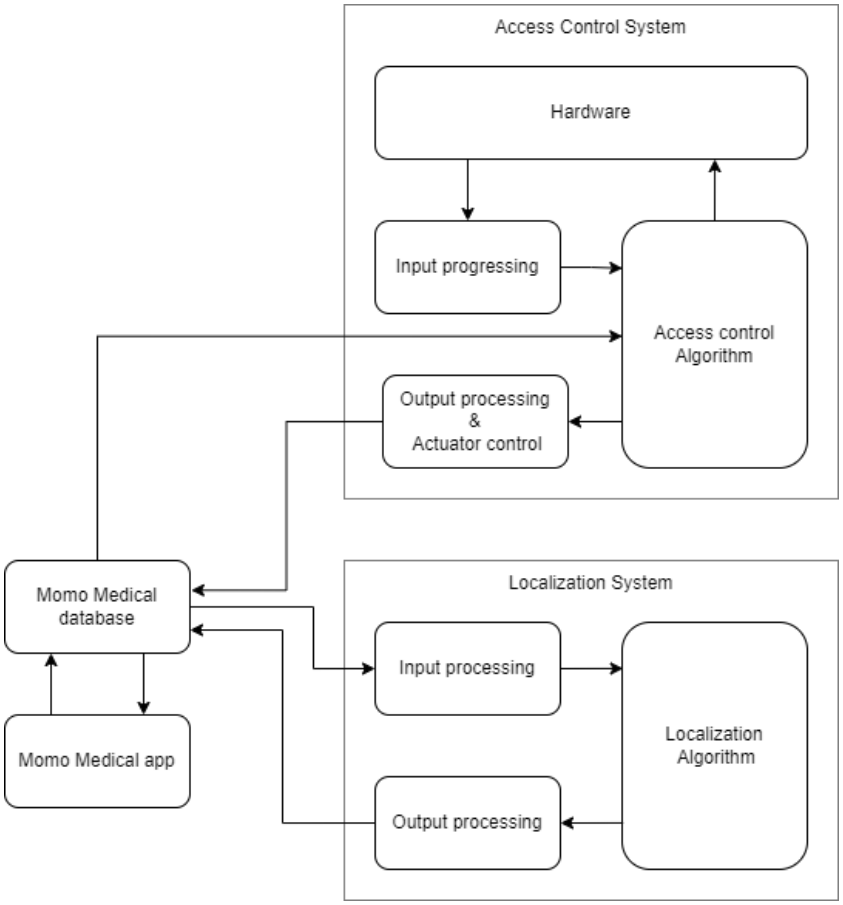


Figure 1.1: Schematic project overview.

1.3. Scoping and Bounding

The aim of this report is to clarify how the team of students from TU Delft solved the problem at hand. It will give a thorough description of the steps set in the project and in addition to that, give the result of the project with additional next steps. The main goal to clarify what steps have been taken to decide in which direction to go for different hardware components. The device in short will consist of two parts, one that will

the elderly will carry with themselves, and one that will be installed on the doorknob or somewhere close to it. What happens after the hardware has made an observation is not important at all and can be configured in any way in then future based on what Momo Medical and their clients wish to achieve.

1.4. State of the Art

The Internet of Things (IoT) is meant to connect anything and everything via sensors over the internet. One of the fundamental characteristics of the IoT is that objects are uniquely identifiable and are communicating with one and other all the time [1]. IoT is the concept fully embracing the product that is to be designed in this project, as the system designed with addition of other systems can form a whole infrastructure of communicating "things" that with the use of "internet" can communicate without limits when it comes to distance and in that way help companies offer service to clients wherever they are. The access control system (ACS) is simply said following the basic concepts of IoT.

There are only a few companies, e.g. Axess TMC[2] and identiv[3] who offer an ACS comparable to the system to be designed in this project. The technologies used by these companies include radio-frequency Identification (RFID), Bluetooth Low Energy (BLE), biometric readers, thermoscanners, infrared sensors, and many more. It is useful to look at the application of some of these technologies in general before moving on.

RFID is one of the technologies that is used often in the IoT. It has been used extensively in health care, and more specifically for tracking, identification & verification, sensing, interventions, and alerts & triggers [2]. Also on enterprise level, RFID is used often, namely for supply chain management, security, movement tracking, and object tracking [3].

BLE is also an commonly used technology in the IoT world. It can be used for indoor navigation, customer communication, ticket management, queue management, large-scale local notification [4].

1.5. Thesis Synopsis

Before starting with a technical design, it is important to know in technical detail what needs to be achieved, which is reason to start with the program of requirements in chapter 2. Afterwards, in chapter 3, comparisons are made between different options for each part concerning the hardware. After that, the necessary software components are discussed in chapter 4. Chapter 5 discusses the design of the prototype and the test methods. Finally, chapter 6 will contain the discussion and chapter 7 will elaborate on the final result and recommendations for future research.

[2] https://www.axesstmc.com/readers-access-control/
[3] https://www.identiv.com/industries/access-control-for-assisted-living-and-nursing-homes

2

Programme of Requirements

In order to properly assess the successfulness of the final product, it is imperative to determine a clear set of requirements regarding this final product. The final product is seen in two ways in this thesis, either the two systems designed for Momo Medical ("the whole system", in other words the combination of the access control system and the localisation system) or the access control system (ACS) which is thoroughly discussed in this thesis. For this reason, the first section in this chapter discusses the requirements that involves both systems designed and the section afterwards discusses the requirements that only involve the ACS. In addition, the scope limitations are discussed.

2.1. Requirement for the Whole System

For the two systems to work well together and with the existing Momo Medical system, conversations with Momo Medical have led to the following requirements which have to be satisfied by the whole system. These system requirements are shown in Table 2.1.

Table 2.1: Whole System Requirements

Requirement symbol	Requirement
SR1	The possible RF interference between the two systems must not affect the working of each system.
SR2	The system must output the location of the present panic buttons at least once a minute.
SR3	The system must be a plug and play system.
SR4	The system must not need any adjustments whilst in use.
SR5	The system must work in indoor environments.

2.2. Requirements for the Access Control System

For the ACS to work as Momo Medical has in mind, it is necessary to pin down the specific requirements for the ACS. The requirements can be divided into functional and non-functional requirements.

2.2.1. Functional Requirements

An overview of the functional requirements is given in Table 2.2.

Table 2.2: Functional Requirements of the ACS

Requirement symbol	Requirement
FR1	The access control system must detect all panic buttons when they are within 1 meter to the system.
FR2	The access control system must recognize a panic button within 100 milliseconds.
FR3	The access control system must connect to WiFi.
FR4	The access control system must have access to the current UNIX time in milliseconds and must send the timestamp 64 bits to the database.
FR5	The access control system must update its data by requesting data from the database every 15 minutes. This data contains the access rights that the access control system will check for.
FR6	The access control system must have a binary output that indicates whether a rule has been abiding or not, e.g. whether someone is allowed to enter or not.
FR7	The access control system must send updates to the database at least once every 3 seconds. These updates must at least contain the unique ID of the panic buttons which were observed by a system, which system has observed these panic buttons and at what time.
FR8	The access control system indicates by using LED(s) whether the system is working correctly or if errors occur.
FR9	The access control system must make sure that the battery life of the existing panic buttons will not be deteriorated.

The reasons for these requirements are as follows. For the ACS to be able to act on time, it needs to at least detect all panic buttons within one meter and scan for these panic buttons often so they are not missed. The reason is that residents who are walking will not wait for the system to act and will be able to continue their normal activities. So, the system needs to find the right balance between responding too soon and too late. Too soon and someone may not be exiting at all, e.g. when someone is just walking in his room and has a distance of 2 meters from the sensor. Too late and someone will enter a place they are not allowed to or the caregiver will be notified too late to help someone.

The WiFi connection is necessary for the system to directly communicate with the database and the updated internal time is to have the database accept the data. The local data should be updated with possible changes in the access rights as set by the caregivers so that the system can act in the desired way which can change throughout the day. The 15 minutes is a heuristic value. Updating the database on the other hand needs to happen as soon as possible so the localization system of the whole system can access the most updated available data and use that as its history.

For the moment, it is chosen for the ACS to have a binary output indicating whether a rule is followed or not, in view of the use case that Momo Medical has in mind.

2.2.2. Non-Functional Requirements

In addition to the functional requirements, there are non-functional requirements set that help to design a system operating as desired. These requirements can be seen in Table 2.3.

Table 2.3: Non-functional requirements

Requirement symbol	Requirement
NFR1	It must be possible to mount the access control system on or next to the door.
NFR2	The software for this system must be written in C/C++.
NFR3	The hardware must run from a 230V mains outlet.

Also the non-functional requirements need a short explanation. Requirement NFR1 is important considering

the aesthetics of the systems and the fact that it will be placed in a building where elderly people live. It should mainly not interfere with the daily activities of the residents. The software language is chosen in such a way as to make the system easy to integrate into the existing Momo Medical system.

2.3. Scope Limitations

Although the solution to be designed has a lot of freedom and can be based on different technologies, there are some limitations. One of the limitations is the cost of the system. The final system should have an acceptable price tag connected to it. What acceptable means here is not exactly indicated, but a cost of maximum 500 euros per ACS has been kept in mind throughout the project. It is also important for the system to use the existing infrastructure already available at Momo Medical for the database communication, which includes the software classes and functions written in C/C++ for this. A schematic overview of the existing infrastructure at which the ACS has to be added can be found in Figure 2.1.

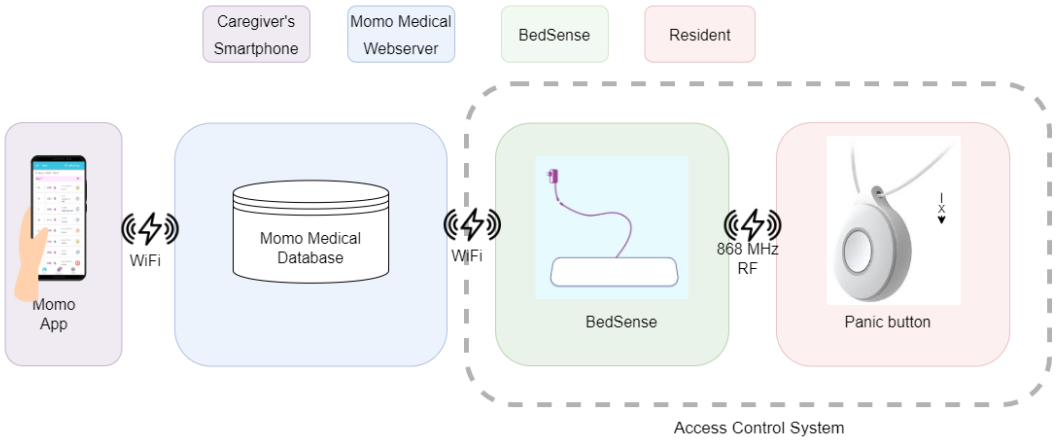


Figure 2.1: Existing Momo Medical infrastructure

3

Hardware Design

The goal of this chapter is to go through the process of the hardware design on which a software will run, so all requirements as stated in chapter 2 are satisfied. It is important to know which wireless communication method should be used to satisfy these requirements. After deciding on the communication method, which section 3.1 will dive into, a deep dive will be made into the hardware in section 3.2 and 3.3. The chapter will then finish with a theoretical discussion of the link budget in section 3.4.

3.1. Communication Methods

There are many options available which give the possibility to determine how close someone or something is to a certain point. Because of the need of using the setup indoors, as stated by requirement SR5, the use of some technologies is just not possible, e.g. GPS [5]. However, there are still many systems and technologies possible to use for the problem at hand, including but not limited to infrared (IR) detection [6], ultrasound [7], NFC [8], RFID [9], BLE [10], and ZigBee [11]. Each of these have their own advantages and disadvantages and the first step would be to look at those.

Infrared and Ultrasound

IR, although one of the most common technologies for localization, faces challenges due to interference from fluorescent light and cannot be used for identification of a person [5]. The advantage of IR technology, as well as ultrasound, in the system to be designed could be that the signal from one room in the residency will not trigger a reader in a different room. An additional advantage of ultrasound is that it is accurate within centimetres [12]. The disadvantage of ultrasound is that interference from reflected signals can take place and cause problems [5] and it needs expensive hardware.

Near Field Communication

Near Field Communication, or NFC is a very short range technology, having a range of approximately 20cm. It is based on similar technology principles as in RFID but is used not only for identification but also two-way communication [13, 14].

Different RF technologies, namely RFID, BLE, and ZigBee have the advantage of needing less hardware and being able to have a larger coverage area when needed. This can be a disadvantage at the same time, as RF signals can penetrate through walls and easily cause interference, both with similar systems in other locations within a building and with completely other types of systems, which happen to work at a similar frequency band [5]. There are also differences between the different RF technologies which are used a lot in the IoT field. For the application in the design at hand, it is important to have a technology meant for short range communication.

RFID

Radio Frequency Identification, or RFID, can be categorised in active RFID, semi-active RFID and passive RFID technologies. All three technologies use a reader and unique tags. These tags contain the information which can be used for identification. The active kind, although being able to store much more data and having

the greatest range, uses a battery-dependent tag and has a higher price and maintenance costs; semi-active RFID systems have the same disadvantages but have the advantage of not adding radio noise - a battery is used for processing data, but the received energy will be used for backscattering the signal to a reader-; and passive RFID are the smallest, cheapest and are not reliant on an internal power source for the tags. The drawbacks can be that they cannot store much data and the data is static. In addition, the readers for passive RFID tags are somewhat expensive [15]. The range of these systems are the highest for active RFID systems and the lowest for passive RFID systems, with the semi-active RFID in between those technologies [16, 13].

Bluetooth Low Energy

Bluetooth Low Energy, or BLE is designed for short-range, low bandwidth, and low latency applications with a low power consumption and setup time [13].

Zigbee

ZigBee is created to be a standard to suite high level, low cost communication protocols creating personal area networks. Specifically, it is used in applications that require a low data rate and longer battery life [13, 17, 18]. It typically operates in a range of 10 meters [17].

3.1.1. Conclusions

First of all, since the design must work at indoor environments as stated by requirement SR5, GPS shall not be used. Furthermore, it is clear that IR will not be the desired solution due to no being personally identifiable. The same reason makes ultrasound also not the ideal solution with the additional problem of interference that will easily take place in small rooms at residencies and high costs. Also, NFC is not the ideal solution as its range is so small as to not satisfy requirement FR1. Solutions that can be used are RFID, ZigBee and BLE. In order for requirement FR9 to be met, the passive RFID technology seems the ideal solution, as the tags do not have the need for an internal power source and thus, besides having a low maintenance cost, does not deteriorate the battery life of the panic button. The only drawback can be the high cost for the readers, which may be possible to be produced at a lower cost. For the same reason and because of the fact of being much simpler to apply in practice, RFID seems to be a better option than ZigBee and BLE, which need a network to be setup while RFID is peer-to-peer [13]. Once RFID has been chosen to be used, an extra requirement should be added to the functional requirements which is that the access control system must satisfy all the GEN2 standards [19].

The schematic toplevel overview in Figure 3.1 already gives an impression of the different parts of the final system.

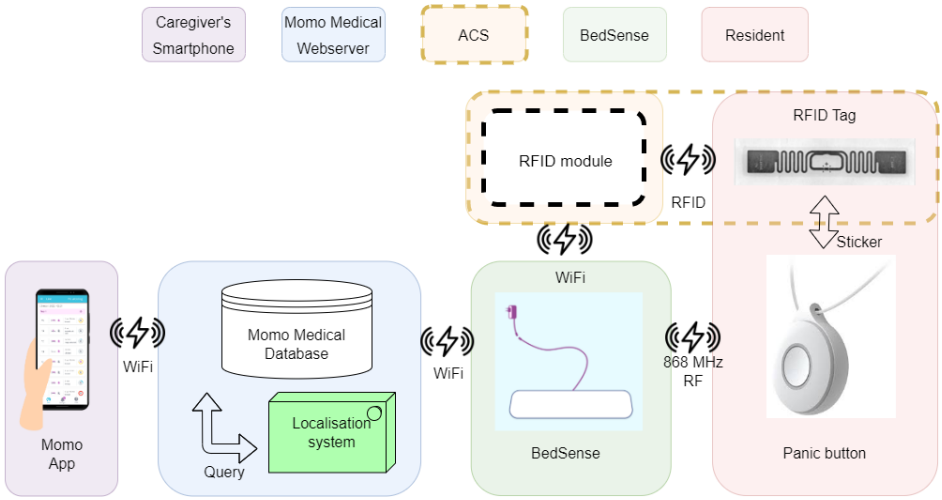


Figure 3.1: Toplevel overview, which includes a to be determined RFID system and the Localisation system.

3.2. Iterations of Designing the Hardware

Since concluded is that the Access Control System shall be based around a passive RFID solution, several RFID modules were tested. Since RFID is a technology that is present at multiple frequency ranges, development boards based around the high frequency HF) range (3-30 MHz) and the ultra high frequency (UHF) range (300-3000 MHz) were tested.

3.2.1. HF RFID: ST25R3911B

The hardware design started with a simple ready to use development board, namely ST25R3911B-Disco, based around a ST25R3911BB RFID reader IC and a STM32 microcontroller IC. The operating frequency is centered around the 13.56 MHz and 27.12 MHz band, which is in the HF band. The maximum output power is 31.5 dBm and supports multiple protocols, containing but not limited to ISO14443A, FeliCa and MIFARE classic, which in fact is a NFC protocol [1]. A few different corresponding tags were tested, the 20-064, 20-044 and 20-015 tags. They were all three based around the ST25TV02K IC. This setup was meant to help decide what frequency band to use. The ST25R3911B-Disco came with supporting software, which was able to automatically tune the antenna and show the amplitude of the received signal power (RSSI). The RSSI, however, was shown without units. Only the relative relationship based on the distance versus RSSI could be measured. A simple distance test was executed as follows:

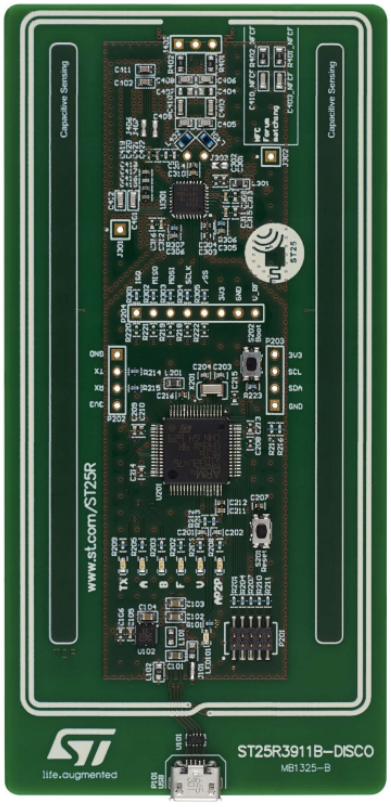


Figure 3.2: ST25R911B development board

The ST23R3911B reader was placed vertically and three different tags were placed vertically as well. The distance between the tags and the reader were varied at a constant rate and at multiples of 5.0 mm, the received signal strength indicator (RSSI) was measured. The results are shown in Figure 3.3, where a low measured RSSI value correspond to a high absolute RSSI value in Watts, and vice versa. It is clear that the absolute RSSI is logarithmically decaying as the distance between the reader and the tag increases. As can be seen, the HF tags were not able to be detected at a distance of 10 centimeters, with some not even detectable above 7 centimeters.

In conclusion: this HF RFID system would not suffice requirement FR1 and will not be used in the ACS.

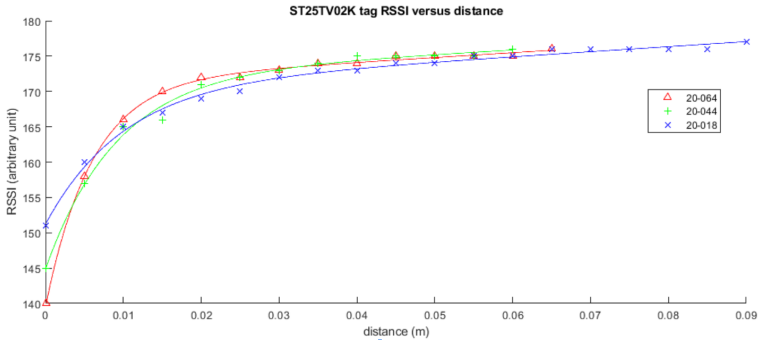


Figure 3.3: RSSI vs distance, using the ST23R3911B module and three tags. Horizontal axis is distance in meters and vertical axis is RSSI in an arbitrary linear unit.

[1] https://www.st.com/en/evaluation-tools/st25r3911b-disco.html

3.2.2. UHF RFID: ThingMagic Nano M6E

Based on the observation above, the next iteration of the hardware design moved on to using a RFID system functioning in the UHF band, which in The Netherlands means using the frequency band of 865-868 MHz [20, 21]. For this, a development board (SparkFun Simultaneous RFID Tag Reader[2]) was found to be an ideal solution as it had a lot of documentation.
This module consists out of the ThingMagic Nano M6E module mounted on a pcb. The basic specifications can be found in Table 3.1.
The Nano M6E module is an integrated module which contains at least an unknown microcontroller and a RF circuit. Unfortunately, ThingMagic does not provide the specific parts embedded in the Nano M6E module.

For the hardware design, an external microcontroller is used connected to the Nano M6E using a UART connection. Examples on how to connect this were found [3]. These examples use a C++ based library which ended up as the basis for the software design, which can be found in chapter 4.

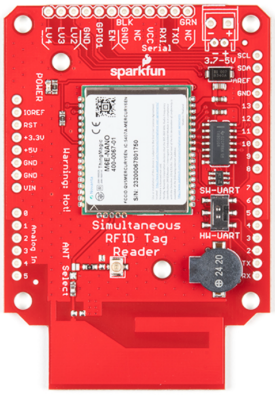


Figure 3.4: SparkFun Simultaneous Reader development board, with the Nano M6E mounted on it

Table 3.1: SparkFun Simultaneous Reader specifications [22]

Function	SparkFun Simultaneous Reader
Microcontroller	Nano M6E - STM32
Flash memory	4MB
SRAM	520 kB
GPIO	4
Operating Voltage	5V
Maximum RF power	27 dBm
Maximum RF input sensitivity	-65 dBm

Microcontroller Choice

Two microcontrollers were selected to be tested. The ATMega328p, placed on an Arduino Uno development board and an ESP32-WROVER-B, placed on a TTGO T8 V1.8 development board were selected, because they were easy to obtain and the members of the subgroup already had some experience with them. In Table 3.2, a comparison of their specifications are given. After a few tests, the ESP32 was chosen for further development, because it was the fact that it has an onboard RF module which supports a WiFi connection, necessary to connect the ACS to the Momo Medical Database as required by requirements FR3 and FR7. Furthermore, it also has increased flash memory, more SRAM and more GPIOs than the Arduino Uno, and it has a significant higher possible clock speed. Finally, the current Momo BedSense contains three integrated ESP32s, and software to communicate to the Momo Medical Database is already in use by the Momo Medical employees. These software files could be reused later on, and after finishing this project, the Momo Medical employees need less time to familiarize themselves with the ACS for further development.

Antenna Choice

The Nano M6E does include an on-board patch antenna. It also has an U.FL RF connector mounted, at which an external 50 Ω antenna can be mounted. The on-board antenna should be disabled by changing a solder joint, if chosen is to use an external antenna. Chosen is to use the vertical polarized GSM-34-900 patch antenna [23], for the fact that is was available to buy.

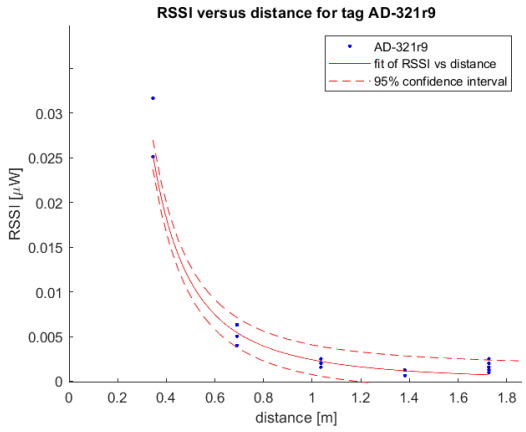
[2] https://www.sparkfun.com/products/14066
[3] https://github.com/sparkfun/SparkFun_Simultaneous_RFID_Tag_Reader_Library

Table 3.2: Comparison between the Arduino Nano and the ESP32-WROVER-B, based on the information by Arduino [24] and Espressif [25]

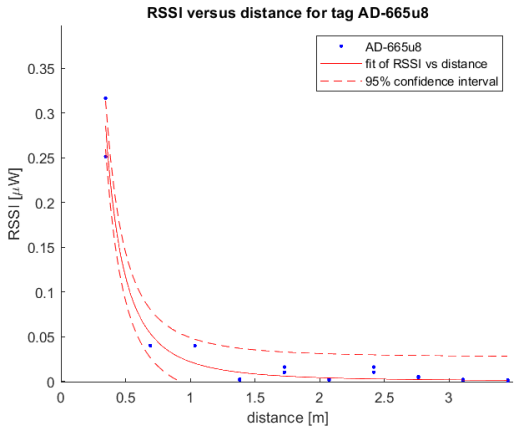
Function	ESP32-WROVER-B	Arduino Uno
Microcontroller	Xtensa 32-bit LX6	ATMega328P
Flash memory	4MB	32kB
SRAM	520 kB	2kB
Max clock speed	240 MHz	16MHz
Operating Voltage	3.3V DC	5V DC
Digital GPIO pins	36	14
Onboard Communication protocols	UART SPI I2C CANbus WiFi Bluetooth	UART SPI I2C

Range Test

For this RFID module, a range test was executed, using three different tags: the AD-172u7, AD-321r9 and AD-665u8. These tags consists out of an IC with a dipole antenna printed around it. The tests were executed by placing the antenna attached to the Nano M6E at one meter above the floor in a typical office environment. The tag was vertically placed to have a polarization match with the vertical polarized antenna. It was placed parallel to the transmitting antenna and one meter above the ground. It was then moved by distances of multiples of the wavelength of the RF wave. This in order to measure the incoming signal at the same phase to dismiss the effect of destructive or constructive interference. The received RSSI was measured for a period of 5 seconds and the mean of those measurements was taken to the next step. The results for the range test for the AD-321r9 tag are shown in Figure 3.5a and for tag AD-665u8 in Figure 3.5b. Tag AD-172u7 had a RSSI of -62 dBm at a distance of 40 centimeters and was not read at a distance of more than 50 centimeters, so no plot was made of its RSSI vs distance relationship.



(a) Relationship between RSSI and distance for the AD-321r9 tag. On the horizontal axis is the distance in [m] between the tag and the RX/TX antenna and on the vertical axis is the RSSI in [μW]. The fit corresponds to the formula RSSI = $22.5 \cdot r^{-2.39}$nW



(b) Relationship between RSSI and distance for the AD-665u8 tag. On the horizontal axis is the distance in [m] between the tag and the RX/TX antenna and on the vertical axis is the RSSI in [μW]. The fit corresponds to the formula RSSI = $2.36 \cdot r^{-2.27}$ nW.

Figure 3.5: RSSI versus distance measurements, for two UHF RFID Tags

Based on these results, chosen is to continue the design using the AD-665u8 tag, since it has an ever higher range when the Nano M6E was reading at full power than required by FR1. When a lower range is needed, decreasing the read power would be a solution.

Orientation of the Antenna and Tag

A small test has been executed to determine the influence of a difference in orientation of the tag with respect to the antenna.. The GSM-34-900 antenna was placed at the center of a room. Then the AD-665u8 tag was

placed vertically to align with the GSM-34-900 antenna at 8 equally spaced locations on a circle with a radius of 50 cm with the GSM-34-900 antenna at the center.
The AD-665u8 tag was always positioned such that it was perpendicular to the GSM-34-900 antenna. The RSSI was measured for 5 seconds and the mean was taken. This was repeated for a circle with a radius of 100 cm and 150 cm. The results are shown in Figure 3.6, where 0 degrees correspond to the position where the antenna was faced frontal. A visualisation of the test setup is shown in Figure B.1. At all measurement locations, the tag was found within the first 100 milliseconds, leading to a dataset of at least 50 datapoints for each measurement location.

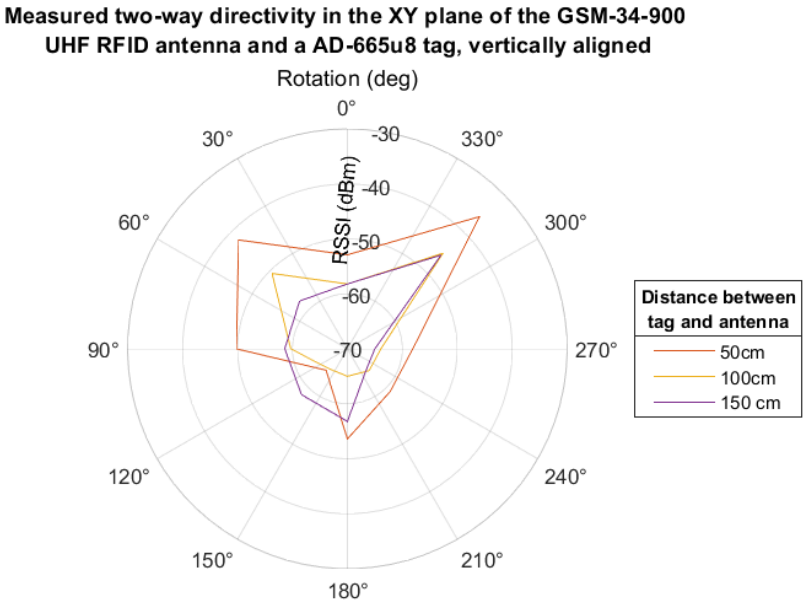


Figure 3.6: Angle of placement of tag with respect to GSM-34-900 antenna vs RSSI. 0° correspond to a frontal view with respect of the GSM-34-900 antenna. A visualisation of this setup can be found in Figure B.1.

Remarkably, the RSSI seems to be the highest around 315 °. 90° to the left, at 45°, the RSSI seems to be high as well. Another remarkable observation is that the RSSI around 135° seems to be smaller at a distance 50 cm than it is at a distance of 150 cm. Many factors do play a role in this experiment, and the reflections of the RF signal at the floor, walls and furniture do impact the overall performance. The most important result of this experiment is that the system finds a higher RSSI when the tag is present in semicircle in the XY plane in front of the GSM-34-900 antenna. Still, at the 1 meter line, the tag was found at each of the 8 points along the circle, indicating that requirement FR1 could be achieved. It can be concluded that, when a tag will be placed vertically, the receiver antenna is able to detect a tag at a larger distance when it is located in the front in comparison with a situation when a tag is located at the rear side of the GSM-34-900 antenna.

Actuator and Error LEDs
As stated by requirement FR6, the ACS must have a binary output indicating whether or not a tag that is observed by the ACS is allowed access. For this, a black box model was made. It has been decided that the ACS should be able to set one of the GPIO pins on the ESP32 high or low, according to the access rights. This has been indicated in Figure 3.7. In this same black box, two output LEDs were placed, which can be turned on or off indicating a critical software or hardware error, such as a loss of connection to the Nano M6E, no WiFi connection or some error state.

3.2.3. Conclusion on Hardware Design Choices
In conclusion, chosen is to design an access control system based on a Nano M6E UHF RFID module with the ESP32-WROVER-B as the microcontroller. The Nano M6E is attached to a GSM-34-900 antenna. The tag used at the moment is an AD-665u8. The ESP32 is responsible for a WiFi communication to the Momo Medical database. A black box model containing error LEDs and an actuator was added to complete the hardware design.The updated system overview is shown in Figure 3.7.

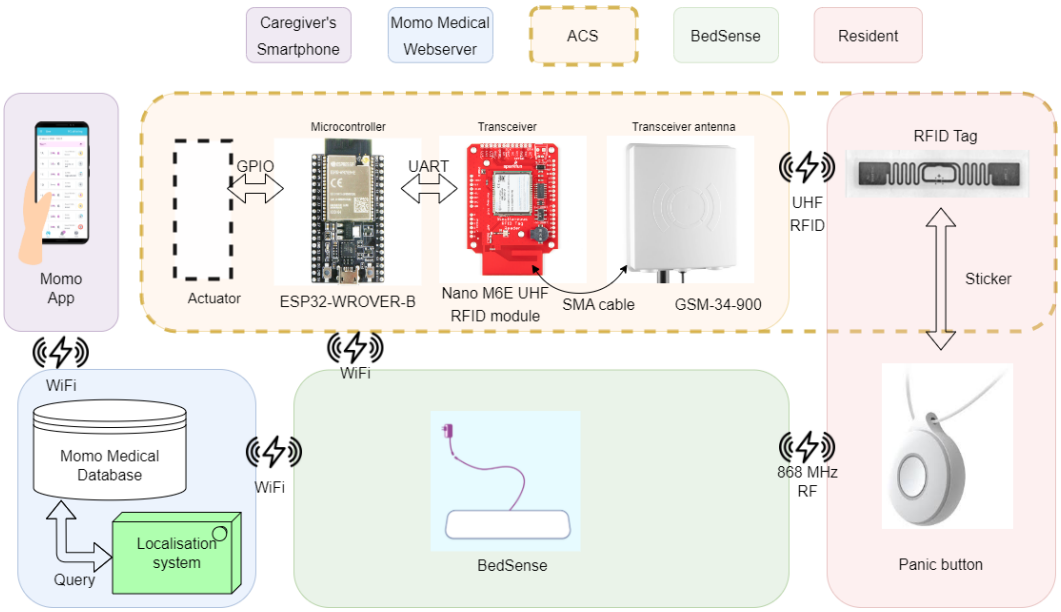


Figure 3.7: Access control system (ACS) as part of the Momo infrastructure system. The access control system (ACS) to be designed is shown in the yellow part of the figure, including the RFID tag, which is placed on the panic button of a resident. The connections between several parts of the Momo systems are shown as well. The place of the Localisation System [26] is shown as well.

3.3. Parts of the System

To gain a better understanding of the system, it is helpful to discuss the different parts of the system. The designed hardware system consists, as we can see in Figure 3.7, out of the tag, the transceiver antenna, the transceiver and the microcontroller. With the addition of the panic button, the database and the localisation system the whole system is formed. Let's take a look at each of these parts.

3.3.1. Transceiver

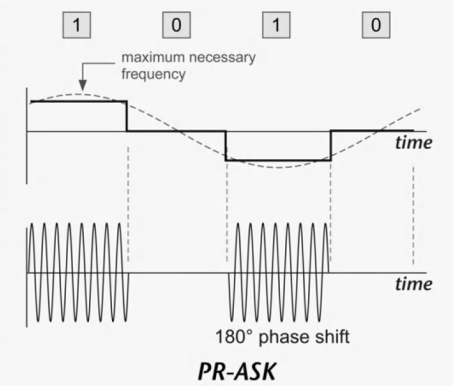
The transceiver consists of two parts, the transmitter and the receiver. The transmitter is meant to send out signals and this way charge the tags and send them data, and the receiver is meant to observe incoming data.

The transmitter part of the RFID has two phases, namely the uplink and the downlink phase. In the downlink phase, the transmitter provides enough power to start up the passive tag and sends the data it wants to the tag after modulation. In the uplink phase, it transmits an unmodulated signal that the tag can use to modulate its data and send back to the receiver [27]. This process is known as backscattering. When reading, the transmitter continuously transmits energy to its surroundings. In order for the receiver to be able to detect the weak signal backscattered by the tag at the same antenna used by the transmitter, a special component known as a circulator is used. This circulator only allows reflected signals to be presented at the receiver, which might otherwise be saturated by the transmitted signal [27]. When different antennae are used for the receiving and transmitting tasks, this circulator is not needed.

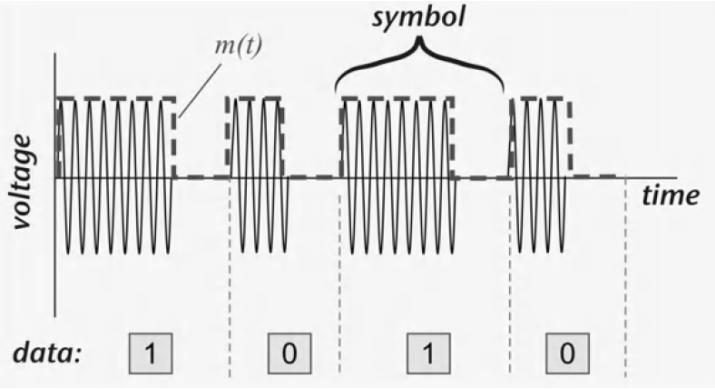
The transmitter of the RFID tag needs to satisfy different requirements for it to be able to be used in practice. One important thing to prevent is interference of the transmitted signal of different transceiver in a nursing home. To prevent this, it is important to minimize the width of the radiated signal. This can be achieved among other things by imposing a spectral mask or choosing a suitable modulation technique.

As indicated before, the Nano M6E is the transceiver used for the final design. This module makes it possible to read around 150 tags/second [22]. The module communicates to a host processor via a TTL logic level UART serial port[28]. Only three pins are required for serial communication: RX, TX and GND. The transceiver is to be controlled with a microcontroller and the provided API, which is explained in more detail in chapter 4. The Nano M6E is by default configured to the Gen2 protocol, in which is stated how the data from a transceiver to a tag should be modulated in order for the tag to harvest enough RF energy to process the data and send it

back to the transceiver. The Nano M6E uses phase-reversal amplitude shift keying (PR-ASK) as the modulation scheme [22], of which an example can be seen in Figure 3.8a. It offers an improved modulation efficiency compare to on-off keying (OOK) and other simple modulation techniques while still allowing simple demodulation by the tags, which is important due to limited energy received at the tag [27]. The downside of using a pure PR-ASK modulation scheme, is that no energy is sent when a string of zeros is sent to a tag. In that case, the tag might receive too little energy to operate correctly. For that reason, the Gen2 protocol allows PR-ASK only if it is using it in a pulse-interval encoding (PIE) variant, of which an example is shown in Figure 3.8b [19]. It makes sure that when a '0' has to be sent, the tag still receives power. The downside of this method is that the symbols '0' and '1' have a different length in time. It is unknown whether or not the Nano M6E uses a PR-ASK with or without PIE, but since the usage of PIE is required by the Gen2 protocol, it is assumed that the Nano M6E uses PR-ASK with PIE.



(a) Phase Reversal - Amplitude Shift Keying (PR-ASK). Image from [27]



(b) A Pulse-Interval Encoding (PIE) envelope used on a On-Off Keying (OOK) modulated signal. Image from [27]

Figure 3.8: Examples of PR-ASK and PIE. Note the absence of energy in the '0' state at the pure PR-ASK envelope. In the PIE case, there is energy transmitted when a '0' symbol is transmitted.

Other allowed modulation schemes by the Gen2 protocols are, besides the PR-ASK protocol, the double-sideband amplitude shift keying (DSB-ASK) and single sideband-amplitude shift keying (SSB-ASK) modulation schemes. Another way of decreasing interference by multiple Nano M6E transceivers is by using a higher Miller value for encoding that still achieves the tag read rates required for the application[22], which is adjustable via the API of the Nano M6E. The Gen2 Miller setting influences how data is encoded when sent from the tag to the reader. Higher Miller values send data at lower rates and are more noise resistant, increasing the module's sensitivity. Lower Miller values send data at higher rates, decreasing the sensitivity somewhat[22].

At the receiver part of the transceiver, an important factor is the receive sensitivity for the backscattered signal from the tags. Receive sensitivity is strongly influenced by the amount of interference caused by the reader's own transmit signal which can be reduced by reducing the transmit level, by the Gen2 Miller setting as in the previous paragraph, and by the region of operation, with the range of 865 to 868 MHz providing better sensitivity for the Nano M6E[22].

3.3.2. Tag

A UHF RFID tag consists out of an IC to which a dipole antenna is attached. Since dipole antennae mostly have a length of λ/n, where λ is the wavelength (approximately 34 centimeters in the case of a 868 MHz signal) and n is an even integer in the order of $1-12$ [27]. In the case of a $\lambda/2$ dipole, the length of the dipole is approximately 17 centimeters. For most UHF RFID tags, this is too large. A technique called meandering is used to fit the length of the used dipole on a smaller surface. How this exactly works is outside the scope of the project. An example of this meandering dipole is seen in Figure 3.9

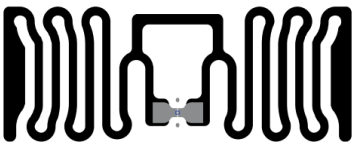


Figure 3.9: AD-321r6 tag with its meandering dipole antenna and data processing IC in the center. Image from [29]

3.3.3. Transceiver Antenna and Tag Antenna

For an antenna to be useful in an application, there are different metrics one needs to take into account. The three most important are the operating frequencies, radiation pattern and the polarization of the antenna. The radiation pattern indicating the the strength of the energy in the sent signal in different directions, the directivity indicating the orientation of the electric field of the radiated signal. In addition to these, the effective aperture, impedance, size, and cost are also important key figures [27].

There are many forms of antennae but a few simple ones are popular. For omnidirectional antennas, the dipole and its variants are chosen often, and when directional antennae are needed, the patch antenna is chosen [27]. Which one of these antennas is used, depends on how the antennae are placed in residencies. If for each door, only one transceiver antenna is used, an omnidirectional antenna is the right choice, as tags at both sides of the door can be read. If, on the other hand, two transceiver antennae will be used on each door, one on each side, the patch antenna is a wise choice. It should be mentioned that the latter one is a better choice for the use case at hand for the simple reason that, when a transceiver antenna is placed both at the inside and the outside of a room, the distinction can be made from whether someone is entering a place or exiting.

Tag antennas in the application at hand face different practical challenges. When it comes to the cost and size of these antennas, they should be as small as possible, while foremost offering the required range of 1 meter as set by requirement FR1. For the UHF RFID technology, the common choice of antenna type is a meandering dipole, as explained in subsection 3.3.2.

The choice for the polarization of the transceiver antenna should be dependent on the polarization of the tag antenna. When the tags have a single-dipole antenna, it can only be read when co-polarized with the transceiver antenna [27]. In this case, to make it possible to read the tag no matter the direction of it, a circularly polarized transceiver antenna is useful, as it has electric fields oriented in all directions. On the other hand, if dual-dipole tags are used, the transceiver antenna can have any polarization. In a dual dipole configuration, two dipole antennae are placed at a right angle to each other, such to have nonlinear horizontal and vertical polarization components. It should be mentioned that there is a trade-off between the use of circularly polarized antennas and large range [27, 30], especially for the tag antennas. Also important, linearly polarized tag antennas are cheaper [27].

For the other factors, the choice is simpler. The antenna bandwidth should contain the 868 MHz band. The size and cost of the antenna define also the bandwidth and the gain of the antenna, with larger and more expensive ones offering higher values for these two factors [27]. And of course, the tag antenna should be as small as possible in order to fit on the existing panic buttons.

3.3.4. Breakdown of the Nano M6E

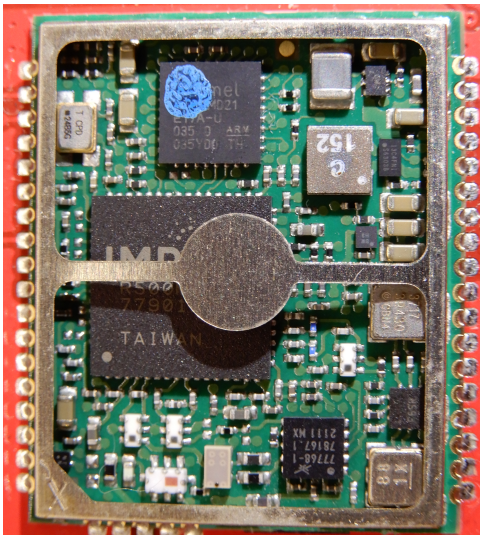
The Nano M6E is a standalone module, of which few specifications are available. To gain insight into the specifications, the module was physically opened and different subcircuits were analyzed. A full color image, as well as an image in which the different subcircuits discussed below are presented in Figure 3.10a and Figure 3.10b respectively. A high resolution image is presented in Figure B.2.

RFID Processing

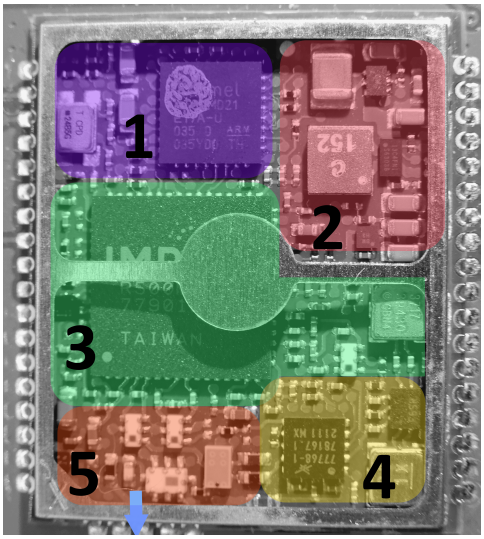
The largest IC on the Nano M6E is the Impinj IPJ-R500 Reader Chip. This IC is responsible for creating the to be transmitted TX signal and processing the incoming RX signal, both according to the ISO18000c/Gen2 standard. The operation frequency range equals 860-960 MHz. It can be configured to be used with only one antenna used for both TX and RX or by using multiple antennae for both TX and RX. The maximum transmitting power equals +20 dBm and the RX sensitivity equals -68 dBm with a Miller value of 4. Both the input and the output are matched to a 50 Ω load [31]. the IPJ-R500 act as both the bare receiver and transmitter of the This subcircuit is shown as **3** in Figure 3.10b.

Microcontroller

A microcontroller subcircuit was present. This subcircuit was based on the Atmel ATSAM21-E17A ARM-0 cortex microcontroller. This microcontroller converts the requests sent by the ESP32 over UART to a request



(a) Inner components of the Nano M6E in full color



(b) Nano M6E decomposed in parts. 1: microcontroller, 2: Power supply, 3: RFID processing, 4: Power amplifier, 5: Filtering and matching circuit. The RX and TX antenna is connected at the blue arrow down left.

Figure 3.10: Inner components of the Nano 6ME, both in full-color (a) and decomposed in the different subcircuits (b).

supported by the IPJ-R500 IC. To communicate between the ATSAM21-E17A and IPJ-R500 a SPI communication bus is used. The microcontroller has a maximum input voltage of 3.63V [32]. This subcircuit is shown as **1** in Figure 3.10b.

Power Amplifier
The TX signal as generated by the IPJ-R500 IC is lead to a SKY77768 power amplifier. This IC has matched input and output impedances of 50 Ω each. It has a maximum input power of 10 dBm and a maximum output power of 28.5 dBm. Remarkable, this power amplifier is designed to be used in the 880-915 MHz band. The Nano M6E is rated to be used below the bandwidth of the power amplifier, as can be seen in Table 3.1[28]. In the Netherlands, the allowed UHF band of 868 MHz [20] lies outside of the specifications of the power amplifier as well. Since the frequency characteristics of the output power are unknown, the effect of this mismatch in design can not certain be determined. The SKY77768 has a maximum input voltage of 4.2V [33]. This subcircuit is shown as **4** in Figure 3.10b.

RF Filter Circuit
Between the power amplifier and the antenna output a filter and matching circuit is present, as well as a circulator as discussed in subsection 3.3.1. This makes sense, since the Nano M6E only has one RF output, used by both the RX as the TX signals. The specific used components are unknown since no part numbers were present on ICs and the passive components were not measured.
This subcircuit is shown as **5** in Figure 3.10b.

Power Supply
An integrated power supply is present. The Nano M6E is rated for an input voltage of 3.7-5.5 V [28], but several of the ICs present in the other subcircuits need a maximum voltage lower than the maximum voltage the Nano M6E is rated for. The power supply subcircuit is based around a TPS63036 buck-boost converter IC, has a input range of 1.8V to 5.5V and can output a current of 1000 mA [34]. Based on the maximum input voltages of the other components, the TPS63036 is set to a maximum output voltage of 3.63 V. This subcircuit is shown as **2** in Figure 3.10b.

3.3.5. Conclusion and Recommendations
To conclude, all subcircuits discussed above are schematically shown in Figure 3.11. The maximum input

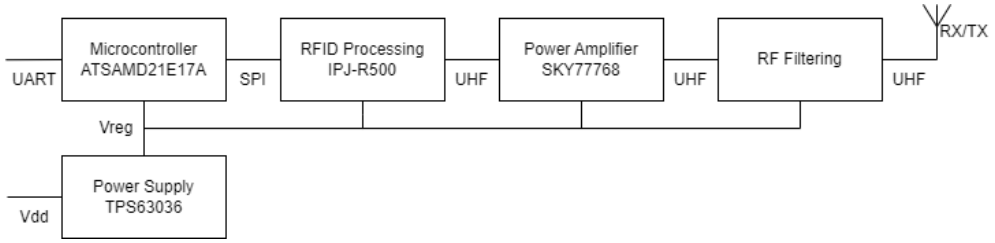


Figure 3.11: Block diagram of Nano M6E internal subcircuits

Table 3.3: Maximum input and output power of different ICs present in the Nano M6E and the Nano M6E module

	Max input power	Max output power
IMP-R500 [31]		+20 dBm
SKY77768 [33]	+10 dBm	+28.5 dBm
Nano M6E[28]		+27 dBm

and output power for the different ICs present in the Nano M6E are shown in Table 3.3. Remarkable is that the RFID processing IC can output a signal with a power higher than the maximum input power of the power amplifier IC. The RFID processing IC cannot be used to its full potential. However, since the power amplifier can output a signal with a power higher than the maximum output power of the RFID processing IC, this is not that much of a problem. The choice for the RFID module is remarkable as well, since the Nano M6E is still being produced, but the IMP-R500 is made End of Life by its manufacturer and will be phased out by the end of 2024 [35]. A redesign of the Nano M6E should be done in the near future, which include a new RFID processing IC.

3.4. Link Budget

In order to determine the minimum output power of the Nano M6E in order to detect tags at a distance of at least 1 meter as set by requirement FR1, it is important to define the link budget. The link budget is defined as a sum of all power gains and power losses. The link budget is split in the forward link budget (from the Nano M6E to the RFID tag) and the reverse link budget (from the tag to the Nano M6E). The used transmit antenna, the GSM-34-900, has a gain of 6 dBi [23].

3.4.1. Forward Link Budget

The forward link budget is split in three categories [27]:

- The power transmitted by the Nano M6E
- The power received by the tag
- The power needed to process the received signal at the tag.

TX Power

The power the Nano M6E can transmit is defined as the TX power P_{TX}. As found in the subcircuits of the Nano M6E in subsection 3.3.4, different subcircuits have different maximum input and output power limits, as summarized in Table 3.3 The maximum output power at the final stage of the Nano M6E is theoretically limited as the maximum output power of the power amplifier IC, which is, according to [33] equal to +28.5 dBm. The Nano M6E, however, is rated at a maximum output power of +27 dBm. According to local regulations [20], the maximum effective radiated power (ERP) in the UHF RFID band in The Netherlands is **+27 dBm**. This maximum can be achieved by the power amplifier.

Path Loss

The maximum received power at the tag is defined as the RX power P_{tag} and is dependant upon the transmitter antenna gain G_{TX}, the tag antenna gain G_{tag}, TX power and the path loss, as stated by the Friis equation and is defined as

$$P_{tag} = G_{TX} G_{tag} P_{TX} \frac{A_e}{4\pi r^2} \tag{3.1}$$

where A_e is the effective aperture of the tag antenna and r the distance between the tag and the Nano M6E. The effective aperture then can be defined as

$$A_e = \frac{\lambda^2}{4\pi} \tag{3.2}$$

where $\lambda = c/f$ is the wavelength of the transmitted signal, f the frequency of the transmitted signal and c the speed of light. In the case of a TX frequency of 868 MHz, this effective aperture leads to $\approx 95\text{cm}^2$.
When combining Equation 3.1 and Equation 3.2, we find a theoretical RX power at distance r as

$$P_{tag} = P_{TX} G_{TX} G_{tag} \frac{\lambda^2}{(4\pi r)^2}. \tag{3.3}$$

This RX power still is limited to a free-space model, where no reflections and backscatter components are present. The last term,

$$\frac{\lambda^2}{(4\pi r)^2} \tag{3.4}$$

is defined as the free space path loss.

Tag Power
Three different tags were tested, and their input power sensitivities are shown in Table 3.4. All three tags are designed around dipole antennae connected to a specific IC as stated in Table 3.4.

Table 3.4: Tags used for testing and their minimum input power and maximum antenna gain

Tag	IC	Minimum input power	Maximum antenna gain at f=865 MHz
AD-172u7	NXP UCODE 7	-21 dBm	1.5 dBi
AD-321r6	Impinj Monza R6	-22.1 dBm	3.5 dBi
AD-665u8	NXP UCODE 8	-22.9 dBm	16.5 dBi

Using Equation 3.3, the tag antenna gains and the minimum input power from Table 3.4, the maximum forward link distance with a TX power of 0.5W can be calculated as

$$R_{f,max} = \left(\frac{\lambda}{4\pi}\right)\sqrt{\frac{P_{TX} G_{TX} G_{tag}}{P_{tag,min}}} \tag{3.5}$$

and the results for the three tested tags can be found in Table 3.5. Remarkable is that the AD-665u8 tag has a much larger maximum forward link distance than the AD-172u7 tag.

Table 3.5: Theoretical maximum forward link distance of three tags in free space

Tag	Theoretical maximum forward link distance (m)
AD-172u7	16.35
AD-321r6	23.37
AD-665u8	114.45

The maximum allowed power loss at that distance follows from the maximum power loss

$$L_{TX,max}[\text{dB}] = P_{TX}[\text{dB}] + P_{tag}[\text{dB}] \tag{3.6}$$

and is calculated for the three tested tags. The results are shown in Table 3.6.

Table 3.6: Maximum power loss in the forward link

Tag	Maximum power loss [dB]
AD-172u7	53
AD-321r9	54.1
AD-665u8	54.9

3.4.2. Return Link Budget

The return link budget can be split in similar categories as the forward link budget:

- The power backscattered by the tag.
- The power received by the Nano M6E.
- Minimum power to be received by the Nano M6E to still progress the signal.

Tag Power

As discussed in subsection 3.4.1, the minimum power needed to process the signal sent from the Nano M6E to the different tags are shown in Table 3.4. A good estimate for the amount of power needed to process the data in the tag is around 33% of the received power, a loss of approximately -5 dB [27]. This backscatter power $P_{tag,b}$ then is defined as

$$P_{tag,b} = 0.33 P_{tag}. \tag{3.7}$$

The signal received at the Nano M6E as backscattered by the tag is defined as P_{RX}, follows a similar Friis equation as used in the forward link budget and equals

$$P_{RX} = G_{RX} G_{tag} P_{tag,b} \frac{\lambda^2}{(4\pi r)^2} \tag{3.8}$$

for distances in the far field. Combining Equation 3.8 with Equation 3.7 and Equation 3.3, the received power at the Nano M6E as function of distance is found to be

$$P_{RX} = 0.33 \cdot G_{RX}^2 G_{tag}^2 P_{TX} \frac{\lambda^4}{(4\pi r)^4}. \tag{3.9}$$

From Equation 3.9, it is clear that the relationship between the received power by the Nano M6E and the distance between the tag and the Nano M6e in free space is given as

$$P_{RX} \propto r^{-4}. \tag{3.10}$$

Since the antenna attached to the Nano M6E has one antenna used for both transmitting and receiving, the RX and TX antenna gain are equal. The sensitivity $P_{RX,min}$ of the Nano M6E is -65 dBm [28], and the theoretical maximum distance at which a tag still can be read then is defined as

$$P_{RX,min} = \frac{\lambda}{4\pi} \sqrt[4]{0.33 \frac{G_{TX}^2 G_{tag}^2 P_{TX}}{P_{RX,min}}}. \tag{3.11}$$

In the free-space case and TX power of 0.5W, the theoretical maximum distances are calculated using Equation 3.11 and Table 3.4 and the results are shown in Table 3.7.

Table 3.7: Theoretical maximum return link distance of three tags in free space

Tag	Theoretical maximum return link distance (m)
AD-172u7	9.88
AD-321r9	12.43
AD-665u8	55.54

The maximum return link distances are smaller than the maximum forward link distances. When the tags are placed at the maximum distances as shown in Table 3.5, the return power equals

$$P_{RX,r_{max}}[\text{dB}] = 2 \cdot L_{return,max}[\text{dB}] + P_{tag,b}[\text{dB}] \tag{3.12}$$

where

$$L_{return,max} = L_{TX,max} \tag{3.13}$$

is the maximum path loss. The results are shown in Table 3.8.

Table 3.8: Power received at the receiver when a tag is placed at the maximum forward link distance for three tags

Tag	Power received at the receiver when a tag is placed at the maximum forward link distance [dBm]
AD-172u7	-79
AD-321r9	-80.1
AD-665u8	-80.9

The lower maximum return link distances as seen in Table 3.7 can be maximized by using a receiver with an input sensitivity of at least the power received at the receiver when a tag is placed at the maximum forward link distance as seen in Table 3.8. The RFID processing IC present on the Nano M6E, the IMP-R500 IC, with a minimum input sensitivity of -65 dBm, cannot achieve this. The manufacturer of this IC, however, does produce similar RFID processing ICs, such as the E510, E710 and the E910 ICs, which, respectively, have an input sensitivity of -82 dBm, -88 dB, and -94 dBm [36]. They will all achieve the task of reading the tag at maximum forward link distance.

3.4.3. Link Budget in Indoor Situations

In indoor situations, however, the RSSI will behave as it does in Equation 3.10. The received attenuation of the forward link RSSI follows an indoor propagation model, such as Egli's model where the propagation loss L is given as

$$L = 20\log_{10}(f) + 40\log_{10}(r) - 20\log_{10}(h_t) - 10\log_{10}(h_r) - 43.7 \tag{3.14}$$

where f is the frequency of the transmitted signal in Hertz, r the distance between the transceiver and the tag in meter, h_t the height of the transceiver antenna in meter, h_r the height of the tag in meter and 43.7 a propagation loss constant [37]. This model is valid in indoor situations and can be simplified to a far-field model of the RSSI at the tag, given as follows, where all units units are in their corresponding decibel version:

$$L \approx PL_{r_0} - 10n \cdot \log_{10}\left(\frac{r}{r_0}\right) + \chi_\sigma \tag{3.15}$$

where PL_{r_0} is the path loss in dB at a reference location r_0 in the far-field and χ_σ is Gaussian distributed with a mean M in dBm and variance σ^2 dBm^2 [38]. This random variable will be nonzero when no fading is present. Fading is the phenomenon where the attenuation of the signal is not only a function of distance, but also when other parameters are involved, such as reflections and multipath propagation.When measured in absolute units, χ_σ is log-normal distributed. The factor n is defined as the propagation loss factor, which is dependant on the physical conditions. For forward link, this factor equals 2 for free space. For indoor environment, this value differs and has to be empirically defined, since it is dependant on all objects in its neighbourhood which can reflect signals between the transceiver and the tag.

For a return link budget, expected is that the path loss exponent n is twice the value of a forward link path loss exponent. Typical return link path loss exponents for office environments can be in the order of 1.8-2.2 for vertical polarized transceiver antennae and vertical oriented tags, depending on the height of the transceiver antenna [39].

The Influence of Reflections on the Link Budget

The added random variable χ_σ is the result of shadow fading, or the shadowing effect. This phenomenon plays an important role in indoor signal propagation and is a function of reflections, as given by equation Equation 3.16, where the forward link path loss is defined as

$$L = \left(\frac{\lambda}{4\pi r}\right)^n \left|1 + \sum_{i=1}^{N} \Gamma_i \frac{r}{r_i} e^{-jk(r_i - r)}\right|^n \tag{3.16}$$

where r_i is the distance of the reflected ray in meters, Γ_i the reflection coefficient, k the wavenumber and N the number of reflected rays [38]. The amount of reflected rays will in practice be infinite, and this equation is only useful for modelling. In Figure 3.12, this equation is modelled, and 5 reflections with distances $r_i \in \{1.1, 1.2.1.3, 1.4, 1.5\}$ meter are added, all with perfect reflection ($\Gamma_i = 1$). More assumptions taken into account here were the frequency (868 MHz) and an antenna and tag height of 1 meter. As can be seen, the

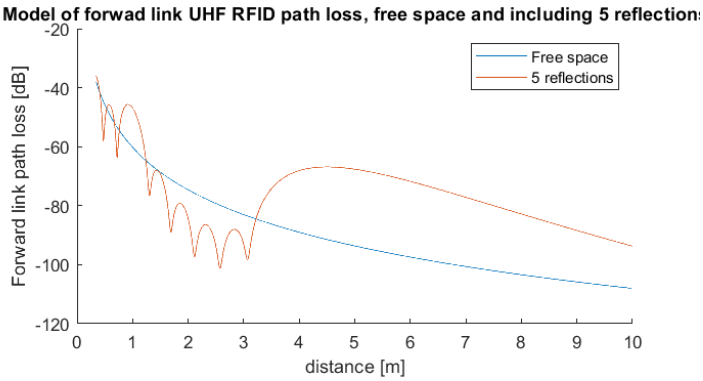


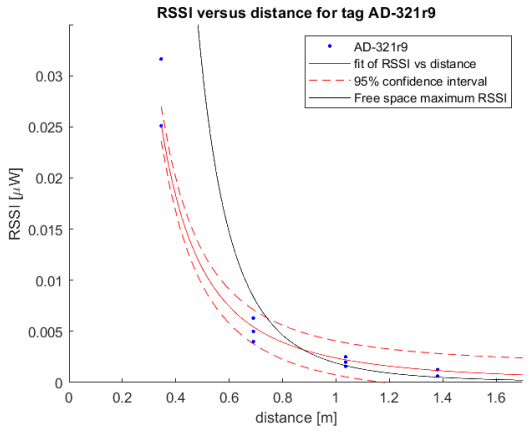
Figure 3.12: Model of a UHF RFID path loss model, with 5 reflections and the free space model as defined in Equation 3.4

path loss is not decreasing at a constant rate. At different distances, the path loss might be equal. This phenomenon can also explain the fact that during the orientation experiment where the results are shown in Figure 3.6, the signal at 135° had a higher RSSI at a distance of 150 cm than at 50 cm. From this, it can be concluded that it is not possible to derive the distance at which the tag sent a signal and thus trying to minimize the read distance to 1 meters is a challenge.

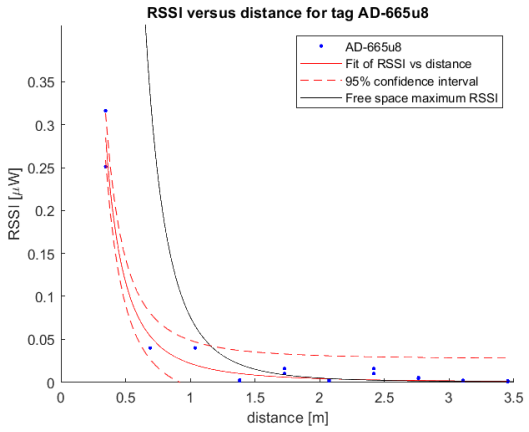
It should be stressed that the results in Figure 3.12 is only based on a model, in real indoor environments, this path loss is even more unexpected. Also, this model is based on the assumption that the tag is worn vertically, facing the transceiver antenna. When the orientation of the tag changes, the path loss changes as well. For now, it will be assumed that the ACS will only detect presence of tags. This information still is quite useful for the localization system.

3.4.4. Validation of the Link Budget Theory

In subsection 3.2.2, a range test was executed. The theoretical relationship of the RSSI and distance as defined in Equation 3.11 was added to the results of this test and shown in Figure 3.13a and Figure 3.13b for tags AD-321r9 and AD-665u8 respectively. For tag AD-665u8, the theoretical maximum falls within the 95% confidence range of the fit over the measurements at a distance of 1.4 meter and more. At a distance lower than 1.4 meter, the measured RSSI is much lower than the theoretical maximum. This can be explained by the fact that the theoretical model is only valid in the far-field. In the near-field, the RSSI has strange behaviour, due to the fact that in the near field the reactive field is stronger than the electromagnetic field.



(a) Relationship between RSSI and distance for the AD-321r9 tag. On the horizontal axis is the distance in [m] between the tag and the RX/TX antenna and on the vertical axis is the RSSI in [μW]. The fit corresponds to the formula RSSI = $2.36 \cdot r^{-2.27}$ μW. Also added is the free-space RSSI as calculated by Equation 3.7 and the maximum antenna gain in Table 3.4



(b) Relationship between RSSI and distance for the AD-665u8 tag. On the horizontal axis is the distance in [m] between the tag and the RX/TX antenna and on the vertical axis is the RSSI in [μW]. The fit corresponds to the formula RSSI = $22.5 \cdot r^{-2.39}$ μW. Also added is the free-space RSSI as calculated by Equation 3.7 and the maximum antenna gain in Table 3.4

Figure 3.13: RSSI versus distance measurements, for two UHF RFID Tags

Also remarkable are the resulting fits

$$\text{RSSI} = 22.5 \cdot r^{-2.39} \mu W \tag{3.17}$$

for the AD-665u8 tag and

$$\text{RSSI} = 2.36 \cdot r^{-2.27} \mu W, \tag{3.18}$$

for the AD-321r9 tag, which are in the form of

$$\text{RSSI} = a \cdot r^{-b}, \tag{3.19}$$

and have an exponent which differs quite a lot from the free-space model where the exponent equals -4 (Equation 3.10). The main explanation is that the fits were calculated on the data which included measurements present in the near field, where Equation 3.11 does not hold. The fact that in indoor situations reflections from floors, walls, furniture and people do influence the signal travelling to and from the tag quite a lot has to be taken into account as well. The difference in path loss coefficient is just a bit higher than the expected exponent in the range of 1.8-2.2, as explained in subsection 3.4.3.

4

Software Design

4.1. Hardware Overview

The main microcontroller of the ACS is, as explained in subsection 3.2.2, an ESP32-WROVER2-B. This module is responsible for communicating between the RSSI reader, where tag information will be read, and the Momo Medical database (referred to as "the database"), where the tag information will be stored for use of localizing the tag by the Localization subgroup. In addition, it is the decision maker for the ACS, deciding whether some specific rule is fulfilled or not. In this chapter, the how of these things will be discussed. So, how will the ESP32 carry out these tasks? The simple answer is by running the software written for it. Let's dive into the technical discussion of the software.

4.2. Programming Language and the ESP_IDF Framework

The programming language used for programming the ESP32 is C/C++. The software program has been developed in Visual Studio Code and makes use of the ESP_IDF framework. This framework contains functions used to access the internal functionalities of the ESP32. These build-in functions are written in C, as are the functions that are responsible for the UART connection to the Nano M6E and extracting data from it. The functions used for processing the data and setting up the communication to the database are written in C++. This choice is motivated by the fact that it was easier to store data in objects that are supported in C++ and not available in C.

4.3. Toplevel Overview

The software package for the ACS exists out of several files where the ACS algorithms are described. In Figure 4.1, a top-level overview of the C/C++ files is shown. The following subsections explain the main functionality of these files.

Main.cpp
In this C++ file, the main algorithm is running. It does not include function definitions, but calls functions from other C/C++ files. The main loop starts by configuring the system setting and afterwards start the reading process that will run until a critical error occurs, if the ESP32 was reset or if its power was shut off.

NanoM6EHelperFunctions.c
This C file is responsible for creating requests for and read responses from the Nano M6E. It contains functions to initialize its settings and can ask it to return at which value settings are set. It can request for tag data and decompose it to several metadata parameters. The request and response structures are explained in subsection 4.4.1 and 4.4.2 respectively. All high-level functions that are defined in this file are shown in appendix A.2.

WifiBap.c
This file contains the function that can be called to setup the WiFi on the ESP32 and also offers event handlers for using the WiFi.

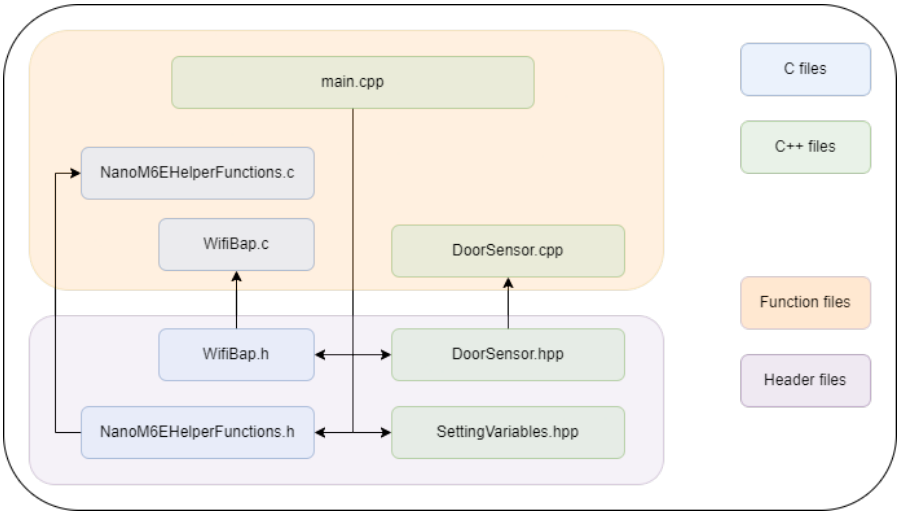


Figure 4.1: Toplevel overview of the C/C++ files. Excluding standard library files

DoorSensor.cpp
This is the main file that has been developed for this project. It contains one class that represents the door sensor as seen from the ESP32's point of view. It will create an object instance of this class, which setup the ESP32's communication methods for WiFi, UART, the GPIOs that needed for LEDs, and the time. Then, it will get the most up to date data from the databases. The how is explained in section 4.8.

This file, in addition, defines what happens in a read cycle, which it divides in a read part and a process part. The read part, specifically, looks whether any data has been received from tags and in that case gets the data. Afterwards, the process part goes through the tag data received and make decisions based on the tag data that comes in. For this, the file has defined functions for processing tag data and also updating the database.

4.4. The UART Connection

The ESP32 and the Nano M6E communicate to each other using an UART connection. How the UART connection is established will be explained in subsection 4.8.1. The ESP32 constructs instruction Messages and the Nano M6E constructs response messages which both will be send over the UART connection. In the following subsections, those messages are broken down into pieces to see how they are constructed. Every instruction and response have at least a basic structure, as can be seen in Table 4.1. The maximum length of a message is limited to 256 bytes.

Table 4.1: Basic message structure[40]

Name	Header	Length	Opcode	Data	CRC
Size (bytes)	1	1	1	0-250	2

A message contains at least the following components:

Header
The header always equals `0xFF` and this byte implies the beginning of a message.

Length
The length tells an user out of how many bytes the data bytes exist. This is the length of the entire message - 5 bytes (1 byte for the header, 1 byte for the length, 1 byte for the opcode and 2 bytes for the CRC).

Opcode
The opcode tells what defines the function of the request or response. A list of all possible opcodes is given in `serial_reader_imp.h`[41]. A selection of these opcodes are implemented in C functions which can be called from the C/C++ files above. These functions are declared in `NanoM6EHelperFunctions.c` and the

high-level functions can be found in section A.2. When the ESP32 sends an request with opcode `x`, the Nano M6E responds with a response using the same opcode. The Nano M6E also can send one-way messages on its own to the ESP32. If that is the case, the opcode always will be `0x22` and these messages can have three different structures, as will be explained in subsection 4.4.2.

CRC
At the end of each message, a Cyclic Redundancy Check (CRC) is executed and two bytes are inserted at the end of the message. The CRC is executed on the entire message, excluding the header byte. These CRC bytes will be checked by the receiver and if the CRC locally calculated with the received message do not correspond to the CRC bytes received, it is assumed that the received data is corrupted and the receiver will discard the corrupted instruction. For now, no error correction will be applied.

The CRC bytes are generated by a simple algorithm that can be found in subsection A.2.27. It deviates from the well-known CRC-16 algorithm (`serial_reader_imp.h`, [41]).

4.4.1. Request Syntax Introduction
To send a request from the ESP32 to the Nano M6E, the request should be in the basic structure as can be seen in Table 4.1. As an example, a request is shown in Table 4.2, which is sent to the Nano M6E by calling the `getReadpower()` function. Its functionality is explained in subsection A.2.3. The opcode used in this example, `0x62`, is extracted from `serial_reader_imp.h` [41].

Table 4.2: Example of a `getReadpower()` command

Name	Header	Length	Opcode	Data	CRC
Data	0xFF	0x01	0x62	0x00	0xBEBD

Depending on the instruction, the data element can consist out 0 to 250 bytes.

4.4.2. Response or One-Way Message Syntax
When the Nano M6E sends data to the ESP32, this message can either be (a) a response to a request or (b) one out the following three one-way messages:

- The Nano M6E sends a keep-alive message (once every 1 sec).
- The Nano M6E sends a temperature log message (once every 0.25 sec).
- The Nano M6E sends a Tag Found response.

Such a response or one-way message can be broken down in more than the four basic elements as shown in Table 4.1. In this section, the possible response or one-way message structures and their specific elements are explained. The structure of a response or one-way message is shown in Table 4.3.

Table 4.3: Response or One-Way Message Structure[40]

Name	Header	Length	Opcode	Status	Data	CRC
Size (bytes)	1	1	1	2	0-248	2

As can be seen, the data part is split: two bytes are taken and form the status bytes, as will explained later. The possible length of the new data part shrinks with 2 to a maximum of 248 bytes, to still accommodate the maximum message length of 256 bytes.

First, the opcode, status, data and length byte contents will be explained. Afterwards, the four different message structures will be explained.

Opcode
In the case of a response to an request, the opcode of the request is used in this response. In the case of one of the other messages, this opcode always will equal `0x22`.

Status
When the Nano M6E has successfully executed the instruction sent by the ESP32, the status bytes will equal `0x0000`. It an error occurred, these bytes will not be equal to `0x0000`. A simple check on this is implemented and explained in section 4.5.

Data
In the case of a 'set' request - a request where the Nano M6E is requested to change a setting - the data bytes will be empty. As consequence, the length byte will equal `0x00`.
When a 'get' request has been sent, the data bytes will equal the specific data that was requested. The length byte will indicate how many data bytes were returned.

Length
The length byte still indicates the number of bytes the data part consists of. In a response or

4.4.3. Keep-alive Message Syntax
This message is automatically sent from the Nano M6E to tell the ESP32 that it still works. It is sent approximately once every second. The response is decomposed in Table 4.4

Table 4.4: Keep-alive Message (KaM) Decomposition[40]

Size (bytes	Name	Description
1	Header	Header, always `0xFF`.
1	Length	Length of message, excluding Header, Length, opcode and CRC length. Since a KaR does not contain any data, this byte equals 0x00.
1	Opcode	`0x22`
2	Status	`0x0400`
2	CRC	CRC executed on entire message, excluding header.

4.4.4. Temperature Log Message Syntax
Once every 0.25s a Temperature Log Message (TLM) is sent to the ESP32. It tells at what temperature the Nano M6E was at time it was sent and has a resolution of 1°C. The response is decomposed in Table 4.5

Table 4.5: Temperature Log Message (TLM) decomposition[40]

Size (bytes)	Name	Description
1	Header	Header, always `0xFF`.
1	Length	Length of message excluding Header, Length, opcode and CRC, always `0x0A`.
1	Opcode	`0x22`
2	Status	`0x0000`
9	RFU	Reserved for future use: `0x00031B028200820001`
1	Temperature	Temperature in °C
2	CRC	CRC executed on entire message, excluding header.

RFU
The reserved for future use (RFU) bytes are set by the manufacturer of the Nano M6E and cannot be altered. They have to be ignored in the temperature processing algorithm.

4.4.5. Tag Found Message Syntax
When the Nano M6E is requested to read for nearby RSSI tags, it will send a Tag Found Message (TFM) with some metadata when a tag is detected. A description on the parts of which the response exist is shown in Table 4.6. For some parts however, a more extensive description is given in following subsections.

Table 4.6: Tag Found Message (TFM) Decomposition [40]

Size (bytes)	Name	Description
1	Header	Header, always `0xFF`.
1	Length	Length of message excluding Header, Length, MSG CRC.
1	Opcode	`0x22` for one-way message from Nano M6E.
7	RFU_1	Reserved for future use.
1	RSSI	RSSI in dBm, 2's complement.
1	Antenna ID	First nibble is TX. antenna, second nibble is RX antenna.
3	Frequency	Frequency in kHz.
4	Timestamp	Time in ms since last keep-alive message.
2	Phase	Phase at which the tag signal was read (0-180 deg).
1	Protocol ID	Protocol ID as set in the `readerSetup()` function.
2	#data bytes	Length of data stored on tag in bits, saved in M
8M.	Data	Data stored on tag.
1	RFU_2	Reserved for future. use. Will always be `0x0F`.
2	#EPC bits	Length of EPC. ID , PC and EPC CRC in bits. Saved in N.
2	PC	Tag Protocol Control.
8N.	EPC	EPC ID.
2	EPC CRC	CRC executed on EPC only.
2	MSG CRC	CRC executed on entire message, excluding header.

RFU
Before moving on, it is good to mention that in a Tag Found Response, there are two parts that are reserved for future use (RFU) and those bytes are independent of the message. RFU_1 corresponds to the message `0x10001B01FF0101` and RFU_2 corresponds to `0x01`. These bytes are set in the Nano M6E by design and cannot be altered.

EPC ID
The EPC ID is the ID of the tag. This ID is used to identify the specific tag. It can be changed with the function `writeEPC(EPC)` and be read with the function `readEPC()`. The maximum length is 496 bits/62 bytes [19]. UHF RSSI tags come with an worldwide unique 96 bit/12 byte EPC ID when bought.

PC
The Protocol Control (PC) is another checksum that is executed on the EPC. The PC bytes will be equal to `0xZZ30`, where `0xZZ` equals $\left\lceil 4 \cdot \text{size(EPC ID)} \right\rceil$. The PC bytes are automatically generated by the Nano M6E and ignored in the software design.

4.5. Error Solving Algorithm

There are several causes for errors. The Nano M6E has the ability to send the possible error status to the ESP32 using the two status bytes as discussed in subsection 4.4.2. An algorithm to solve the errors which during development were seen regularly was designed. When an unknown error message was sent in the status bytes, the ESP32 resets the Nano M6E and itself. The error codes for which error solving algorithms were constructed are shown in Table 4.7. A complete overview of all possible error messages and their cause can be found in [22].

Table 4.7: Error codes for which an error solve algorithm has been designed

Status bytes	Cause	Solution
0x0000	No error	Continue
0x0101	Opcode not supported	Reset the Nano M6E and rerun the Nano M6E setup functions
0x0400	An operation, such as writing or reading a tag failed	Reset the Nano M6E and restart the Nano M6E setup functions
0x0401	A command involving reading or writing failed because of missing protocol/region/power setup parameters	Reset the Nano M6E and rerun the Nano M6E setup functions
0x0504	Temperature above rated temperature	Stop the reading of tags at the Nano M6E for at least 5 minutes
other	n/a	Reset the Nano M6E and restart the Nano M6E setup functions

4.6. Instruction Send Algorithm

Now the message structure is clear, the algorithm to send and receive data can be explained. The basis for the Nano M6E algorithm to send instructions was found[1] and heavily modified. A Nassi-Schneidermann diagram of this algorithm can be seen in Figure 4.2. In the following sections, the individual components will be explained.

4.6.1. Request Message Construction

The requests will be build according to the structure that was explained in Table 4.1. The data part of specific requests will be constructed as needed by specific requests and will be explained in section A.2.

4.6.2. Send Request

The built instruction will be placed in the outgoing buffer. When the buffer was not sent within 100 milliseconds, the ESP32 will clear the outgoing buffer and send the instruction again. The ESP32 will do this for a maximum of 5 times. If it then still fails to send the instruction, the ESP32 and Nano M6E will reboot.

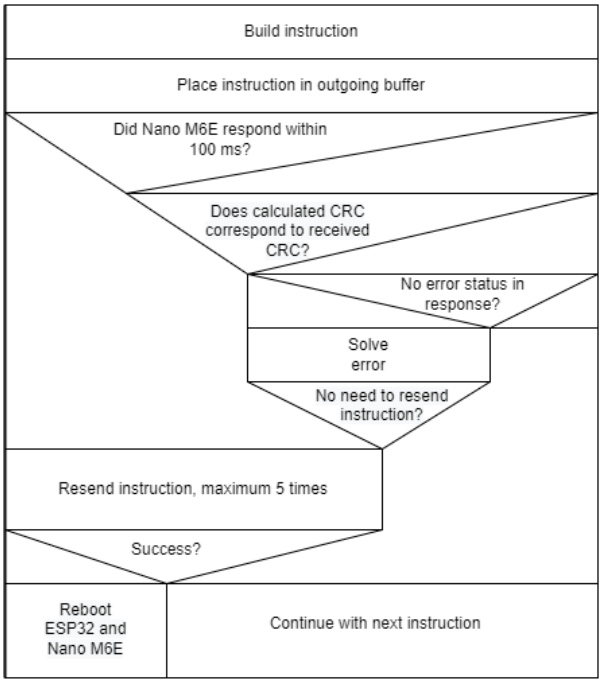


Figure 4.2: Nassi-Schneidermann diagram for the software running on the Nano M6E

If the Nano M6E did receive the instruction, it responds with a response as shown in Table A.38. The ESP32 will calculate the CRC based on the length, opcode, status and data bytes. If that calculated CRC does not correspond to the CRC that was placed in the response, something has gone wrong and the message was received wrongly. The ESP32 will send the instruction again. and will continue to this for a maximum of 5 times. If it then still fails to receive a response with a correct CRC, the ESP32 and Nano M6E will reboot.

If the received CRC was correct, the response was most likely not corrupted. The ESP32 then will look at the status bytes. If the status bytes will be equal to `0x0000`, the instruction was received correctly, the Nano M6E

[1] `https://github.com/sparkfun/SparkFun_Simultaneous_RFID_Tag_Reader_Library`

executed the instruction correctly and did tell the ESP32 that it did so. The ESP32 now will continue with the next instruction. If the status bytes are unequal to `0x0000`, some error may be present. It will resolve the error based on the information in the status bytes, as explained in section 4.5. If the error could not be solved, the ESP32 will send the instruction again and will continue to do this for a maximum of 5 times. If it then still fails to receive a response with status bytes `0x0000`, the ESP32 and Nano M6E will reboot.

4.7. Interference with the Existing Momo Medical Hardware

The existing Momo Medical hardware exist out of several parts, as shown in Figure 2.1. The connection between a panic button and surrounding BedSenses uses an 868 MHz RF connection. Since the ACS operates at that frequency band as well, some precautions had to be made in order for both systems to work simultaneously.Good to keep in mind is that When a resident presses their panic button, the panic button tries to connect to the surrounding BedSenses and if that fails, it repeats that sequence several times.
Tests have proven that the panic button was unable to connect to a BedSense, even when the distance between a BedSense and a panic button was less than one meter and the distance between the panic button and an ACS that was continuously reading was more than 5 meters. To compromise this problem, use can be made of different algorithms, e.g. active panic listening algorithm or pulse mode operation. Let's look at these two.

4.7.1. Active Listening Algorithm

The ACS can be listening actively at all time to see whether the any other device is sending signals at its operating frequency; then if no other device is, the ACS can start its search for tags, otherwise it can switch to low power mode and try again later.

4.7.2. Pulse Mode Operation

An easier solution is called Pulse Mode Operation (PMO). In this operation mode, the reader is not reading constantly but alternating between periods of reading and non-reading, in the latter phase it can instead process what has been read and communicate with the database. The time for both reading and non-reading phase can both be easily adjusted in the code. Due to small random variations, interference will be reduced.

4.7.3. Conclusions

The PMO algorithm was tested. Using a read cycle consisting out of a read time of 200 ms and a down time of 100 ms, the communication from a panic button to a BedSense, which were both close to the ACS was not influenced by the ACS. When the PMO was disabled; e.g. the ACS was continuously reading, there was no communication possible from the panic button to the BedSense. Chosen is to implement this algorithm in the software design.

4.8. Main Algorithm

In this section the main loop is described. A schematic overview is given in the Nassi-Schneiderman diagram in Figure 4.3. In the following subsections, each part indicated in this diagram will be explained.

4.8.1. UART Setup

The connection between the ESP32 and the Nano M6E is made possible by using a Universal asynchronous receiver-transmitter (UART) connection, where either (1) the ESP32 sends instructions in the form of instruction messages to the Nano M6E and it responds by confirming the reception of the instruction or (2) the Nano M6E sends a response message.

The UART setup function `uart_config()` mainly initializes the UART connection to a baud rate of 115200 baud, sets the physical TX and RX pins on the ESP32 and sets the parity and stop bits to the 8N1 standard, where there are 8 bits, no parity and one stop bit per data frame. This also is the standard configuration of the Nano M6E[28]. Other baud rates are possible, and in this function, one can set the baud rate at which the ESP32 should be listening to one of the standard baud rates in the range of 9600 - 921600 baud. The possible baud rates are shown in subsection A.2.1 This setup function checks if the arguments used to initialize the UART connection are valid and if not prints an error message.

4.8.2. WiFi Setup

The ESP32 can serve both as an access point and as an station for WiFi communication. For the purposes of the hardware built, the ESP32 needs to access the WiFi as a client and so should be a station. The setup of the WiFi starts by connecting to a hard-coded WiFi network with a given authentication mode, e.g. WPA2, WPA3, WPA2/WPA3, etc., and afterwards instructs the ESP32 to function as a station to be able to send data to the server and database of Momo Medical. To be able to send data, a HTTP client needs to also be setup, which is done afterwards with certificates so the client can authenticate itself to the database.

4.8.3. UNIX Time Setup

The UART setup also involves setting up the ESP32 time. The ESP32 uses two hardware timers but both of these timers indicate the time since system's startup. To adjust the time to UNIX time, use was made of a network time protocol (NTP) server after connecting to WiFi by making use of a simple network time protocol library. With this the internal timer of the ESP32 is calibrated. This calibrated time is used to attach a timestamp in UNIX time to the data sent to the database, when sending tag data.

4.8.4. Ready to Work

After the initialization steps, as described above, have been executed, the ACS starts reading forever, if no external problems occur. The read cycle, as indicated in section 4.3 consists out of two parts, one in which the ACS is reading tags and one in which it is processing what it read. This is also where pulse mode operation comes at play, explained in subsection 4.7.2. That algorithm will decide when to switch between the reading and processing data.

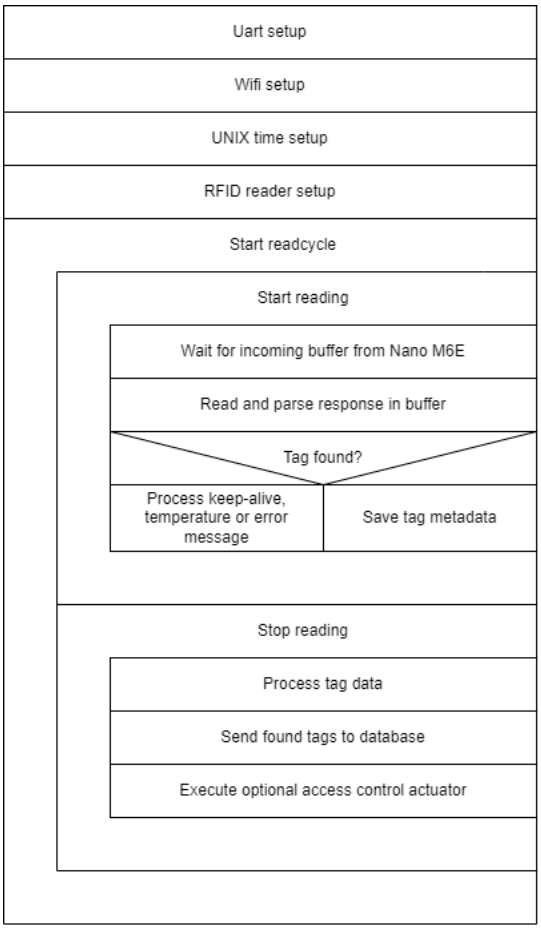


Figure 4.3: Main algorithm for the ACS

But not all messages received by the Nano M6E will be treated the same. Two reads are far less important at this stage, namely the messages regarding temperature and keep-alive. The focus has been only on tag data message for the moment.

For the system to be able to execute the actions necessary and update the data in the database, it reads the tag data indicated in Table A.34. After processing the data, some data is sent to the database, namely the RSSI of received signal, the id of the tag read, the id of the place the ACS belongs to, and the time of reading. In addition, the processed data is checked to see whether the tag read is fulfilling some requirements from the database or not and according to that outputs a binary 0 or 1.

4.9. Tag Metadata

During a readcycle, the struct `tagInfos` as indicated in Table A.34 will be updated when a new tag is found. In this section, the algorithms for several parameters which will be placed in the struct `tagInfos` will be explained.

4.9.1. Count
The count parameter corresponds to the number of times a specific count has been read during one readcycle. It will reset after each readcycle.

4.9.2. Timestamp
Two options are implemented for updating the struct `tagInfos` with a timestamp:

- The timestamp of the tag reading with the highest RSSI, or
- the timestamp of the last time this tag was read during the readcycle.

The second option will be used by default.

4.9.3. Dynamic Calculation of RSSI
The struct `tagInfos` contains three parameters concerning the RSSI of a specific tag found during one readcycle:

- The maximum RSSI,
- the mean RSSI, and
- the variance of the RSSI.

Which of the maximum RSSI or mean RSSI will be sent to the database can be configured. By default, the ACS sends the mean RSSI to the database. As discussed in section 3.4, the linear RSSI x_n is log-normal distributed and the RSSI in decibels is Gaussian distributed with a mean RSSI of

$$M = \frac{1}{k}\sum_{n=1}^{k} x_n \tag{4.1}$$

and an unbiased variance of

$$\sigma^2_{RSSI} = \frac{1}{k}\sum_{n=1}^{k} (x_n - M)^2 \tag{4.2}$$

To calculate a variance of the RSSI of a tag during one read cycle, it would be memory efficient to not store all RSSI values during the read cycle, but rather update the last stored variance with a new RSSI value. The algorithm to calculate the variance as given in Equation 4.2 does, however, calculates the RSSI after completing the read cycle, since it needs the mean μ which only can calculated after completing a read cycle.
Another algorithm [42] exists for calculating the variance, which is recursive and discards every last calculated variance S^2_{n-1} to update the new S^2_n for $1 < n < k$ where k is the total amount of times a specific tag is read per read cycle

$$S^2_n = S^2_{n-1} + \left(\frac{n-1}{n}\right)(x_n - M_{n-1})^2 \tag{4.3}$$

This works in theory. In practice, however, when using floating point numbers, this algorithm suffers from rounding errors.
Yet another algorithm, one that is implemented in the code for the ACS, is based on both algorithms discussed above [42]. First, during the read cycle, the mean RSSI value is updated and then the squared error to the updated mean is calculated and added to the last squared error as

$$E_n = E_{n-1} + (x_n - M_{n-1})(x_n - M_n) \tag{4.4}$$

where

$$M_n = \frac{M_{n-1}\cdot(n-1) + x_n}{n} \tag{4.5}$$

is the updated mean. One can easily see that Equation 4.4 resembles the sum of squared errors in Equation 4.1, and by dividing Equation 4.4 by the total number of samples k we find the recursively calculated variance

$$S^2_{RSSI} \approx \frac{E_k}{k}, k > 2. \tag{4.6}$$

This algorithm is implemented in the function `updateTagInfoWithRssiMeanAndSd()`, whose structure is explained in subsection A.3.3.

5

Design and Results

In this chapter, a summary on the complete design of the access control system will be given, after which it will be tested. The results of the tests will be given in section 5.2.

5.1. Final Design

Before looking at how well the system works, it is good to shortly go through the full final design. Let's look at the final hardware and software part, and the easy integration of them.

5.1.1. Hardware Design

Recall the conclusion on the hardware design: An ESP32 microcontroller, which will be connected using a WiFi connection to the Momo Medical database, is connected to the Nano M6E UHF RFID reader module using an UART connection. Attached to the Nano M6E is a vertical polarized patch antenna, the GSM-34-900. It is used to observe UHF RFID tags, which are attached using an adhesive layer onto panic buttons vertically. Attached to one of the GPIO pins on the ESP32 is a black box model of an actuator. The hardware design was explained in chapter 4. In the final design, the ESP32 microcontroller runs a program that indicate to the Nano M6E to either read or not, and the ESP32 also processes all the incoming data from the Nano M6E en outputs a binary value. The Nano M6E, thus, functions only as a transceiver being able to interact with tags and the microcontroller, with the antenna allowing it to send and receive signals.

5.1.2. Software Design

The Nano M6E has a internal microcontroller, on which a software is placed by the manufacturer of the Nano M6E. For a program to control the Nano M6E, it must use the API written for this software. This is what the program, discussed in chapter 4 running on the ESP32 is doing.

5.1.3. Bringing it all Together

Altogether, the final design consists out of a software system written which is ran on the ESP32 to use an Nano M6E to read RFID tags and based on the data read output a binary value that can be used by some other system. The access control system has been designed to be able to be able to do the following:

1. Observe UHF RFID tags up to a distance of at least 1 meter.
2. Change the ID of a given ACS unit.
3. Change the EPC of a tag.
4. Create a connection to the Momo Medical database.
5. Save data of observed tags, such as RSSI, EPC, custom data and a timestamp of observation.
6. When a tag was observed, send the found EPC, mean RSSI, timestamp of observation and ACS ID to the database.

5.2. Results

In this section, the specifications from subsection 5.1.3 will be confirmed by testing the design.

Distance
As can be seen in Figure 3.6, the AD-665u8 tag could be observed at a distance of 1.5 meter from the GSM-34-900 antenna within 100 milliseconds. It is concluded that requirement FR1 was achieved, as well as requirement FR2.

5.2.1. Reading Data

During a read cycle of one second of reading and one second of processing, nearby tags were observed. By printing some data in the program, the data as claimed in was printed at the end of the read cycle, which can be seen in Figure 5.1. This figure shows that indeed the timestamp at which a specific tag was observed last is stored in UNIX time, including milliseconds as required by FR4. The number of times a specific tag was seen in last read cycle is stored, and the data regarding the tag id (EPC), mean RSSI, and what the ACS ID is. This corresponds to which location (here door 1, which corresponds to ACS ID `00000001`), the ACS is placed.

```
(47105) Tag found: at timestamp 1671103028385, 28 times, EPC: 2, mean RSSI: -24.000000 dBm, at door 1
(47105) Tag found: at timestamp 1671103028471, 27 times, EPC: 3, mean RSSI: -26.000000 dBm, at door 1
(47115) Tag found: at timestamp 1671103028184, 31 times, EPC: 33a39c1d, mean RSSI: -50.000000 dBm, at door 1
(47135) Tag found: at timestamp 1671103028403, 37 times, EPC: 33a39c1e, mean RSSI: -45.274017 dBm, at door 1
(47145) Tag found: at timestamp 1671103028390, 29 times, EPC: 33a39c1f, mean RSSI: -20.000000 dBm, at door 1
(47155) Tag found: at timestamp 1671103028419, 38 times, EPC: 77f6850d, mean RSSI: -49.555607 dBm, at door 1
(47165) Tag found: at timestamp 1671103028265, 39 times, EPC: 77f6850e, mean RSSI: -59.755470 dBm, at door 1
(47175) Tag found: at timestamp 1671103028477, 31 times, EPC: 77f6850f, mean RSSI: -24.000000 dBm, at door 1
```

Figure 5.1: Observed tags in one read cycle, with their data

The timestamp of the last printed tag, when converted to a human-understandable date and time, corresponds to Thu Dec 15 2022 12:17:08 GMT+0100. This is as expected, since it corresponds to the time of the verification test. 8 tags were observed during this read cycle. Two of them were previously given another EPC: tag 2 and 3. The remaining 6 tags have their pre-programmed EPC. All tags were seen many times which gives a high certainty that they were in the neighbourhood of the GSM-34-900 antenna for some time. This can be confirmed by the fact that the RSSI for some tags was quite high (-20 dBm for 33a39c1f). All tags were said to be seen at door 1, which is correct. From this, it can be concluded that the data required to be sent to the database could be saved for further processing.

5.2.2. Database Connection

Now the data can be saved locally, it is time to send this data to the Momo Medical database. The function `sendDataToDB()`, which is called after each readcycle, is responsible for doing this. Using data visualisation tool called Grafana, the data received by the database can be seen. The results can be seen in Figure 5.2, where the data regarding ACS with ID 1 is printed.

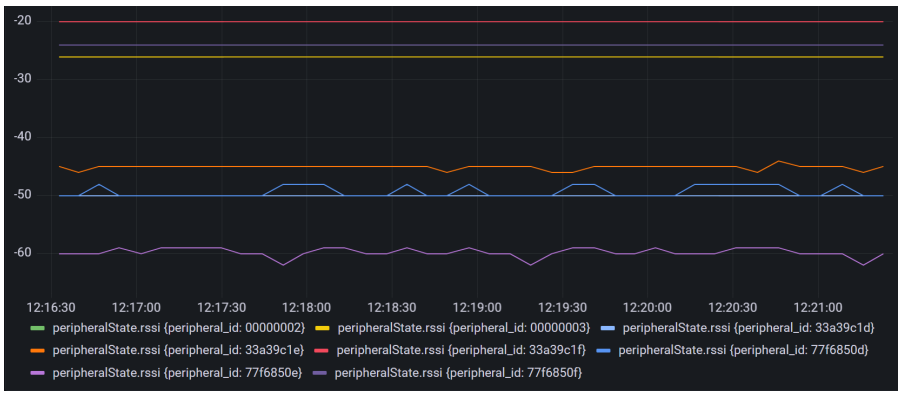


Figure 5.2: Visualization of the tag data as observed by ACS with ID `0000001`, sent to the Momo Medical database using Grafana. On the horizontal axis is time (GMT+1) and on the vertical axis is the RSSI in dBm. The lines correspond to the 8 tags which were observed, with their EPC as peripheral_id.

When comparing the data seen in Figure 5.1at timestamp 12:17:08 with the data printed in Figure 5.1, the found tag EPCs and their corresponding RSSI values correspond to the data saved locally. From this, it can be concluded that the connection between the ACS and the Momo Medical Database was correctly established, satisfying requirements FR3, FR4 and FR7. The Localisation System subgroup now has access to this data,

which can be used for their localisation algorithm. Since the ACS knows which panic buttons pass certain doors or locations, e.g. elevators or staircases, the Localisation System can know if panic buttons travelled to other floors inside nursery homes, which can be useful to gain a higher confidence of a three-dimensional location inside a building.

6

Discussion

Due to its technical and practical nature, this project had most of its requirements set from the beginning on, with minor additions/changes throughout the project. The time has come to put the final product side-by-side with the requirements documented in section 2.2 and see whether all the requirements have been satisfied.

Starting with the full system requirements that were set in the program of requirements, it can be seen that all of the requirements have been achieved. It is good to mention that although he location is determined once a minute (SR2), the ACS updates more often. For SR3, it is good to mention that for the system to work in a nursing home, some installation has to be done. For example, the layout of the rooms needs to be translated to the map format used by the system. Also, the ACS sensors need to be placed at every door room and other passageways. However, no (extensive) calibration is needed. The localization system has been designed such that it works after giving it general layout parameters, such as the distances to different type of neighbouring rooms. The ACS does not need any calibration at all.
Looking at the functional requirements, it has already been indicated that FR1, FR2, FR3, and FR4 have been satisfied. Now, not much has been discussed regarding FR5, specifically about **getting** updates from the database. The reason is because the check for access rights at the moment is based on hard-coded data. This does mean that the functionality of getting data from the database is already available and that by some tweaking, Momo Medical can have that functionality fully ready. The second part of FR5 is thus satisfied but not the first part of it. Further, FR6, FR7, FR8, and FR9 have also been satisfied.

Looking at the non-functional requirements, NFR2 and NFR3 have been fully satisfied, but NFR1 has been a challenge. The search for the necessary hardware was the most challenging part before starting with the actual hardware and software implementation. It not only took much more time than desired, it also did not result in the team finding all the hardware modules that can be used in large scale application. And NFR1 is not fully satisfied as the antenna of the system does not have a practical dimensions for the use Momo Medical has in mind. But not all parts of the hardware is unhandy; the ESP32 microcontroller is a really useful module that can be reused with any other RFID transceiver (and its antenna). This has the added benefit that the software written for the microcontroller, which contains the WiFi communication, database connection, and the tag data processing, can all be reused.

7

Conclusion, Recommendation, and Future Work

The challenge in this project was to design and build an access control system that could regulate access to rooms and sections by different people in nursing homes. The time spent on the project has resulted in a hardware system consisting out of the SparkFun Simultaneous RFID Reader, with the Nano M6E module on its board, a GSM-34-900 patch antenna connected to it, and the ESP32-WROVER-B microcontroller. The hardware system is dependant on the fact that the people should carry a UHF RFID tag with at all times and based on those tags can recognize the different people. This tag should be placed on a panic button in a vertical orientation. The microcontroller is in addition dependant on a software written for this project, which enables it to process tag data it gets from the Nano M6E and communicate via WiFi to send data to the database. The final system satisfies most requirements set in the beginning of the project but needs some tweaking to be able to get updates from the database and to be useful for commercial implementations.

For improving the system, there is still some work to be done. As mentioned above, the first thing to do is to get the updates necessary from the database, for example all new tag ids added to the system with the update of which tag belongs to which room/section. Afterwards, focus should be set on reducing the hardware set in size so it can be installed wherever desired. One more improvement to the system is to add more conditions to what should happen when a specific tag is found and what the priorities are; for this questions like "when one person is allowed to enter a room and one is not but they both try to enter, should the door open or not?" should be answered. The implementations of these conditions are quite easy as the software for the "door sensor" itself has been written with this in mind.

Bibliography

[1] Shancang Li, Li Xu, and Shanshan Zhao. “The internet of things: A survey”. In: Information Systems Frontiers 17 (Apr. 2014). doi: 10.1007/s10796-014-9492-7.

[2] Wen Yao, Chao-Hsien Chu, and Zang Li. “The use of RFID in healthcare: Benefits and barriers”. In: 2010 IEEE International Conference on RFID-Technology and Applications. 2010, pp. 128–134. doi: 10.1109/RFID-TA.2010.5529874.

[3] R. Weinstein. “RFID: a technical overview and its application to the enterprise”. In: IT Professional 7.3 (2005), pp. 27–33. doi: 10.1109/MITP.2005.69.

[4] Bluetooth Low Energy (BLE) 101: A Technology Primer with Example Use Cases. 2014.

[5] Alseny Diallo, Zaixin Lu, and Xinghui Zhao. “Wireless Indoor Localization Using Passive RFID Tags”. In: Procedia Computer Science 155 (2019). The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019),The 14th International Conference on Future Networks and Communications (FNC-2019),The 9th International Conference on Sustainable Energy Information Technology, pp. 210–217. issn: 1877-0509. doi: https://doi.org/10.1016/j.procs.2019.08.031. url: https://www.sciencedirect.com/science/article/pii/S1877050919309457.

[6] Damir Arbula and Sandi Ljubic. “Indoor Localization Based on Infrared Angle of Arrival Sensor Network”. In: Sensors 20.21 (Nov. 2020), p. 6278. doi: 10.3390/s20216278. url: https://doi.org/10.3390/s20216278.

[7] Jun Qi and Guo-Ping Liu. “A Robust High-Accuracy Ultrasound Indoor Positioning System Based on a Wireless Sensor Network”. In: Sensors 17.11 (Nov. 2017), p. 2554. doi: 10.3390/s17112554. url: https://doi.org/10.3390/s17112554.

[8] Busra Ozdenizci, Vedat Coskun, and Kerem Ok. “NFC Internal: An Indoor Navigation System”. In: Sensors 15.4 (Mar. 2015), pp. 7571–7595. doi: 10.3390/s150407571. url: https://doi.org/10.3390/s150407571.

[9] Alseny Diallo, Zaixin Lu, and Xinghui Zhao. “Wireless Indoor Localization Using Passive RFID Tags”. In: Procedia Computer Science 155 (2019), pp. 210–217. doi: 10.1016/j.procs.2019.08.031. url: https://doi.org/10.1016/j.procs.2019.08.031.

[10] Lu Bai et al. “A Low Cost Indoor Positioning System Using Bluetooth Low Energy”. In: IEEE Access 8 (2020), pp. 136858–136871. doi: 10.1109/access.2020.3012342. url: https://doi.org/10.1109/access.2020.3012342.

[11] Mahmut Aykaç, Ergun Erçelebi, and Noor Baha Aldin. “ZigBee-based indoor localization system with the personal dynamic positioning method and modified particle filter estimation”. In: Analog Integrated Circuits and Signal Processing 92.2 (Apr. 2017), pp. 263–279. doi: 10.1007/s10470-017-0969-4. url: https://doi.org/10.1007/s10470-017-0969-4.

[12] Fabian Höflinger et al. “Indoor-localization system for smart phones”. In: 2015 IEEE International Workshop on Measurements Networking (MN). 2015, pp. 1–6. doi: 10.1109/IWMN.2015.7322974.

[13] Shadi Al-Sarawi et al. “Internet of Things (IoT) communication protocols: Review”. In: 2017 8th International Conference on Information Technology (ICIT). 2017, pp. 685–690. doi: 10.1109/ICITECH.2017.8079928.

[14] Farzad Samie, Lars Bauer, and Jörg Henkel. “IoT technologies for embedded computing: A survey”. In: 2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS). 2016, pp. 1–10.

[15] Bluetooth Low Energy vs. Radio Frequency Identification, p. 5. url: https://www.link-labs.com/bluetooth-low-energy-vs-rfid-ebook.

[16] B lsinglawi et al. “RFID Systems in Healthcare Settings and Activity of Daily Living in Smart Homes: A Review.” In: 6 (2017), pp. 1–17. doi: 10.4236/etsn.2017.61001..

[17] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. “A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi”. In: IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society. 2007, pp. 46–51. doi: 10.1109/IECON.2007.4460126.

[18] R. Porkodi and V. Bhuvaneswari. “The Internet of Things (IoT) Applications and Communication Enabling Technology Standards: An Overview”. In: 2014 International Conference on Intelligent Computing Applications. 2014, pp. 324–329. doi: 10.1109/ICICA.2014.73.

[19] The Global Language of Business. EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID. 2015.

[20] Regeling gebruik van frequentieruimte zonder vergunning en zonder meldingsplicht 2015. url: https://wetten.overheid.nl/BWBR0036378/2021-12-15. (accessed: 05.12.2022).

[21] The Global Language of Business. Overview of UHF frequency allocations (860 to 960 MHz) for RAIN RFID. Nov. 2022.

[22] ThingMagic. THINGMAGIC NANO USER GUIDE. Version 09172018. Sept. 2018.

[23] Ltd. Chang Hong Information Co. 860-960 MHz PANLE 6 dBi FOR N JACK.

[24] Arduino. Arduino UNO R3 Product Reference Manual. 2022.

[25] ESPRESSIF. ESP32-WROVER-B ESP32-WROVER_IB Datasheet. Version 1.9. 2022.

[26] Marijn Boringa, Job van Erp, and Corné Ploumen. An Indoor Localization System for Nursing Homes Based on RSSI. 2022.

[27] Daniel M. Dobkin. The RF in RFID. 2nd ed. Elsevier, 2012. isbn: 9780123945839.

[28] ThingMagic. ThingMagic Nano Design Guide. Mar. 2015.

[29] Avery Dennison. UHF RFID Inlays AD-321r6. 2015.

[30] Daniel Deavours and Daniel Dobkin. “UHF Passive RFID Tag Antennas”. In: Oct. 2010, pp. 263–303. isbn: 9780470681923. doi: 10.1002/9780470973370.ch9.

[31] Impinj. Indy R500 Reader Chip(IPC-R500) Electrical, Mechanical, Thermal Specification. 2012.

[32] ATMEL. Atmel SAM D21E / SAM D21G / SAM D21J ARM-Based Microcontroller. 2014.

[33] Skyworks. SKY77768 SkyHi Power Amplifier Module for WCDMA , HSDPA , HSUPA , HSPA+ , LTE – Band VIII (880–915 MHz). 2013.

[34] Texas Instruments. SKY77768 SkyHi Power Amplifier Module for WCDMA , HSDPA , HSUPA , HSPA+ , LTE – Band VIII (880–915 MHz). 2013.

[35] Impinj. Product Change Notification: End-of-Life Notification for Impinj Indy RAIN RFID R500 and R2000 Reader Chips. Accessed on December 14, 2022. 2022. url: https://support.impinj.com/hc/en-us/articles/8865436761107-Product-Change-Notification-End-of-Life-Notification-for-Impinj-%20Indy-RAIN-RFID-R500-and-R2000-Reader-Chips.

[36] Impinj. Impinj E900 Series RAIN RFID Reader Chips. 2022.

[37] P. Lin. Determining the relative position of a device in a wireless network with minimal effort.

[38] Bao Hua Liu et al. “The impact of fading and shadowing on the network performance of wireless sensor networks”. In: International Journal of Sensor Networks 3.4 (2008), p. 211. doi: 10.1504/ijsnet.2008.019006. url: https://doi.org/10.1504/ijsnet.2008.019006.

[39] Antonio Lazaro, David Girbau, and David Salinas. “Radio Link Budgets for UHF RFID on Multipath Environments”. In: IEEE Transactions on Antennas and Propagation 57.4 (2009), pp. 1241–1251. doi: 10.1109/TAP.2009.2015818.

[40] ThingMagic. Autonomous Configuration Tool User Guide. Mar. 2016.

[41] ThingMagic. Mercury API.

[42] B. P. Welford. “Note on a Method for Calculating Corrected Sums of Squares and Products”. In: Technometrics 4.3 (1962), pp. 419–420. doi: 10.1080/00401706.1962.10490022.

Appendices

A

C/C++ functions

In this appendix, the goal is to give an overview of all the functions used to create the software for the hardware. Highlighted text, indicates default values. For each function, a table is shown with inputs and outputs in addition to the explanation provided.

A.1. Main.c

No specific functions but the main function is defined in this file. The main function has been extensively explained in section 4.8.

A.2. NanoHelperFunctions.c

This file contains C functions which are used to configure the Nano M6E, read and write data to tags etc.

A.2.1. `setBaud()`

Sets the baud rate of the Nano M6e. Possible values can be chosen from the range [9600 - 19200 - 38400 - **115200** - 230400 - 460800 - 921600] baud [28], where the bold value is the default baud rate.

Table A.1: Inputs and outputs of `setBaud()`

Inputs		Outputs
`baudRate`	uint32_t	void

A.2.2. `setReadPower()`

Sets the power at which the Nano M6E reads. Range 0-27 dBm (0-500 mW). Range for input `readPower` = 0-**2700 cdBm**

Table A.2: Inputs and outputs of `setReadPower()`

Inputs		Outputs
`readPower`	uint16_t	void

A.2.3. `getReadPower()`

Ask the Nano M6E at which power it is initialized to read. Returns this read power. Range 0-27 dBm (0-500 mW). Range for output `readPower` = 0-2700 cdBm

Table A.3: Inputs and outputs of `getReadPower()`

Inputs	Outputs	
void	`readPower`	uint16_t

A.2.4. `setWritePower()`

Sets the power at which the Nano M6E can write data adn EPCs to its internal memory. Range 0-27 dBm (0-500 mW). Range for input `writePower` = 0-2700 cdBm

Table A.4: Inputs and outputs of `setWritePower()`

Inputs		Outputs
`writePower`	uint16_t	void

A.2.5. `getWritePower()`

Ask the Nano M6E at which power it is initialized to write data and EPCs. Returns this write power. Range 0-27 dBm (0-500 mW). Range for output `writePower` = 0-2700 cdBm

Table A.5: Inputs and outputs of `getWritePower()`

Inputs	Outputs	
void	`writePower`	uint16_t

A.2.6. `setRegion()`

Sets the region of the module to cover for the legal frequency bands. Possible regions are found in Table A.8.

Table A.6: UHF RFID regions and arguments, from `tmr_regions.h` [41]

Region	**Argument**
Unspecified	`0x00`
North America	`0x01`
European Union	`0x02`
Korea	`0x03`
India	`0x04`
Japan	`0x05`
PR China	`0x06`
European Union 2	`0x07`
European Union 3	`0x08`
Korea 2	`0x09`
PR China (840 MHz)	`0x0A`
Australia	`0x0B`
New Zealand	`0x0C`
Reduced FCC region	`0x0D`
5 MHz band	`0x0E`
Israel	`0x0F`
Malaysia	`0x10`
Indonesia	`0x11`
Philippines	`0x12`
Taiwan	`0x13`
Macau	`0x14`
Russia	`0x15`
Singapore	`0x16`
Japan 2	`0x17`
Japan 3	`0x18`
Vietnam	`0x19`
Thailand	`0x1A`
Argentina	`0x1B`
Hong Kong	`0x1C`
Bangladesh	`0x1D`
European Union 4	`0x1E`
Open region	`0xFF`

Table A.7: Inputs and outputs of `setRegion()`

Inputs		Outputs
`region`	uint8_t	void

A.2.7. `getRegion()`

Returns the region the Nano M6E is initialized at. The returned byte corresponds to a region specified in Table A.6.

Table A.8: Inputs and outputs of `getRegion()`

Inputs	Outputs	
void	`region`	uint8_t

A.2.8. `setAntennaPort()`

Sets the port at which the antenna can be found. The Nano M6E only supports an RX antenna on port `0x01` and a TX antenna on port `0x01`. These are used in initializing the Nano M6E.

Table A.9: Inputs and outputs of `setAntennaPort()`

Inputs	Outputs
void	void

A.2.9. `setTagProtocol()`

Tells the Nano M6E which UHF RFID protocol to use. Sets the Nano 6ME to read tags with the Gen2/ISO18000-c protocol, when `protocol` is set to `0x02`. Other protocols are possible. In Table A.10, the supported protocols and their corresponding arguments are listed. The arguments are extracted from `tmr_tag_protocol.h` [41]. The default protocol used is the **GEN2/ISO18000-C** protocol.

Table A.10: Supported UHF RFID protocols

Protocol	None	ISO180006-B	GEN2/ISO18000-C	ISO180006-B_Ucode	IPX64	IPX256	ATA
Argument	`0x00`	`0x03`	`0x05`	`0x06`	`0x07`	`0x08`	`0x1D`

Table A.11: Inputs and outputs of `setProtocol()`

Inputs		Outputs
`protocol`	uint8_t	void

A.2.10. `getTagDataBytes()`

Returns the length in bytes of the data string embedded in a tagReadMessage as specified in Table 4.6.

Table A.12: Inputs and outputs of `getTagDataBytes()`

Inputs	Outputs	
void	`length`	uint8_t

A.2.11. `getTagEPCBytes()`

Returns the length in bytes of the EPC embedded in a tagReadMessage as specified in Table 4.6. Default EPC length = 12 bytes.

Table A.13: Inputs and outputs of `getTagEPCBytes()`

Inputs	Outputs	
void	`length`	uint8_t

A.2.12. getTagTimestamp()

Returns the timestamp in ms a tag was read since last keep-alive signal as specified in Table 4.6.

Table A.14: Inputs and outputs of `getTagTimestamp()`

Inputs	Outputs	
void	`timestamp`	uint16_t

A.2.13. getTagFreq()

Returns the frequency (in kHz) a tag was read at.

Table A.15: Inputs and outputs of `getTagFreq()`

Inputs	Outputs	
void	`frequency`	uint32_t

A.2.14. getTagRSSI()

Returns the RSSI a tag was read with in dBm. Encoded as two's complement with a resolution of 1 dBm.

Table A.16: Inputs and outputs of `getTagRSSI()`

Inputs	Outputs	
void	`rssi`	int8_t

A.2.15. getTagSignalPhase()

Returns the signal phase a tag was read at. Range 0-180 degrees.

Table A.17: Inputs and outputs of `getTagSignalPhase()`

Inputs	Outputs	
void	`phase`	uint16_t

A.2.16. getTemperature()

Returns the internal temperature of the Nano M6E in degrees Celsius.

Table A.18: Inputs and outputs of `getTemperature()`

Inputs	Outputs	
void	`temperature`	uint8_t

A.2.17. getPowerMode()

Returns the power mode the Nano M6E is operating at.

Table A.19: Inputs and outputs of `getPowerMode()`

Inputs	Outputs	
void	`power mode`	uint8_t

A.2.18. getHopTable()

Returns the table in which all frequencies the Nano M6E reads the tags.

Table A.20: Inputs and outputs of `getHopTable()`

Inputs	Outputs	
void	`hop table`	uint32_t[]

A.2.19. enablePowerSave()

Set the power save mode. When enabled, it has less power consumption. When disabled, it has better reading sensitivity.

Table A.21: Inputs and outputs of enablePowerSave()

Inputs		Outputs
power mode	bool	void

A.2.20. readTagEPC()

Saves the EPC of length epcLength at the address of epc. If no tag is found before a timeout of timeOut ticks has been reached, this function returns a status code of 0x0C. When successfully stored an EPC, this function returns a status code of 0x0B

Table A.22: Inputs and outputs of readTagEPC()

Inputs		Outputs	
*epc	uint8_t	status	uint8_t
epcLength	uint8_t		
timeOut	uint16_t		

A.2.21. writeTagEPC()

Writes a new EPC newEPC of length newEPCLength to the first tag that will be detected before a timeout of timeOut ticks has been reached. If successful, this function returns a status code of 0x0B. If not successful, this function returns a status code of 0x0C.

Table A.23: Inputs and outputs of writeTagEPC()

Inputs		Outputs	
*newEPC	char	status	uint8_t
newEPCLength	uint8_t		
timeOut	uint16_t		

A.2.22. resetNano()

Resets the Nano M6E by pulling its enable pin low for 55 milliseconds.

Table A.24: Inputs and outputs of resetNano()

Inputs	Outputs
void	void

A.2.23. configReader()

This functions calls several other functions and is used to configure the Nano M6E. The functions called are

1. stopReading() to stop any ongoing readcycle.
2. setAntennaPort() to initialize the rx/tx antennae ports.
3. setRegion(region) to set the operating frequency/region.
4. setBaud(baudRate) to set the baudrate at which the Nano M6E should communicate.
5. setReadPower(readPower) to set the power at which the Nano M6E should read.
6. setWritePower(writePower) to set the power at which the Nano M6E should write.
7. setTagProtocol(protocol) to set the operating protocol.

Table A.25: Inputs and outputs of `configReader()`

Inputs		Outputs
`region`	uint_8t	void
`baudRate`	uint16_t	
`readPower`	uint16_t	
`writePower`	uint16_t	
`protocol`	uint8_t	

A.2.24. `configReaderDefault()`

This function calls the function `configReader()` with the default values for `baudRate`, `readPower`, `Region` and `protocol` as shown in Table A.26.

Table A.26: Default parameters for `configReader()`

Parameter	baudRate	readPower	writePower	region	protocol
Value	115200	2700	2700	0x08	0x05
Description	baud	cdBm	cdBm	EU3	Gen2/ISO18000-c

Table A.27: Inputs and outputs of `configReaderDefault()`

Inputs	Outputs
void	void

A.2.25. `parseResponse()`

Parses the automatic responses the Nano M6E sends to the ESP32. Returns what kind of signal is received, which are described in Table A.28. Used in other functions to extract useful data from these messages.

Table A.28: Response

Status	Description
2	CRC failed, response discarded
4	Opcode not equal to 0x22, not an automatic response
5	Temperature response, once every ~1 second
6	Keep-alive response, once every ~0.25 seconds
7	Max allowed temperature exceeded - reading stopped
8	A tag was found
9	Error message
10	Error

Table A.29: Inputs and outputs of `parseResponse()`

Inputs		Outputs	
`length`	uint8_t	`status`	uint8_t

A.2.26. `getTimeMs()`

Returns UNIX time but in ms.

Table A.30: Inputs and outputs of `getTimeMS()`

Inputs	Outputs	
void	`time`	int8_t

A.2.27. calculateCRC()

This function calculates a Cyclic Redundancy Check (CRC) and returns the two CRC bytes which will be added to the message to be sent or compared to the CRC bytes of the received response.
It is calculated by using the following algorithm, extracted from `serial_reader_l3.c`[41].

```
uint16_t crc = 0xFFFF;

  for (uint8_t i = 0; i < len; i++)
  {
    crc = ((crc << 4) | (u8Buf[i] >> 4)) ^ crctable[crc >> 12];
    crc = ((crc << 4) | (u8Buf[i] & 0x0F)) ^ crctable[crc >> 12];
  }

  return crc;
```

The array `crctable` is defined as an array of 16 bytes, where `crctable[i]` = `crctable[i-1]` + `0x1021`, with `crctable[0]` = `0x0000` and `crctable[1]` = `0x1021`.

Table A.31: Inputs and outputs of `calculateCRC()`

Inputs		Outputs	
`message`	uint8_t	`crc`	uint16_t
`length`	uint8_t		

A.2.28. readData()

The function `readData()` is used to read incoming data with length `dataLengthRead` in bytes. It starts reading at byte `address`, and stores the read data at the address of `dataRead`.
When nothing could be read before a timeout `timeout` in ticks was reached, this functions outputs a `status` code `0x0C`. When data was read successfully before a timeout was reached, this functions outputs code `0x0B`.
The data that should be read is dependant on the chosen `bank` as depicted in Table A.32.

Table A.32: Bank description

Bank 0							
bytes	[0]	[1]	[2]	[3-4]	[5-8]	[9-12]	
description	Success	length in bytes	opcode	status	Kill password	Access password	
Bank 1							
bytes	[0]	[1]	[2]	[3-4]	[5-6]	[7-9]	[8-8+N]
description	Success	length in bytes [N]	opcode	status	CRC	PC	EPC
Bank 2							
bytes	[0]	[1]	[2]	[3-4]	[5]	[6-8]	[9-38]
description	Success	length in bytes	opcode	status	CIsID	0xaaabbb Vendor ID (a) Model ID (b)	Unique ID
Bank 3							
bytes	[0]	[1]	[2]	[3-4]	[5-5+N]		
description	Success	length in bytes [N]	opcode	status	User data		

The inputs and outputs are shown in Table A.33.

Table A.33: Inputs and Outputs of `readData()`

Inputs		Outputs	
`bank`	uint8_t	status	uint8_t
`address`	uint32_t		
`*dataread`	uint8_t		
`dataLengthRead`	uint8_t		
`timeOut`	uint16_t		

A.3. Doorsensor.cpp

This C++ file contains functions that are responsible for everything that happens before or after data is read by the Nano M6E, like initializing the system and taking decisions based on tags seen. First, an important `struct` will be declared. This `struct` will be used throughout this C++ file. After the struct declaration, the functions declared in this C++ file will be explained.

A.3.1. `struct TagInfo`

The struct TagInfo will be used in several functions and stores the metadata of a tag read. It looks like the following:

```
struct TagInfo
{
    uint32_t id;
    uint8_t idByteLength;
    int8_t rssi;
    float rssi_sd;
    float rssi_mean;
    uint32_t frequency;
    uint64_t timeStamp;
    uint16_t phase;
    uint8_t count;
};
```

where the individual components are explained in Table A.34.

Table A.34: Description of components of struct `TagInfo`

Parameter	Description	Type	Unit
id	Tag EPC	uint32_t	n/a
idByteLength	Length of tag EPC	uint8_t	n/a
rssi	RSSI	int8_t	dBm
rssi_sd	Variance of received RSSI of a specific tag read in an ongoing read cycle	float	dBm
rssi_mean	Mean RSSI of a specific tag read in an ongoing read cycle	float	mW
frequency	Frequency on which a tag was read	uint32_t	kHz
timestamp	UNIX time at which this tag was read	uint64_t	ms
phase	Phase this tag was read at	uint16_t	degrees
count	How many times this tag has been read in ongoing read cycle	uint8_t	n/a

A.3.2. `bytesToInt()`

This function converts a array of `tagEPCLenght` bytes to an integer. Used to convert the standard EPC, which is received as an array of bytes and stored in `tagEPC[]` to an integer, which can be sent to the database.

Table A.35: Inputs and outputs of `bytesToInt()`

Inputs		Outputs	
`tagEPC[]`	uint8_t	`tagId`	int32_t
`tagEPCLength`	uint8_t		

A.3.3. updateTagInfoWithRssiMeanAndSdAndCount()

During a read cycle, this function uses the new RSSI value and the stored RSSI mean and variance, due to previous reads in the current readcycle, to update the RSSI mean and variance values, as discussed in subsection 4.9.3. It also updates the tagCount parameter, which is a measure for how many times this specific tag has been read in the ongoing readcycle.

Table A.36: Inputs and outputs of updateTagInfoWithRssiMeanAndSdAndCount()

Inputs		Outputs
tagInfo	TagInfo	void

A.3.4. getTagInfo()

This function will be called when a tag has been read. It then extracts the metadata from the tag to save this in the struct TagInfo. It starts doing this by calling the functions

```
int8_t rssi             = getTagRSSI();
uint8_t count           = 0;
uint8_t idLength        = getTagEPCBytes();
uint16_t phase          = getTagSignalPhase();
uint32_t frequency      = getTagFreq();
uint64_t readTimeStamp  = getCurrentTimeMs();

uint8_t tagIdBytes[idLength];
getTagEPC(tagIdBytes, idLength);
uint32_t tagId = bytesToInt(tagIdBytes, idLength);
```

where the parameters for the read RSSI (rssi), length of the EPC (idLength), timestamp of reading (readTimeStamp), phase (phase), frequency (frequency) and EPC (tagId) are set. It then calls the function updateTagInfoWithRssiMeanAndSdAndCount(), where the updated RSSI mean and variance and the amount the tag with this specific EPC is read in the ongoing read cycle are set. Then, it fills in the struct tagInfo of type TagInfo

```
TagInfo tagInfo =  {.id = tagId, .idByteLength = idLength, .rssi = rssi,
                    .rssi_sd = rssi_sd, .rssi_mean = rssi_mean, .frequency = freq,
                    .timeStamp = readTimeStamp, .phase = phase, .count = tagCount};
```

This function concludes by storing this metadata-filled struct for the whole read cycle.

Table A.37: Inputs and outputs of getTagInfo()

Inputs	Outputs
void	void

A.3.5. Read()

This function checks how many bytes there are in the incoming RX buffer on the ESP32. If there are any bytes in this buffer, they will be read and stored in the global variable MSG[] by calling parseResponse(). This function also returns a status, with the meaning of different statuses depicted in Table A.38:

Table A.38: Response reactions

Status	Meaning	Action
2	CRC failed	Discard response
4	Not opcode 0x22	Logs the incoming response
5	Temperature	Logs the temperature
6	Keep-alive response	Calls the `getTimeMS()` function in order to connect the correct UNIX timestamp to the tag .timeStamp parameter
7	Overheating alarm	Stops reading immediately, logs the timestamp
8	A tag was found	Calls the `getTagInfo()` function to process this response
9	An other error occurred	Logs the error code included in the response.
10	This cannot happen, but is implemented to catch exceptions	n/a
default	This cannot happen, but is implemented to catch exceptions	n/a

Table A.39: Inputs and outputs of `Read()`

Inputs	Outputs
void	void

A.3.6. `checkTagsFound()`

This function will be called after each `stopReading()` function. It will iterate over all the tags read during the readcycle and will log how many different tags were found in the last read cycle, what their mean and max RSSI values were. Also, based on the tags seen it will decide the output of the ACS.

Table A.40: Inputs and outputs of `checkTagsFound()`

Inputs	Outputs
void	void

A.3.7. `setupTime()`

This function is responsible for retreiving the UNIX time using the WiFi connection, as explained in subsection 4.8.3.

Table A.41: Inputs and outputs of `setupTime()`

Inputs	Outputs
void	void

A.3.8. `getCurrentTimeMs()`

This function returns the current UNIX time but in ms.

Table A.42: Inputs and outputs of `getCurrentTimeMs()`

Inputs		Outputs
time_ms	uint64_t	void

A.3.9. `uart_config()`

This function will initialize the uart connection between the ESP32 and the Nano M6E as discussed in subsection 4.8.1.

Table A.43: Inputs and outputs of `uart_config()`

Inputs	Outputs
void	void

A.3.10. `startReadCycle()`

This function is called from the main to indicate to the ESP32 to drive the Nano M6E is such a way so it reads. After some time, the reading is stopped and the ESP32 starts up the processing part of the program. In between, check is done to see whether everything is running or some error or unexpected event has occurred.

Table A.44: Inputs and outputs of `startReadCycle()`

Inputs		Outputs
timeOn	uint16_t	void
timeOff	uint16_t	

A.3.11. `turnOnGreenLed()`

This function turns on a green LED located on GPIO `GREEN_LED_GPIO_PIN`

Table A.45: Inputs and outputs of `turnOnGreenLed()`

Inputs	Outputs
void	void

A.3.12. `turnOnRedLed()`

This function turns on a red LED located on GPIO `RED_LED_GPIO_PIN`

Table A.46: Inputs and outputs of `turnOnRedLed()`

Inputs	Outputs
void	void

A.3.13. `turnOffLeds()`

This function turns off all external LEDs

Table A.47: Inputs and outputs of `turnOffLeds()`

Inputs	Outputs
void	void

A.3.14. `encodeMessageAndAddToBuffer()`

This function encodes a message so that it can be sent using HTTP. It stores the encoded message in messagesToSendBuffer which it receives as input, see A.48.

Table A.48: Inputs and outputs of `encodeMessageAndAddToBuffer()`

Inputs		Outputs
`message`	MomoMessages_GeneralMessage *	void
`total_message_size_in_bytes`	int &	
`messagesToSendBuffer`	std::vector<std::string> &	

A.3.15. createGeneralMessage

Table A.49: Inputs and outputs of `createGeneralMessage()`

Inputs		Outputs	
`tagInfo`	TagInfo	`message`	MomoMessages_GeneralMessage

A.3.16. `sendDataToDB()`

This function goes through all the tags found and stored in the `DoorSensor` object and with the help of other functions creates a message containing the necessary data to be sent to the database and sends that data to the database.

Table A.50: Inputs and outputs of `sendDataToDB()`

Inputs	Outputs
void	void

B

Figures

In this appendix, several figures are placed for clarification purposes.

B.1. Antenna test setup

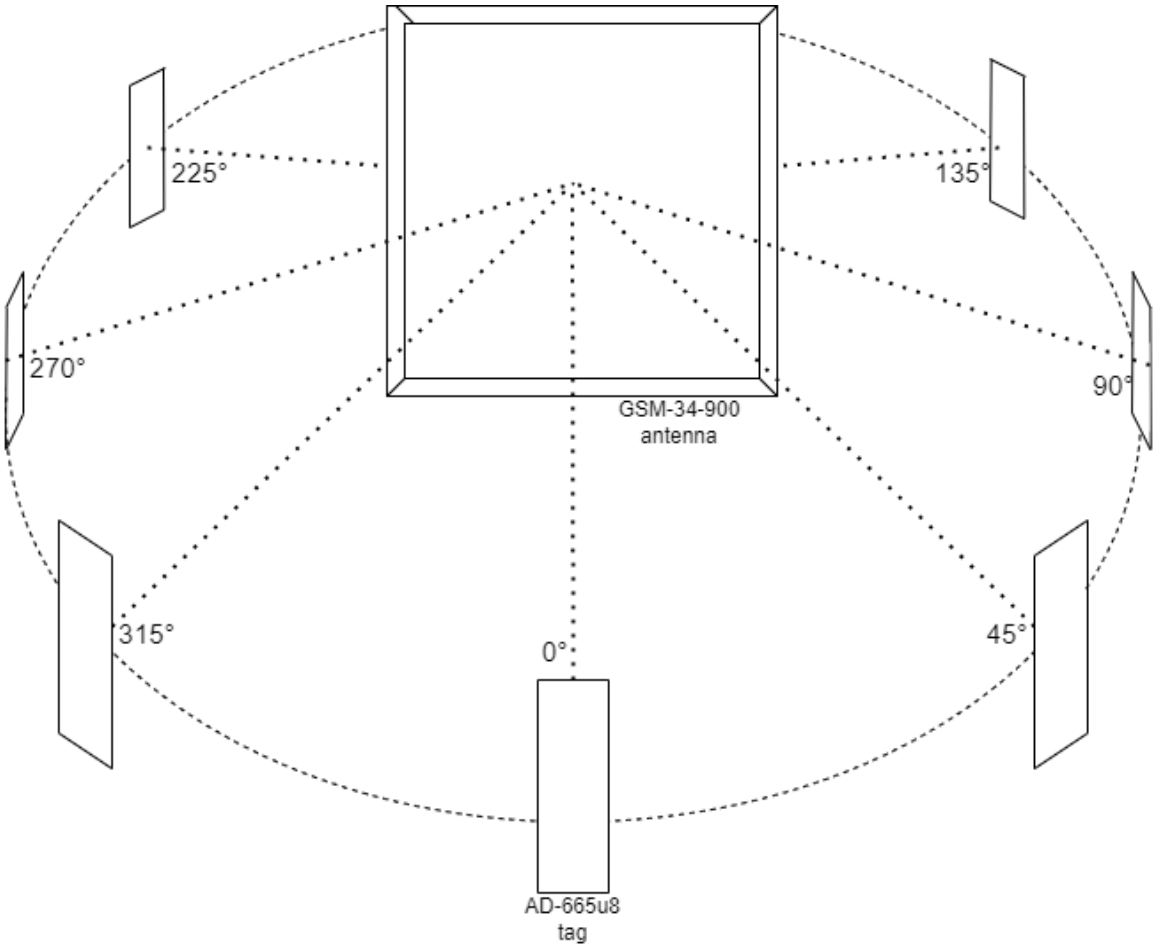


Figure B.1: Two-way directivity measurement setup. The AD-665u8 tag was placed vertically and perpendicular to the GSM-34-900 antenna. The tag was placed at the same height as the GSM-34-900 antenna (1m above the floor) and at angles 45 degrees difference. This was repeated for a distance of 50, 100 and 150 centimeter to the antenna.

B.2. Breakdown of the Nano M6E

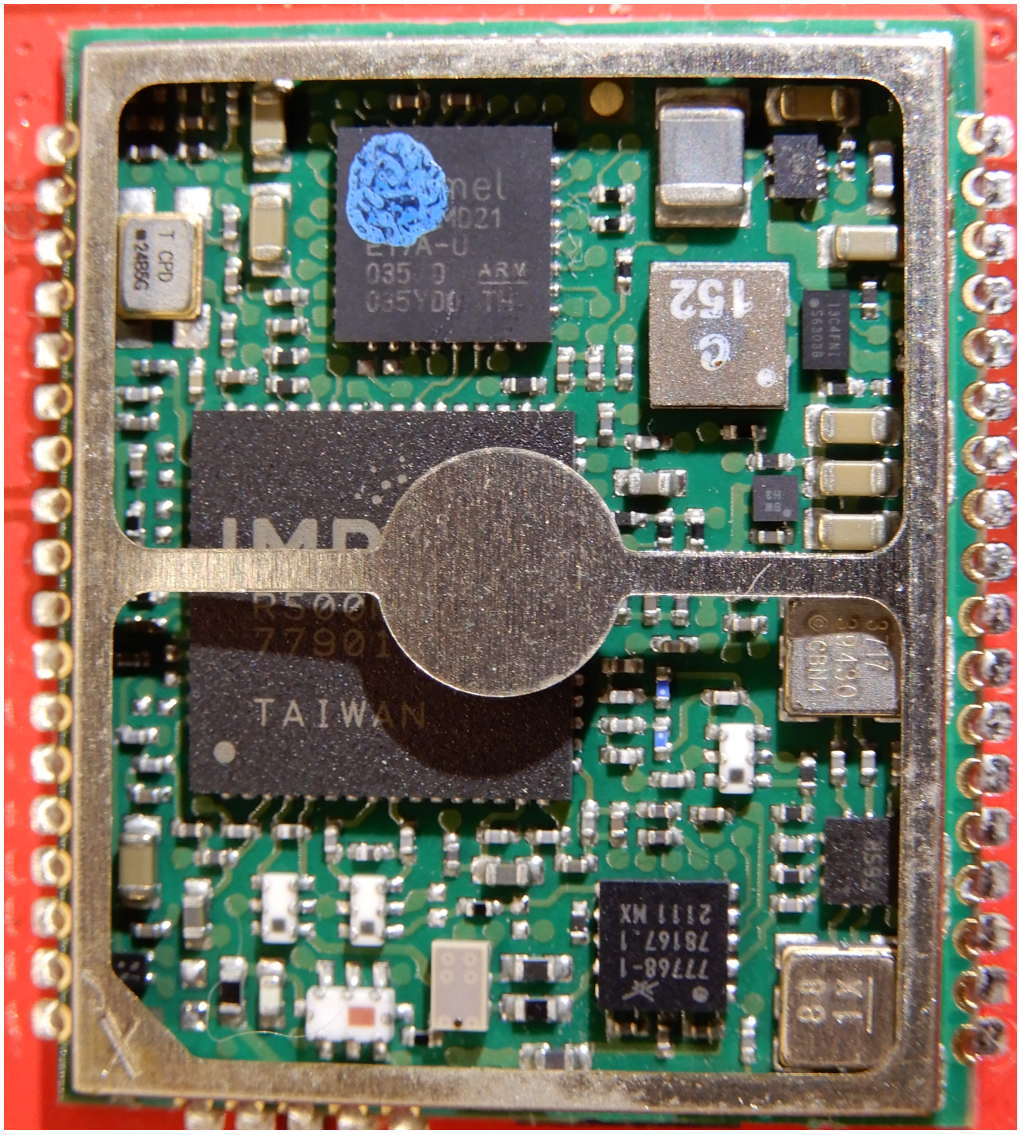


Figure B.2: Nano M6E inner components