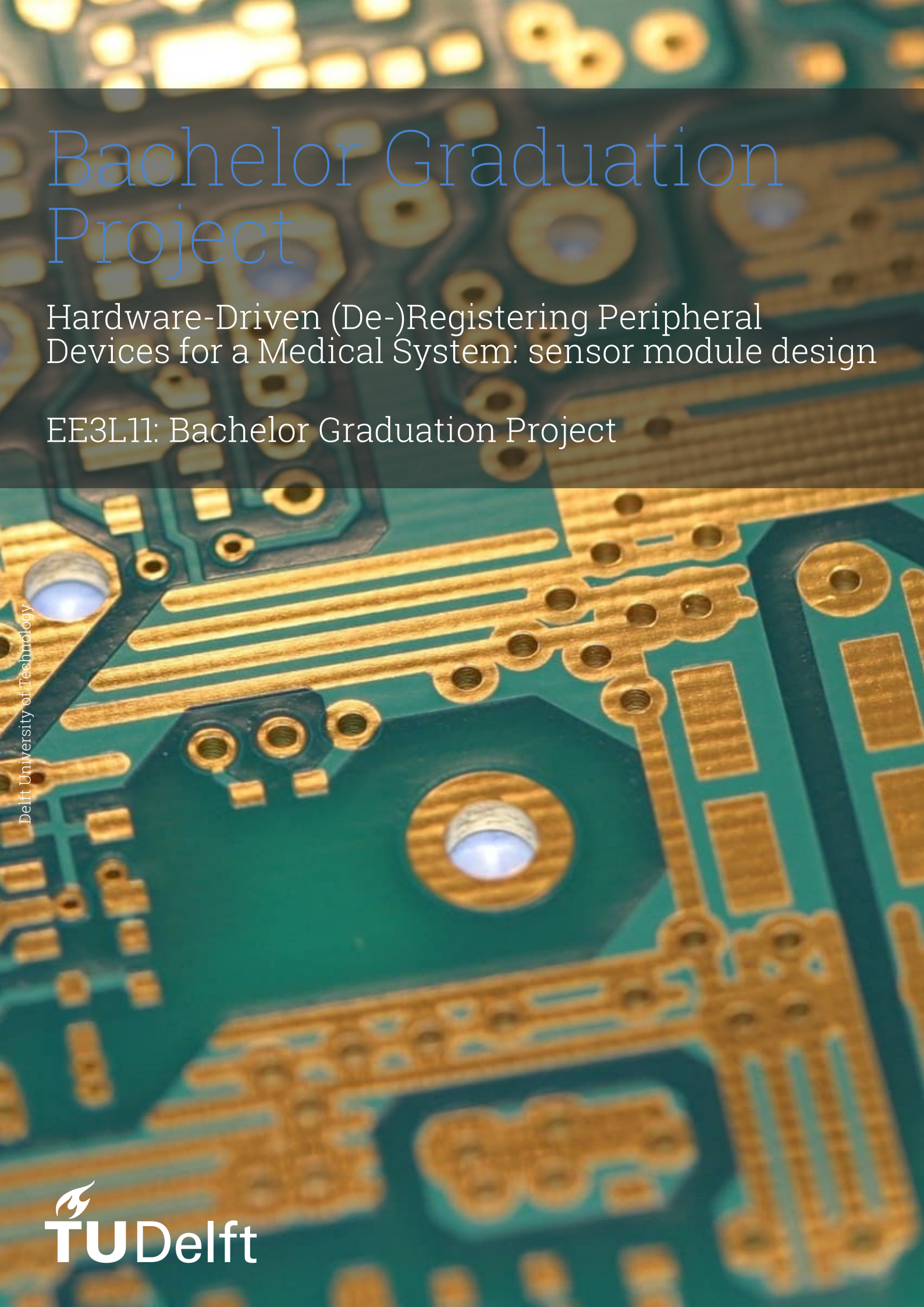


Bachelor Graduation Project

Hardware-Driven (De-)Registering Peripheral Devices for a Medical System: sensor module design

EE3L11: Bachelor Graduation Project

Delft University of Technology

TU Delft

Bachelor Graduation Project

Hardware-Driven (De-)Registering Peripheral Devices for a Medical System: sensor module design

by

Malik Fernald 5333180
Hugo Snelder 5491940

2026-01-19

Supervisor:	Paddy French
Supervisor:	Kianoush Rassels
Review Commission:	Massimo Mastrangeli
Project Duration:	October, 2025 - january, 2026
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

TU Delft

Abstract

This thesis presents the design and implementation of a sensor module used in a Remote Medical Monitoring System(RMMS). This RMMS aims to be unobtrusive and record the Activities of Daily Living(ADL). The proposed sensor module is built around an ESP32-S3 (Lilygo TQ-T Pro) and integrates all the sensors on custom PCBs. It includes interchangeable environment sensors(BME280 or ENS160 + AHT21), mmWave radar sensors(C1001 or HMMD) and a light sensor, with capabilities of powering an external ONVIF camera depending on the sensor module variant. This was done with embedded software written in Micropython using an asynchronous(uasyncio) main loop, automatic sensor detection, and secure data transmission using MQTT over mutual TLS. It also integrates a hardware-driven (de-)registration program developed in a companion project.

The system was validated through measuring the connectivity and timing delays of the system. The resulting MQTT round trip time of the sensor module is less than 100 milliseconds and the throughput is more than 700 bytes per second. The overall delay between sensing and receiving it over the LAN is less than one second. The casing tests showed that the thickness of the outer shell influenced the measurement results for the radar sensors. However limitations are present for the sensor module, due to unreliable "ghost" readings from the C1001 mmWave radar sensor. Overall a modular and deployable sensor module was created that can be integrated in a RMMS with validated performance of the communication and runtime delays.

Acknowledgements

We would like to thank prof. dr. Paddy J. French and ir. Kianoush Rassels for supervising our bachelor graduation project, for making time to meet with us, and for being available whenever questions arose. Your guidance and feedback helped us push our work further and develop as engineers.

We would also like to thank each other for the nice collaboration and the helpful insights we offered each other during this project.

We would also like to thank the subgroup responsible for the (de-)registration protocol for their support during the integration and generally their help throughout the project.

Finally, Hugo Snelder would like to thank his parents, sister, little brother, girlfriend and his cat Gilles for their continuous support and encouragement during his studies, especially during both challenging and rewarding moments.

Malik Fernald would like to thank his parents for their support, helpful insights and compassion. He would also like to thank his brother and little sister for keeping up his spirit and providing encouragement during more difficult stretches.

Contents

1 Introduction

1.1. Problem statement

The elderly population in the modern world keeps on growing. Meanwhile, birthrates are declining. This will cause the number of elderly people per person under 65 to increase in the future [1]. This in turn will lead to an increased demand for care and a rise of costs in healthcare for governments. This is not only a problem in the Netherlands, but a global one[2]. The increased access to good healthcare all over the world, and decreasing birthrates as countries keep on developing leads to less untimely deaths from preventable diseases. This demographic shift towards an older population is a major challenge for societies in the future.

In order to combat this issue the industry will move more towards Remote Medical Monitoring Systems(RMMS). These systems can combat the strain on medical services caused by an ageing population. Most products on the market are smart wearable devices which are powerful tools to use. However they can discomfort the patient and require the user to have good digital literacy[3]. In order for (older) patients to get on-board with RPM a less unobtrusive system should be created to monitor patients at home. These systems monitor a patient's Activities of Daily Living(ADL) at home, it does this without the patient having to interact with it. Which gives more freedom instead of being tied to a wearable device[4].

A RPM system consists of multiple sensor modules connected with one another through the Wi-Fi router. This interconnectedness via Wi-Fi is called the Internet of Things(IoT). When every room has such a sensor module with ambient sensors that track activity from a distance, then a complete picture of the persons ADL can be created and processed. This processing happens on a remote server, but that is outside the scope of this project. In the house there is also a Single Board Computer(SBC) placed that controls all the traffic, and monitors the system to check whether everything works correct. When a patient is distressed it will be registered through the data collected, and help will be send immediately to rescue the patient. That is the advantage of such a system. It will be monitoring constantly and must always works.

1.2. Thesis synopsis

This thesis will go over the design and implementation of the sensor modules described previously in section 1.1. These sensor modules are an integral part of the system, without it there is no data, no monitoring and thus no product. Multiple sensor modules have been designed but there is no good implementation or software backing them up to create a one package RPM system that just works. In order to realise this PCB's will be designed for different use cases. The software used will be implemented such that it will detect the sensors and run differently based on which sensors are put on the sensor module.

1.3. Structure of thesis

This thesis is divided in multiple chapters, first it describes relevant background research in chapter 2. After which, in chapter 3, it will go over modern implementations of medical monitoring systems and what this thesis contributes to that field. The requirements for the thesis and the final product is described in chapter 4. Chapter 5 describes the hardware used and why it is used, and this ties in well with chapter 6 which talks about the software implementation for the sensor modules. With all this information the design of the PCB can finally be discussed in chapter 7. The tests and its results will be

described in chapter 8. From the tests a final casing design is discussed in chapter 9. Afterwards chapter 10 will go over the costs of the final product. the results will be discussed in chapter 11. Finally, a conclusion will be drawn in chapter 12, this chapter will also go over the future work that can be done.

2

Background Research

2.1. Communication protocols for microcontrollers

In order to get data from the sensors to the microcontroller there should be a data exchange protocol in place. There are a couple standardised protocols developed for microcontroller application. The next sections will go over the protocols, their advantages and disadvantages.

The workings of the I2C protocol

The Inter-Integrated Communication(I2C) protocol is a communication protocol that makes use of two wires to enable communication between multiple devices. The protocol utilises a Serial Clock Line(SCL) and a Serial Data Line(SDA) wire. Each device connected to the I2C bus has their own unique address, such that each device can be addressed individually for their data. The controller controls the I2C bus by realizing the data transmission; generating the clock signal; having the addressed device be in receiver mode. The data rate of I2C depends on the mode it is initialized in, it goes from 100 $Kbit/s$(standard mode) and 400 $Kbit/s$ (Fast mode) all the way up to 3.4 $Mbit/s$ (high speed mode).[5][6]
The data frame for I2C consists typically of 20 bits. The first bit is generated by the transmitter in order to let the receivers know that data will be transmitted over SDA. Then the controller sends a 7 bit address in order to select the receiver (in total the transmitter can control up to 128 different receivers with a 7 bit address). It appends a read/write bit to let the addressed receiver know it wants to read its data, or write data to it. The receiver responds with sending an acknowledge(ACK) bit. When the ACK bit is received by the controller 8 bits of data will be transmitted from the receiver to the controller over the I2C bus or vice versa depending on the read/write bit sent previously. when it is received an ACK bit will be send, and finally a stop bit.[5][6]

The workings of the UART communication protocol

The Universal Asynchronous Receiver Transmitter(UART) protocol is a communication protocol that makes use of two wires to communicate with peripheral devices. These two wires are labelled Receive(RX) and Transmit(TX). These wires are connected respectively to the TX and RX of the connected device. Due to the asynchronicity of the protocol there is no shared clock signal, however there is a baud rate generator that ensures that the data is send and received at the same rate. When the baud rate of the receiver and transmitter do not match, the data will not be received correctly. The data transmission speed is controlled by the baud rate generator, a higher baud rate leads to a higher transmission speed. The baud rates used are multiples of 9600 bits per second(bps). The limitation of this protocol is that only one device can be connected at once, thus for each UART peripheral device connected two pins are occupied of your microcontroller's input/output(IO) pins.[7]
The data is send in frames, thus small packets of data bits at once. In order to start transmitting a data frame the transmitter sends a start bit over the UART, such that it notifies the other device that it will be transmitting data. When the receiver receives this start bit it knows that data will be send over the UART, and it will listen until a stop bit is send at the end of the data frame. In order to determine if the data send is correct there is also a parity bit that checks whether the send data has changed during transmission through checking if the summation is even or odd. After the data has been transmitted the UART bus will go idle waiting for the next start bit to be send.[8]

The workings of the SPI communication protocol

The Serial Peripheral Interface(SPI) protocol is a serial communication protocol that makes use of four wires to communicate with peripheral devices. It makes use of a Serial Clock(SCK) signal; a Master

Out Slave In(MOSI) signal; a Master in Slave Out(MISO) signal; and a Slave Select(SS) signal. The transmitter(Master) generates the clock signal and addresses the receiver(slave) through its SS signal by pulling it down. In order for this to work each connected peripheral device has its own SS wire while sharing the MISO and MOSI wires. When the SS signal is pulled down data is transmitted bidirectional. Only the connected receiver whose SS pin is pulled down processes the data send and sends data to the transmitter. This is called full duplex communication.[6][9]

2.2. The workings of the ONVIF protocol

The Open Network Video Interface Forum (ONVIF) protocol is a protocol widely used to allow interoperability between devices like IP cameras, network video recorders (NVRs) and management software. ONVIF uses profiles to organize different features. These profiles can be seen as different levels of support. Each profile adds specific functions. There are five main profiles: Profile-S, Profile-G, Profile-T, Profile-M, and Profile-A.
Profile-S is the most commonly used profile. It handles basic video streaming, allows you to move the camera if it is capable of Pan, Tilt and Zoom(PTZ) and it sends basic alerts.
Profile-G is capable of recording to an NVR and allows for the search, save, and playback of recorded video material.
Profile-T has improved video quality and adds better motion detection. It also supports newer video compression formats like H.265, which helps to save storage space without compromising on the quality of the videos.
Profile-M supports AI-powered functions such as facial recognition, object detection or smart motion tracking.
Profile-A is used for security systems that control doors, keypads, and card readers and makes sure that the cameras can communicate properly with a building access system.

Although ONVIF enables devices from different manufacturers to communicate, it does not include built-in mechanisms for fault tolerance or automatic failover when a device or network component fails.

2.3. Micropython

Micropython is a lightweight version of the Python 3 programming language. It is optimised to run on microcontrollers. In order to achieve this a lot of modules standard for Python 3 are not supported making it possible to run Micropython within just 16k of Random Access Memory(RAM) and 256k of codespace[10]. Micropython supports a plethora of different microcontroller paltforms to be run on i.e. ESP32, pyboard, Raspberry Pi RP2xxx, NXP i.MXRT 10xx etc. The small amount of modules implemented in micropython are specifically chosen for microcontroller applications, thus there are also packages present for accessing low-level hardware of the microcontroller. Micropython is also a good way to rapidly prototype and develop IoT devices with well supported peripherals. The development is also cross platform, thus the developed software of the IoT device can be implemented on multiple microcontroller platforms. This creates versatility and adaptibility needed during the prototyping phase.[11]

2.4. MQTT

The Message Queuing Telemetry Transport(MQTT) protocol is a data transfer protocol with a central broker server, henceforth called broker, that uses a publish/subscribe model for sending messages to and from connected clients. A client connected to the broker is decoupled from all other connected clients, because every message goes through the broker. In order for a client to receive messages from the broker it should show its interest by subscribing to a topic, this client is called a subscriber. When a client wants to share information with interested subscribers it should publish the information to the broker. When subscribing and publishing clients specify which topic they want to publish or subscribe to. The broker sees to it that all data traffic gets from the publisher to the subscriber.[12] The advantages of MQTT are mainly that it is open-source, lightweight, consumes less power and a lot of devices can be connected at once without clients knowing of the other clients presence. It also supports Quality of Service(QoS), QoS is a protocol that ensures the delivery of messages. There are three levels of QoS defined by MQTT. These are:

- QoS 0: The message is send once and the broker does not send an acknowledge if the message has been received. It is a "shoot and run", because there is no guarantee of delivery.
- QoS 1: The client will broadcast the same message multiple times, until it receives an acknowledgement from the broker that it has received the message.
- QoS 2: This implements a four-way handshake for ensuring message delivery. This causes it to absolutely ensure the message has been received correct, but it is also the slowest.

Compared to other messaging protocols it is widespread and easy to implement, and for IoT applications considered the best fit. Yet, the only downside is that it has no security features build in.[13][14]

2.5. TLS

The Transport Layer Security(TLS) protocol is a communication security protocol designed for the communication between a client and a server. Such a security protocol is implemented such that no one can eavesdrop on or alter the messages send between the client and the server. TLS is used in combination with a data transfer protocol to encrypt the data. The protocol works as follows: The client connects to a server and asks for its certificates. The client verifies the certificate by checking if it has been created by a familiar certificate authority. The connection ends if the certificate is incorrect. The client and server agree on session keys to use for encrypting the data. The server can also verify the client by checking its certificates[15]. This is called mutual TLS (mTLS). In order for the verification process to work the client needs to have the client authority certificate from the client authority that created the certificates of the server. In order for mTLS to work the client needs to also have certificates created by the same client authority as the ones of the server. The encryption algorithm used for the keys and the certificates is not predetermined, thus the security of the protocol can be easily updated when needed[16]. A communication security protocol is needed for IoT applications since these function on the Local Area Network they are prone to attacks. The integration with MQTT is a common usecase for IoT applications that need security. Yet, Paris et al. showed that the power consumption of MQTT with TLS on IoT devices is still a lot more than without TLS, and the system overhead required is a lot more with than without it, but when the message per minute rate is low this is somewhat mitigated[17].

3

State of the Art Technologies

The medical sector is an innovative field which embraces new technologies constantly in order to make the process of personal medical care more efficient and user friendly. This innovation is important to continue to meet demand in the future. Currently, there are a plethora of technologies used to measure a persons health, even without the use of large medical equipment. With current innovations, health monitoring can be done without going to a hospital. This shows signs of people moving towards personal health monitoring, and the hospitals will follow this up with RMMSs. This chapter will go over these current personal medical systems and devices and research into RMMS. It will also discuss the current gaps in research and what this thesis will add to the current field.

3.1. Medical personal sensing technologies

Currently RMMSs already exist, but there is a big difference between research and commercial products. Nowadays many in use RMMSs rely on the active participation of the user i.e. wearing a device or logging information from a device at home. Current research is looking into Passive Remote Monitoring(PRM): sensors placed in the home that detect activity patterns, mobility and some vital signs, without interaction when installed. This is to help the staff with care for nursing homes or in home care.

Passive monitoring of patients

Most PRM systems designed for peoples homes integrate multiple sensors to detect abnormalities in a persons daily activities. These sensors in the sensor network can include motion sensors, door sensors, bed sensors, environment sensors and sometimes cameras. Such sensor network collects data, and sends it to a remote server that processes all the data.[18][19][20] Donelle et al. [21] researched the effectiveness of RMMS to reduce the risk of being admitted to higher level care. It did not show any differences but indicates that it is an idea that is being integrated and evaluated for real-world application.

Elderly people can encounter many risks during the day. When they have an accident at home the most important thing is a fast response time. There are a lot of differing ideas of how such a RMMS should work. A vision-based fall detection was researched that implemented AI to detect falls and alert staff. This system reduced the response time significantly [22]. Other studies focus on the detection in and around the bed specifically. The INBED system is a system for bed-exit detection that is a concept for complete fall-prevention with detection and alerts for patients in the geriatric ward[23].

Such bed systems also include many sleep sensing technologies to track sleep health. A recent review paper looked into current unobtrusive bed monitoring methods. It showed how sensors in the bed can track sleep, activity, heart rate and breath rate. It concluded that the field remains promising, but that instruments with clinically acceptable accuracy are still in the future [24]. Strauven et al. [25] showed an example of an application for bed sensor technologies that can help the nursing home staff in regards of their core tasks by differentiating states the patient can be in during the night time.

Other vital sign technologies handy for RMMS's are camera based sensing technologies. The Oxehealth Vital Signs device is software that implements camera based footage with an near field infrared illuminator to take spot measurements of a persons heart rate, breath rate and pulse in a controlled environment. The measurements only work when the person is laying or standing still.[26][27]

Camera based detection systems are often subject to privacy concerns and a more privacy friendly alternative is millimetre wave(mmWave) technology for detecting vital signs of individuals. These

sensors can detect (micro-)motion without needing a photo of the sensed area. A recent review paper by Wu et al.[28] showed the current pipelines used in mmWave sensors and the challenges still ahead of accurate detection such as subject motion, multi-person separation and deployment constraints.

Current RMMS systems

There are currently some systems on the market with the discussed technologies. Vayyar Care has brought an imaging radar for fall detection to market that works in varying conditions[29]. The Care@Home system from Essence SmartCare is a system that incorporates multiple sensors into a network to detect the ADL of a person with interaction from the user[30]. SafelyYou created a fall detection system that uses AI video to detect falls and act accordingly[31]. From these systems a common design language can be extracted that is useful for a RMMS. There should be multiple sensors distributed in an area that detect different events. These events should be send and anomalies should be detected, and when such anomaly is detected a notification should be send to indicate that someone needs help.

3.2. Gaps in the research

The problem with the current systems are that they are mostly experimental and require interaction. This is what the system implemented in this paper should overcome. It will go over the implementation and development of a sensor module used for a RMMS that utilizes environmental and non-invasive personal sensing methods. The other problem is that research is not focused on upscaling and deployment. This paper will look at how such a deployment implementation can work with an implemented onboarding protocol on the sensor modules. This thesis will also go over scalability with developing and easy to deploy sensor module for rapid system scaling and sensor modularity. It will also address privacy concerns raised by implementing secure data transmission for sensitive personal data. It will look into the feasibility of such a system on the provided hardware.

4

Programme of Requirements

The eventual goal of this project is to design a small system. This system is part of a larger system in which it, of course, needs to work reliably. This overall system will not be considered, since the system designed in this project is able to act fully on itself. So, this programme of requirements will mainly consider requirements on one of the subsystems installed at home. It will consider how the printed circuit board(PCB) should function; what components it needs to have, and the overall form factor of the system.

1. The following sensors should be able to connect to the PCB:
    - BME280 (pressure, humidity and temperature sensor)
    - ENS160 + AHT21(environmental sensor)
    - WaveShare HMMD mmWave sensor(Human Micro-Motion Sensor)
    - DFRobot SEN0623 C1001(mmWave Human Detection Sensor)
    - A sensor for light detection
    - camera only for power(onvif profile T, IR, length of max 7.5 cm)
2. the resulting system must work.
3. the system should be responsive from data acquisition to the broker receiving the payload should be below 1 second.
4. there should be indicator LEDS to signal power on/off, Wi-Fi connection and camera on/off.
5. The system must have a physical button to reset/reboot the microcontroller.
6. A speaker, light and microphone should be part of the camera.
7. The size of the end product should be able to fit in your hand.
8. The PCB design should work and be correctly integrated with the code of the sensor module.
9. The final product should integrate into the current system.
10. The device registration should be implemented according to the provided registration scheme.
11. The sensor module should be able to receive commands from the SBC, and act accordingly.

5

Hardware for the RMMS's Sensor Module

The hardware that was chosen for the final design consist of multiple components in multiple categories. Most of the components were provided to us by our supervisors, yet the choices they made were adequate and are the best match for the final product. The hardware consists of the following categories: micro-controller, sensors, camera and others(LEDs, resistors and transistors). First, an overview of all the components will be given in this chapter. Afterwards it will describe the physical features and use cases of the hardware that need further explanation.

5.1. Overview

- Microcontroller:
  - Lilygo TQ-T PRO esp32-s3 4mb flash 2mb PSRAM[32]
- Sensors:
  - WaveShare HMMD mmWave sensor (Human Micro-Motion Sensor)[33]
  - DFRobot SEN0623 C1001 (mmWave Human Detection Sensor)[34][35]
  - BME280 (pressure, humidity and temperature sensor)[36][37]
  - ENS160 + AHT21 (environmental sensor)[38][39]
  - GL5516 (photoresistor)[40]
- Camera:
  - HQCAM TOP Store 5MP
- Miscellaneous components:
  - Resistors: 4x 330Ω, 2x 10$k\Omega$, 1x 1$k\Omega$
  - Capacitors: 1x 330nf, 1x 100nf
  - Transistor (NPN, rated for atleast 2A)
  - L7805 voltage regulator[41]
  - Light Emitting Diodes(LEDs)

5.2. Microcontroller

The microcontroller used to control the sensor module is the Lilygo TQ-T PRO. This microcontroller makes us of the ESP32-S3 chipset. This is chipset has a high versatility and is well documented. What makes the Lilygo TQ-T Pro easy to work with is its small size. The dimensions(Width X Height X Length) of the microcontroller are: 18mm x 14mm(9mm without pins) x 33mm. An overview of the dimensions can be seen in figure C.1. The pinout of the device can be seen in figure C.2. This shows that the microcontroller can supply 5V and 3.3V, and it has 11 General Purpose Input/Output(GPIO) pins available for use, and a ground pin. There are enough GPIO pins for our use case. The microcontroller also has an Anolog-Digital Converter(ADC) built in.

5.3. Environmental sensors

The environmental sensors used for the sensor module are the BME280 and the combination of the ENS160+ath20 sensors on one board. These sensor boards both measure temperature and humidity of the environment. The BME280 is the simpler sensor that only reads pressure next to the previously stated values, while the ENS160+ath20 measures air quality(CO2: ppm, TVOC: ppb, AQI) on top of temperature and humidity. These sensors are interchangeable on the PCB because all their pins exactly line up with each other, and they both use 3.3V. They communicate both through the I2C bus with the microcontroller. The code implementation is described in section 6.5.

5.4. Radar sensors

The radar sensors used for the sensor module are mmWave sensors that make use of FMCW to sense the person in the room. The WaveShare HMMD mmWave sensor is a generic presence and movement detection sensor that will be used for the modules in the generic rooms i.e. living room, study room and kitchen. It uses a 24GHz wave to determine a persons presence and their movement, yet the wavelength is too long in order to determine other factors like heart or respiratory rate. This is not implemented for these rooms because most falling accidents happens in the bedroom, bathroom or by the stairs[42], and to keep the cost of the overall system lower. For the bed- and bathroom the DFRobot SEN0623 C1001 sensor is used. This sensor can detect a persons presence, but also their heart and respiratory rate. The 60GHz wave used for detection has a resolution that can detect the small changes of the chest in order to determine the heart rate and respiratory rate of the person being detected. The sensors both use the UART protocol to communicate with the microcontroller, while both are capable of using SPI UART was chosen because only one sensor is connected to this bus at a time, thus making the PCB design less complex. They both can connect to the same data pins on the microcontroller, and how this is done is explained in section 6.6. The WaveShare HMMD mmWave sensor uses 3.3V, while the DFRobot SEN0623 C1001 uses 5V for power.

5.5. Camera

The camera module used for this project is a 5MP camera from 'HQCAM TOP Store' on AliExpress. The camera has a 3.7mm lens with an angle of about 85 degrees. The total size of the module is 38x38mm. It supports the ONVIF protocol, as well as the H265 compression algorithm. It has motion detection with up to 4 independent detection areas. It can be connected to the internet over WiFi as well as via an Ethernet cable. The camera has a 1 CH input for a microphone and a 1 CH input for headphones or a speaker. The power supply is DC12V, 2A and the power consumption is 1.5 W.

5.6. Other hardware of note

There are a couple of components on the board that do not need their own section to describe their function and their functionality. These components will be discussed in the coming short subsections.

5.6.1. Light sensor

The GL5516 [40] is a photoresistor which is implemented on our PCB in order to extract the local light intensity. This is done to determine if the lights are turned on or off. This helps with measuring the patients ADL. The circuit can be found in figure 5.1. The photoresistor is a non-linear component for extracting light, and measurement is done to determine the resistance over the photoresistor instead of directly measuring the light intensity. The light intensity is then extracted from the resistance according to the following formula 5.1. This formula was extracted from the sensors documentation using power laws with $\gamma \approx 0.5$ [40]. The code implementation is described in subsection 6.7.

$$L = 10\left(\frac{R_f}{R_{LDR}}\right)^2 \tag{5.1}$$

5.6.2. Voltage regulator

In order to power the camera and the microcontroller on the Bedroom/Mirror PCB(described in section 7.3) from the same voltage source, a voltage regulator was needed such that the microcontroller gets

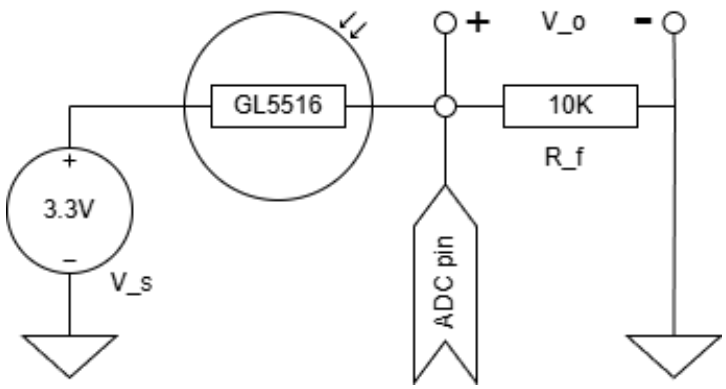


Figure 5.1: Figure showing the circuit for sensing the lux value with the GL5516 light sensor.

its desired 5V DC input and the camera gets its desired 12V DC input. The L7805 voltage regulator is used to regulate the 12V DC input received from the wall adapter. The input can be the same 12V DC source that is used to power the camera. The output will be converted down to 5V, which is the voltage needed to power the microcontroller and via the microcontroller the other components can be powered. The L7805 is chosen specifically because it can handle the current that is necessary to deliver sufficient power to the other components. In comparison to more commonly used voltage regulators can handle the voltage, but cannot deliver a high enough current.

6

Embedded Software Design for the RMMS's sensor module

The code used on the Lilygo TQ-T Pro microcontroller is written in Micropython. This is an efficient implementation of the python 3 programming language. It is specifically optimized for microcontrollers with its limited memory. Due to this it only includes a select subset of the python 3 libraries.[10]

6.1. System overview

The sensor module should implement two different programs: registration and deployment. The registration program should work according to the registration protocol provided by the other sub-group[13]. The implementation of the registration is described in section 6.4. The implementation of the deployment script is described in section 6.3. An overview of the entire systems and the implemented modules is shown in figure 6.1.

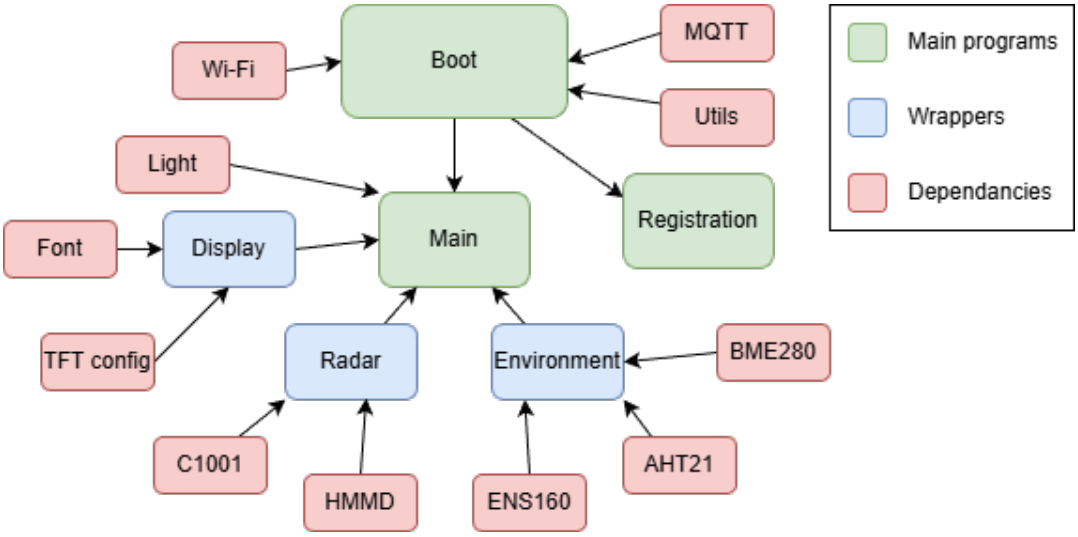


Figure 6.1: Figure showing a system overview of the software on the sensor module.

6.2. Boot

The boot software are the first lines of code read by the processor, thus it is used for initializing all the processes on the microcontroller. When the microcontroller boots up it loads the configuration from the config.json file, this file contains all the information for the boot up process like the Wi-Fi configuration. Two examples of the configuration file can be found in appendix B. After loading the data it connects to the Wi-Fi network loaded from the configuration file. When the device is connected the Wi-Fi LED is turned on in order to show the user that the device is connected. It now scans the network for an UDP broadcast in order to get the server address of the broker. Doing an UDP broadcast is like shouting, everyone can hear it and it is unencrypted. When it has received the address it scans the list of labels that the SBC connects to. If the sensor modules label is not present in the send list the

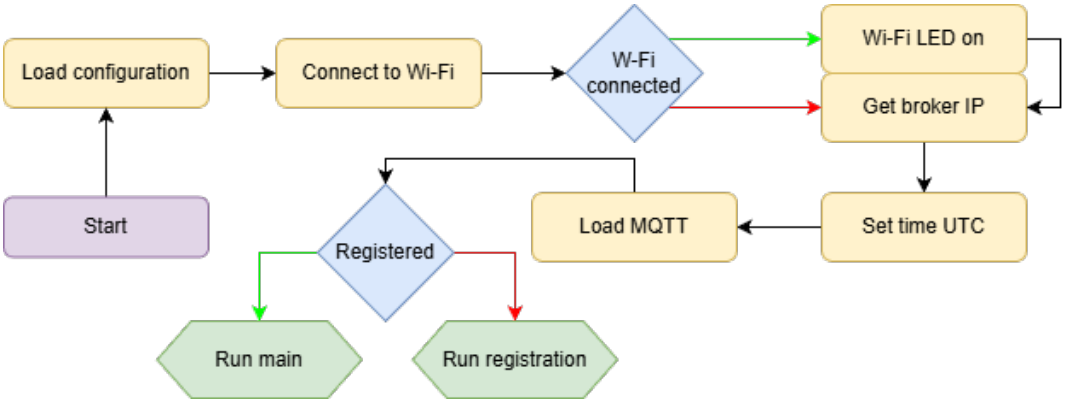


Figure 6.2: Flowchart that shows the boot cycle of the sensor module.

sensor module resets. Now that the microcontroller has the correct IP address of the SBC it should connect to a Network Time Protocol(NTP) server in order to set the local time on the microcontroller to UTC. It does this such that the Client Authority(CA) certificate is valid for the Transport Layer Security protocol set in place, further explanation of how this works can be found in section 2.5. When the local time is correctly set, the microcontroller loads its MQTT configuration for sending data to a specified server on the Local Area Network(LAN). The device checks itself if it is registered by checking the registered value loaded from the configuration. If the device is not registered in the database, it will enter the registration sequence described in section 6.4. If it is registered, it will run the deployment program of the system, also known as the main program described in section 6.3. A flowchart of the whole boot sequence is shown in figure 6.2.

6.3. Main

The main program of the sensor module handles its core functionality. It combines all the different functionalities of the sensor module during deployement into a main loop. This loop consists of multiple coroutines that are handled by the uasyncio library. This is implemented because the microcontroller waits a long time on Input/Output(IO) bound tasks, and with this library it can execute a different code piece when it is waiting for an IO bound task. The main loop consists of the following coroutines: wifi, clock sync, sensor data, MQTT send, MQTT callback, MQTT connection and MQTT receiver.

Before the main loop is run the microcontroller needs to initialize the connected sensors and display. It starts by initializing the I2C for the environment sensor(s). The microcontroller will then initialize the environment sensor(s), further explanation can be found in section 6.5. Afterwards the radar sensor is initialized; this progress is explained in section 6.6. The connected sensors are then saved in the configuration file of the sensor module. When this is done the display will be initialized, the functionality and features are described in section 6.9. The sensor module connects to the MQTT server, and publishes that it is connected. This is done so the SBC knows that a sensor module is connected before it receives sensor data. The whole main startup process is shown as a flowchart in figure 6.3.

WiFi coroutine

The WiFi coroutine controls the WiFi connection of the sensor module. It checks whether the sensor module is connected to the WiFi network, and reconnects if it is not connected. It also handles the WiFi LED on the sensor module and turns it off when it is not connected and back on when it is reconnected.

Clock sync coroutine

This coroutine handles the synchronization of the Real Time Clock on the sensor module. It synchronizes the RTC every hour with the NTP server. This is needed because the RTC can drift over time.

Sensor data coroutine

This is actually the main coroutine of main event loop. It reads the changed sensor data from the light sensor and both wrappers for the radar and environment. It packages them into a dictionary and checks

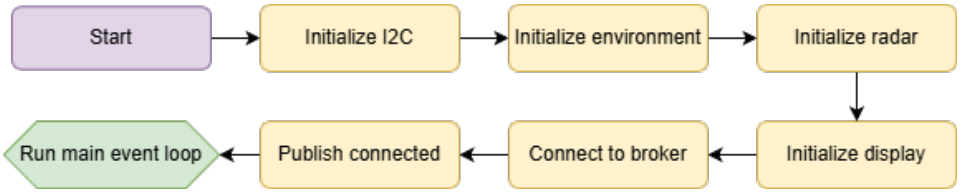


Figure 6.3: Flowchart that shows the start up of the main program of the sensor module before the main event loop.

whether the data has changed from the previous time a change had occurred. When the received data is different a timestamp based on the RTC is added to the dict and the data is put into a queue for it to be send to the SBC with the MQTT publisher coroutine. In order to do this it uses the helper function build payload that builds the payload based on only changed sensor data.

MQTT publish coroutine

The MQTT publish coroutine goes hand-in-hand with the sensor data coroutine. This coroutine gets the payload that sits the longest in the sensor data queue and pops it from the queue and sends the payload via MQTT to the SBC with the MQTT publish function. This design was implemented in order to relieve the sensor data task from having many IO tasks that lag the system. This implementation lets the system have a sort of buffer that stores data when the changes are too fast to send immediately.

MQTT connection coroutine

The MQTT connection coroutine handles all things with the MQTT connection to the correct server. It reconnects to the server when the connection is lost. It also resubscribes to the right topics when it has reconnected to the server.

MQTT receiver

The MQTT receiver coroutine frequently checks if any incoming messages where received on the subscribed MQTT channels.. When an incoming message is received. The MQTT callback will be called, by the MQTT package not in the code, in order for handling messages.

MQTT callback

This coroutine starts when the MQTT callback is activated. This coroutine exists because data parsing takes a long time to run, thus blocking a lot of other processes when it is not a coroutine. This callback functions reads the received messages and acts accordingly based on the messages received. All the functions implemented and the according actions the sensor module does can be found in table 6.1

Table 6.1: Table with all the functions the sensor module can receive from the SBC, and the according actions of the sensor module.

Received function	Action performed by the sensor module
ping: True	Send "Connected: True" to topic deployment/[device label]/data.
action: deactivate	Stop sending sensor data to the SBC.
action: activate	Start sending sensor data to the SBC.
action: deregister	Disconnect from the WiFi network and the MQTT server, remove the values for registered, label and broker in the configuration file and reset the sensor module.
action: deregister_sbc	Disconnect from the WiFi network and the MQTT server, remove the values for broker in the configuration file and reset the sensor module.
shutdown: True	Disconnect from the Wi-Fi and sever the MQTT connection and remove the values for broker in the configuration file.

6.4. Registration

The protocol subgroup has written the Micropython registration script, and it was integrated by the PCB subgroup. The registration protocol was designed by the protocol subgroup, and this section will

be heavily inspired by their work. If you want more in depth analysis of all the steps and how the detection algorithms work you can read their thesis[13].

The function of the registration script is to register the sensor module in a database. This is achieved by giving each sensor module a QR-code in which a label is encoded. The reading of this physical QR-code on the sensor module is done during the registration process by a camera that has multiple sensor module's QR-codes in his line of sight. All the QR-codes are then read by a camera. The labels are send to each sensor module using MQTT with TLS. The detection of the correct sensor module is done by using a blinking LED light. The whole registration process will be described in the following subsections.

Connecting and temporary names

The sensor modules connect to the correct broker, the one they have gotten during the boot(section 6.2), and wait for a send request. In this initial send request the sensor modules send their UUID. When the UUID has been received by the server, a temporary name will be send back from the server to the sensor module. When the temporary name has been received the sensor module reconnects to the server with the temporary name as its client ID. The sensor module also subscribes to a new topic based on this temporary name. Now every sensor module has its own topic it can subscribe to, and the server can differentiate every sensor module based on their temporary name. A check-up is sent in order see if the connection has been made correctly.

Receiving label and client certificate

Now that both the server and sensor module have reached unanimity on the names of each sensor module the actual registration can begin. The server will send a nonce(number only used once) to the first sensor module in its database. This sensor modules starts blinking its WiFi LED at 5 Hz. It also sends the nonce back to the server. The nonce is used to verify the sensor module with the blinking LED. Meanwhile, the blinking LED is detected in the frame of the camera and the corresponding QR-code is read from the subframe based on the location of the blinking LED. The server knows which sensor module is blinking based on the request it has send to the temporary name. The server also knows the corresponding label based on the LED and QR detection. With all these pieces in place the label and the corresponding client certificate is send to that particular sensor module. It turns its WiFi LED off now, and the sensor module disconnects from the server.

Verification

The sensor module connects to the secure port of the server with its received client certificate, and it subscribes to the topic of its label. The server sends a final check-up to the sensor modules topic and when it receives the correct information it sends a command to save and shut down. The sensor module then saves its label and UUID in the configuration file, the value registered is set to "True" and the device resets itself.

6.5. Environment class

The environment class is a wrapper for the BME280 and the ANS160/AHT21. It does three things. It detects the connected sensor and initializes the correct class e.g. It detects the BME280, and then initializes it using the BME280 class. It also handles all the data reading from the sensors, thus it packages all the data from the sensor to the output of a single function.

Initialization of the environment sensors

The environment sensors are on the I2C bus of the microcontroller this bus is already initialized in the boot (section 6.2). The initialization function checks if the I2C is initialized otherwise the communication does not work. The function also checks if there is already an environment sensor specified in the configuration file of the microcontroller. If so, it tries to initialize that sensor immediately. If none is specified the microcontroller scans the I2C bus, and based on the received addresses on the bus it initializes the correct sensor. This was implemented in order to have a faster boot time when the system is reset.

Environment sensors data reading

The reading of the data makes use of two helper functions. One to round the data and one that checks if the data has changed from the previous value. If it has changed it is added to the output dictionary of the function. The read function first initializes this dictionary only with the source of the data, the source is the sensor that was initialised during the initialization phase. When the dictionary is initialized one of the outputs will be read from the sensor, and rounded by the rounding function. The rounding unit for each value is specified in the configuration file. When the value is rounded. It will be compared to the last value that it had when the value was requested from the sensor. If it has changed it will be added to the output dictionary as [unit]: [value]. This process is repeated for every value that the sensor can sense. When all the values have been read the dictionary will only be output if it includes at least one of the values the sensor can read. When the output dictionary only includes the source the dictionary only "None" will be passed.

6.6. Radar class

The radar sensors have the same hierarchical structure as the environment sensor, but they communicate through UART with the microcontroller. This communication protocol is used because only one radar sensor is connected at once, thus we need less wires compared to SPI making the PCB simpler to design. More information can be found in section 2.1. In order to detect the connected sensor there is a radar wrapper. Because only one device can be connected at once, through the UART, there are no individual addresses to scan. The radar wrapper has the same three functions as the environment wrapper: it detects the sensor, it initializes that sensor and it reads all the data from that sensor.

Initialization of the radar sensor

The initialization of the radar sensor is different than that of the environment sensors since the UART communication protocol does not use addresses like the I2C protocol, thus the UART bus cannot be scanned like an I2C bus. First the microcontroller checks if there is a radar sensor specified in the configuration file. If so, the microcontroller immediately initializes the specified sensor. It checks the sensor connection by requesting data. If no data is received from the sensor the microcontroller tries to probe the radar sensor with initialization commands of the possible radar sensors. First the HMMD and then the C1001. If one of them gives a response the initialization is complete. The source will be saved based on the type of sensor that has responded to the initialization command.

Radar sensors data reading

The data reading of the connected radar sensor makes us of the same structure and principles as the environment sensor. It starts out with a dictionary that only includes the source of the data. It then requests data from the sensor and checks if the data has changed from the previous send value, it uses the same helper function as the environment sensor to achieve this. If it has changed, the changed value will be added to the dictionary as [unit]: [value], if not it will be discarded. When the output dictionary only contains information of the source, the output of the function will be "None". Otherwise a dictionary will be returned with only the changed values.

6.7. Light sensing code

The light sensor uses the ESP32S3 integrated Digital-Anolog Converter(DAC) to sense the voltage level over the series resistor of the GL5516. This is a 12-bit ADC that uses a total of 4096 levels to quantize the voltage level in order to extract the actual voltage the ADC output is converted to voltage according to formula 6.1. Now with the voltage across the series resistor calculated the resistance of the LDR can be calculated according to equation 6.2. Finally the lux value can be calculates using equation 5.1.

$$v_o = \frac{ADC\_output}{4096} \cdot v_s \tag{6.1}$$

$$R_{LDR} = R\left(\frac{V_s - V_o}{V_o}\right) \tag{6.2}$$

6.8. Code for MQTT connection

The MQTT manager implements a class for the MQTT connection, disconnection, publish/subscribe and callback. It uses the umqtt.simple package designed specifically for micropython and microcontroller applications [43]. This package has all the functions needed to connect, disconnect, publish messages, subscribe to messages and create a callback function.

Connecting and disconnecting

The connect function is a function to which the broker address is passed and it connects to that specific broker on the local network. The function implements connection to two different ports on the server. For mutual TLS port 8883 is used; in order to correctly connect using mutual TLS the correct certificates need to also be passed to the server in order for the mutual verification to work. Port 1884 is also implemented only for server side verification. This port makes use of a password and username to connect to the broker. The registration process uses both connection methods port 1884 for initial connection and registering the device and port 8883 for final verification. The main loop of the microcontroller always uses port 8883 to connect to the broker. When the correct parameters are passed to the function it loads the certificates and creates the Secure Socket Layer(SSL) parameters for the connection. These SSL parameters set the cryptographic constraints i.e. they set the requirements for the connections. The SSL parameters are different depending on which port is used, for mutual TLS more certificates are required thus more parameters are defined. Now when all the correct parameters are set and the correct certificates are loaded the sensor module can connect to the broker. When an error occurs when trying to connect the sensor module retries to connect. In total it will try to reconnect 5 times. When connected the sensor module sends a short message to the server every minute and gets a response in return. This process is called pinging. When the sensor module does not receive anything when it pings the server it sets the connected parameter to False. This parameter is checked in the main loop to check if the sensor module is connected. When the sensor module wants to disconnect it immediately disconnects from the broker.

Sending and receiving messages

A function is implemented to send sensor data in the JSON format to the correct topic based on the specified topic that is passed to the function. The publish function in umqtt.simple accepts only utf-8 encoded data, thus the sensor data and topic are encoded to that format and the sensor data is send afterwards. The sensor module needs to subscribe to a topic in order to receive data from the broker. The subscribe function implements this by using

6.9. Display code

The display used on the Lilygo TQ-T Pro uses the SPI bus to communicate with the microcontroller. The display itself uses a page system that can be cycled through. It loads all the required data from the sensors and the microcontroler to display to the end user. In total there are four pages that can be cycled through with the front facing buttons on the microcontroller. The pages display the following:

- **Page 1:** practical information like the location of the device, sensor id, wifi status and version.
- **Page 2:** the current time and data.
- **Page 3:** motion, presence, heart rate and breath rate(last two only for the C1001 sensor).
- **Page 4:** all environment measurements.

The update of the pages is handled through an interrupt handler that only updates the display when one of the scroll buttons is pressed. The S3LCD module handles the driving of the display and should be included on the firmware for the display to function.

7

Design of the PCB

An important part of developing the hardware for a peripheral device like this is the Printed Circuit Board (PCB) design.

A good PCB design facilitates the assembly of the final device, allows for a more compact and space-efficient product, allows for more feasible mass production, ensures more reliable electrical connections, makes maintenance, repairs and debugging easier, and makes the overall product sturdier and more mechanically stable. [44][45]

7.1. Requirements

A good PCB design starts with ideation and planning, followed by several concepts and prototypes. Regularly checking the design for potential errors and making improvements where necessary ensures a high-quality product.

For this specific device, the PCB design should meet a few requirements:

- The PCB should be optimized for the smallest area possible. This ensures that the final product is as small as it can be feasibly be, which allows for more widespread use.
- The components installed on the PCB should be positioned in such a way that the sensors and peripheral devices can be used without restrictions.
- The final design should be comprehensible and easily adjustable in the event that a different developer needs to make changes to the device.
- The final product should be as cost-effective as possible in order to make the product more accessible to clients with a lower budget.

7.2. Preparation

Before the actual design process begins, it is good practice to make a clear list of the components that should be included on the PCB along with their dimensions, pin layout, and any other potential special characteristics.

A comprehensive list of the components has already been given in chapter 5. To elaborate on that, a list of dimensions of the largest component is given here to give a better picture of the real-estate needed to house every component on the PCB. The dimensions of every component can be found in table 7.1.

Table 7.1: Dimensions of The main components on the PCB.

Component	Length	Width
Lilygo TQ-T PRO	32 mm	18 mm
WaveShare HMMD	20 mm	20 mm
DFRobot SEN0623 C1001	22 mm	22mm
BME280	15.3 mm	11.6 mm
ENS160 + AHT21	16 mm	11 mm
Camera	38 mm	38 mm

By far the largest component is the circuit board containing the camera module, so in order to save

space and allow for more flexibility during the distribution of space in the design process, this module will not be directly mounted on the PCB, but will be incorporated in the casing.

7.3. Area Planning

After taking a quick glance at the dimensions listed in table 7.1, it becomes clear that trying to fit all the components on a single PCB will result in a final product that is impractically large. For this reason, it is better to split the PCB into smaller boards with a more specific function rather than having one big board that is modular and universal. In this way, the overall size of the PCB can be reduced while still maintaining the desired function.

For this specific project, it is best to have three different PCB designs: one generic PCB that will be used in most of the rooms, one PCB that will be used for both the devices that will be mounted above the patient's bed and behind the mirror in the bathroom, and finally one PCB that can be used by surgeons and incorporated into a wearable device like a headband.

To be able to make an accurate estimation of the eventual sizes of the PCBs, it needs to be clear which component should be on which board. To figure this out, the function of each different kind of device should be listed.

Generic PCB

The Generic PCB will be inside the device that will be present in most of the rooms in the patients house. This device should include the microcontroller, since this is the 'brain' of the device and handles all computing and communication. In addition to this, the device should also include the BME280 or the ENS160+AHT21. These devices will be used to acquire information about environmental factors such as room temperature, CO_2 levels, and air quality. Finally, the device should be equipped with the HMMD mmwave sensor to determine the position of the patient within the room. All of the other components can be omitted for now for the sake of simplicity, since they do not play a significant role in the layout planning due to their relatively small size.

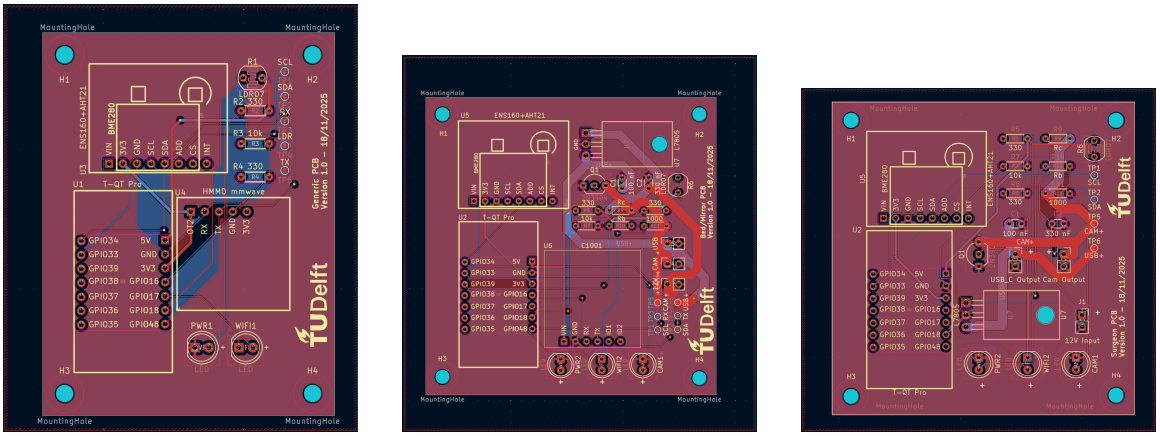
Bedroom/Mirror PCB

The Bedroom/Mirror PCB will be inside of the device that will be attached directly above the patients bed, as well as behind the mirror in the bathroom. Similarly to the Generic PCB, this PCB will have to include the microcontroller for the same reasons as stated before. The main difference between this PCB and the Generic PCB is that this PCB has a connection for the camera module. In order to supply sufficient power to the modules, there also needs to be an input for a 12V 2A power supply, since the camera module demands this. However, the microcontroller runs on 5V, so to supply power to that component, the 12V supply needs to be converted to a 5V supply. In addition to this, the Bedroom/Mirror PCB has the more accurate C1001 mmwave sensor. This sensor has a pinout that differs from the mmwave sensor used on the generic PCB, so this has to be taken into account as well. Similarly to the Generic PCB, this PCB also has an area designated for both environmental sensors (BME280 and ENS160+AHT21).

Surgeon PCB

The Surgeon PCB will be the smallest PCB of the three, since it will require the least amount of components and needs to be incorporated into a wearable device for surgeons that does not impede their ability to perform their job. The main components that need to fit onto this PCB will again be the microcontroller and also the camera module.

Aside from the specific components mentioned for each PCB, there needs to be some space left over for the smaller, generic components that are required for the circuit to function such as resistors, capacitors, connector pins, mounting holes and every board also has space for a Light Dependent Resistor (LDR) that allows for the measurement of light intensity in the room, as well as indicator LEDs for the powerstate, wifi status, and camera powerstate for the boards with a camera.



(a) The layout for the Generic PCB. **(b)** The layout for the Bedroom/Mirror PCB. **(c)** The layout for the Surgeon PCB.

Figure 7.1: Figures showing the final designs for the layout of the PCBs.

7.4. Schematic Design

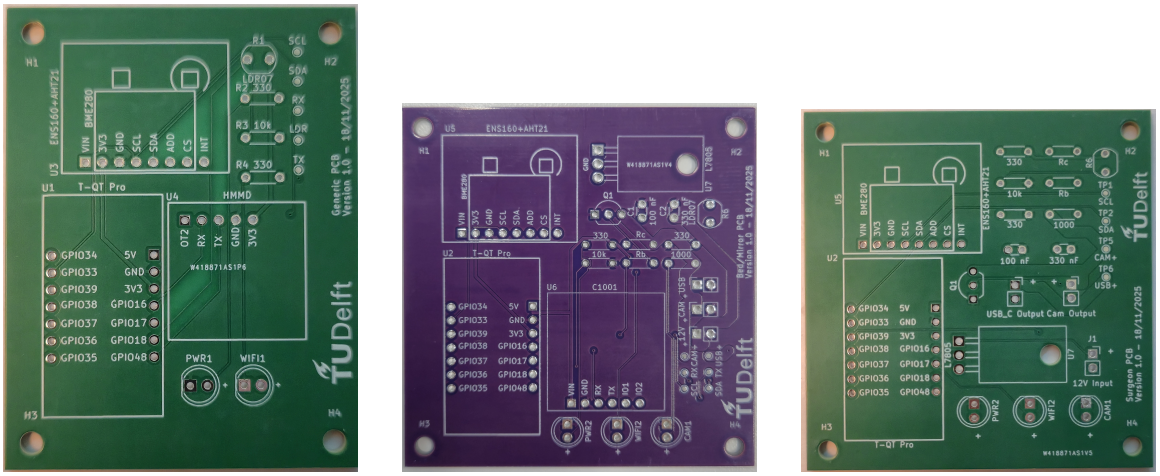
Now that the components on every PCB have been established, it is time to lay out the connections necessary for the device to function properly. Most of the connections are straightforward and can be decided based off of the information in the respective data-sheets for the components. Figures D.1, D.2, and D.3 in the appendix show the schematics for the Generic, Bedroom/Mirror, and Surgeon PCBs, respectively.

7.5. PCB Layout and Routing

As the schematics are established, the focus can shift to the physical PCB layout, which includes the routing of the connections. There are several points that should be taken into account while working on the PCB layout. First of all, the dimensions of the board should adhere to the limitations set by the manufacturer that will produce the PCBs. Secondly, it should be made sure that the components are laid out in such a way that there is no interference between the signals. However, in this specific case there are no high frequency signals that could cause such interference, so this is not much of an issue in the scope of this project. Lastly, during the routing process, special care should be taken to widen the traces that carry a significant amount of current. Making such traces too narrow can cause overheating because of the high resistance of the narrow trace. This in turn can cause voltage drops or even physical damage to the PCB in the long term.

Taking into account the aforementioned rules and careful planning led to the resulting PCB designs shown in figures 7.1a, 7.1b and 7.1c for the generic PCB, bedroom/mirror PCB and surgeon PCB, respectively.

After the designs were checked, the files were sent to the manufacturer, and the resulting product can be seen in figures 7.2a, 7.2b and 7.2c.



(a) The final Generic PCB. **(b)** The final Bedroom/Mirror PCB. **(c)** The final Surgeon PCB.

Figure 7.2: Figures showing the physical PCBs after manufacturing.

8

Testing and Results

8.1. Casing

In order to test the system, a setup needs to be created that is free of interference . These setups will be based on the use cases for the eventual product, thus the gathered test data accurately represents the real-world data. Two different PCBs are designed, which will be incorporated into a casing to protect the hardware. To make sure this casing does not interfere with the data acquisition, sufficient testing has to be done to ensure unobstructed and accurate use of the device.

8.1.1. Testing With PLA

For this prototype, the casing will be 3D printed. The material used for this will be Polylactic Acid, also known as PLA. PLA is a biodegradable bioplastic from renewable sources like cornstarch. Using PLA will allow for fast prints, low material costs and structural integrity that is sufficient for a prototype of the casing. Since the casing will cover the entire device, testing needs to be done to find out to what extend this specific material obstructs the sensors and hinders the data acquisition.

Test Setup

The setup used to test will be as follows: the sensors will be placed on the PCB as would be the case in the actual device. Several PLA squares of 2 cm by 2cm will be printed with different thickness and 100% infill. The sensor data will first be collected without any obstructions. This will serve as the control group. After that the PLA squares will be placed on top of the sensors to block their path and again the data will be collected. This will be done for different thickness squares as well as squares with slits of different width to see if making small openings in the squares makes a difference. In addition to this there will also be a test for a mirror which is made out of mirror foil attached to acrylic glass. The process will be repeated for all sensors and incorporated into a table 8.1.

Results

Table 8.1: Test results for the different thickness PLA plates and the mirror.

Sensor	0.5 mm PLA	1.0 mm PLA	2.0 mm PLA	3.0 mm PLA	2.0 mm Slit PLA	1.25 mm Slit PLA	0.75 mm Slit PLA	Acryllic Mirror
HMMD	Motion Detected	Motion Detected	Motion NOT Detected	Motion NOT Detected	Motion Detected	Motion Detected	Motion Detected	Motion NOT Detected
C1001	-	-	-	-	-	-	-	-

The C1001 mmwave sensor did not work in the end as the values it output were not reliable, so the results from this sensor were not taken into account.

For the environment sensors, it did not matter whether any of the plates were placed in front of the sensor, as long as the sensor was still in contact with the air in the environment. For this reason, these sensors were excluded from the table as the results were not interesting.

8.2. Wi-Fi stability

In order to have a reliable system the connectivity of the system needs to be reliable. To determine the Wi-Fi stability the main thing that can be measured is signal strength. In order to determine if the signal

is strong enough different test measurements have been designed to determine the stability in certain applications. The microcontroller is tested on a channel frequency of 2.4 GHz, other frequencies like 5 GHz or 6 GHz are not supported by the ESP32-S3 chip. The tests were performed in a real world environment with interference from other networks present. The tests are not representative of all environments and show a baseline of the performance in a real world environment, thus not all. The tests were performed using a local hotspot from a smartphone for its Wi-Fi network and the complete sensor module in its casing. A smartphone hotspot was used in order to easily manipulate the test arrangement. The tests were performed in a straight hallway of 12 meters. an overview of the test setup can be found in figure 8.1.

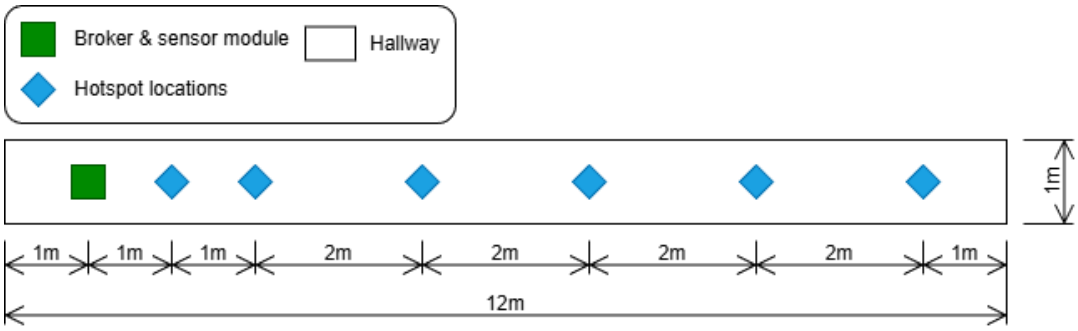
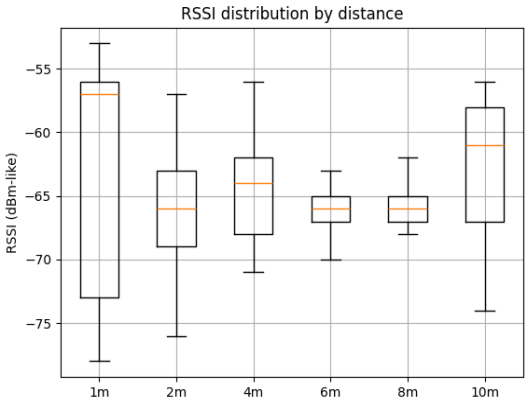
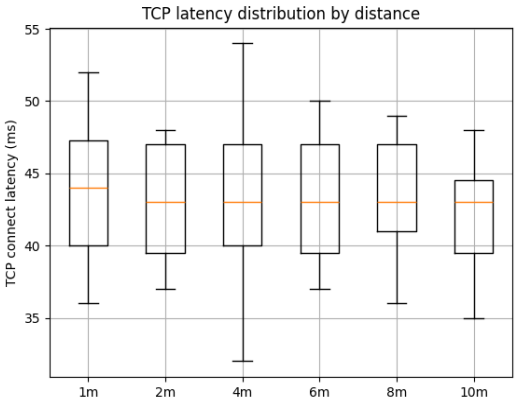


Figure 8.1: Figure showing the test setup used for the Wi-Fi tests.

The test was done at the following distances between the Hotspot and the sensor module: 1m, 2m, 4m, 6m, 8m and 10m. The measured values are the Received Signal Strength Indicator(RSSI) and the TCP latency. The results for the different distances can be found in figure 8.2. The data is gathered over a period of one minute. Figure 8.2a shows that the RSSI for the signal is well within a good margin for a stable connection across a range of 1-10 meter(s). The variations displayed are rather a position based variability. The multipath propagation of the Wi-Fi signal causes there to be dead zones were the signals interfere such that the RSSI is lower. This explains the differences shown in the graph. Figure 8.2b shows that during a line-of-sight test for a distance between 1-10 meter(s), the time it takes to establish a TCP connection remains approximately the same. This indicates that there is no distance based degradation in the time to connect to the port of the broker.



(a) The distribution of the RSSI at differing distances.



(b) The distribution of the TCP latency at differing distances.

Figure 8.2: Figures showing the results of the Wi-Fi tests.

8.3. MQTT connection

Delay of messages send

In order to test the delay of the system. the time difference between sending and receiving a message is measured. This delay between sending and receiving is often called the latency. In order to measure the latency the sensor module sends its timestamp in nanoseconds since the epoch, in our case for micropython 00:00,00 01-01-2000, to the SBC. The SBC gives the message a timestamp in nanoseconds based on the epoch of the microcontroller when it has received a message. The difference between the two timestamps is calculated to get the latency encountered between each measurement. However the difference between the clock of the sensor module and the SBC is different due to the sensor module receiving the time from the NTP server with a delay. This causes us to test for messages being send both ways from the SBC to the sensor module and from the sensor module to the SBC. With the measurements the average Round Time Trip can be calculated. This is the time for a message to be send from the sensor module to the SBC and back from the SBC to the sensor module. The RTT is calculated with equation 8.3, here $(t_1 - t_0)$ signifies the delay from the server to the sensor module minus the offset and $(t_3 - t_2)$ signifies the average delay from the sensor module to the server plus the offset.

$$(t_1 - t_0) = d_{cs} + \theta \tag{8.1}$$

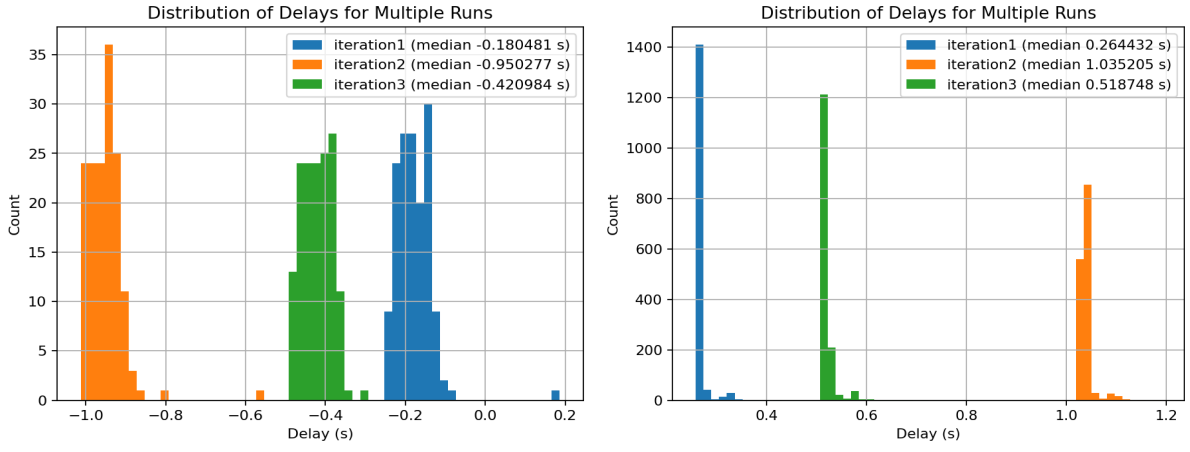
$$(t_3 - t_2) = d_{cs} - \theta \tag{8.2}$$

$$RTT = (t_1 - t_0) + (t_3 - t_2) = (d_{cs} + \theta) + (d_{cs} - \theta) = d_{cs} + d_{sc} \tag{8.3}$$

Since the offset is a constant the RTT can be calculated from the averages of the test. This results in equation 8.4.

$$\Delta RTT = \Delta(t_1 - t_0) + \Delta(t_3 - t_2) = \Delta(d_{cs} + \theta) + \Delta(d_{cs} - \theta) = \Delta d_{cs} + \theta + \Delta d_{cs} - \theta = \Delta d_{cs} + \Delta d_{sc} \tag{8.4}$$

During the tests 1500 messages were send from sensor module to the server and 150 messages were send from the server to the sensor module. The first two runs were performed in the exact same location, while the third test was performed at a different location trying to replicate the same test setup. The results from the test can be found in figure 8.3. The results show that the spread of the messages received by the sensor module is larger than that of the messages received by the SBC. This difference is presumed to be due to the asynchronous nature of the main event loop being used, thus the message being received at different times based on the asynchronous process.



(a) Messages from the SBC to the sensor module.

(b) Messages from the sensor module to the SBC

Figure 8.3: Measured time difference for send messages.

The averages and the resulting RTT can be found in table 8.2. The RTT from the tests show that the delay for a message send one way is at maximum 0.1 seconds. The delay can be estimated to be half the maximum RTT, this is under the presumption that the delay for a message is around the same for both

ways. This concludes that the delay is approximately ~0.05 seconds.

Table 8.2: Calculated average delays and RTT of the MQTT tests.

Latency measured on sensor module (s)	Latency measured on SBC (s)	Average RTT (s)
-0.1805	0.2644	0.0840
-0.9503	1.0352	0.0849
-0.4210	0.5187	0.0978

MQTT throughput of the sensor module

The MQTT throughput time indicates how much information is successfully transmitted in a second. The test was performed with the code that will be used for the sensor module there were no modifications in the send times, thus it does not indicate the max throughput, but it does indicate the throughput when it is deployed. The MQTT throughput is calculated by using equation 8.5. It is calculated by using the data from the same test used to calculate the RTT. The size of the payload send to the SBC from the sensor module is 94 Bytes, when the MQTT headers, information like packet size, topic and QoS, are included the payload is 121 Bytes long. The total time of the test was calculated by subtracting the time included in the last message from the one included in the first message. The data of the three performed tests can be found in table 8.3

$$throughput = \frac{N_{msgs}}{t_{total}} \cdot payload\_bytes \tag{8.5}$$

The results show that the system remains stable under the same conditions. It can be concluded that the sensor module sustains a rate of ~7-9 messages send every second to the broker. The MQTT throughput of the system is between 700-840 bytes per second, and 900-1080 bytes per second including the MQTT headers. The ~15% reduction between the first two tests and the third test indicates that the throughput depends on the local environment of the system.

Table 8.3: Results and the calculated MQTT throughput of the MQTT tests.

Number of messages	Total time (s)	Throughput (bytes/s)	Throughput with headers (bytes/s)	Message rate (msg/s)
3678	416.3	830.5	1069	8.83
1752	197.9	832.2	1071	8.85
2360	315.3	703.6	905.7	7.49

8.4. Main event loop performance

As described in section 6.3, the main loop is build around an asyncio event loop which allow multiple tasks to share run time with each other. The overall timing behaviour of the sensor module still depends on how long individual tasks run without yielding for other tasks or whether there are blocking operation that do not allow for yielding. To test the runtime behaviour of the sensor module an extra task was added to log the event loop stall time for two tasks: MQTT receive task and the sensor task. The stall behaviour of the sensor module was logged for 120 seconds. The test was performed with the C1001 radar sensor and the ENS160+ATH21 environment sensor connected. This is the configuration of the sensor module that loads the microcontroller the most. The results of the test can be found in figures 8.4 and 8.5. The event loop stall shows the actual time between the intended and the reference wake-up time of a reference task in our case this reference wake-up time is every 100 milliseconds. This captures the global pauses of the system that affect all tasks. The results show mostly short pauses with a handful of long pauses present. These long stalls can reach approximately 1.6-1.7 seconds, shown in figure 8.4. These pauses coincide with the large scheduling jitters present in both tasks, shown in figure 8.5b. This shows that the delay is caused by loop-wide blocking events rather than isolated timing drifts

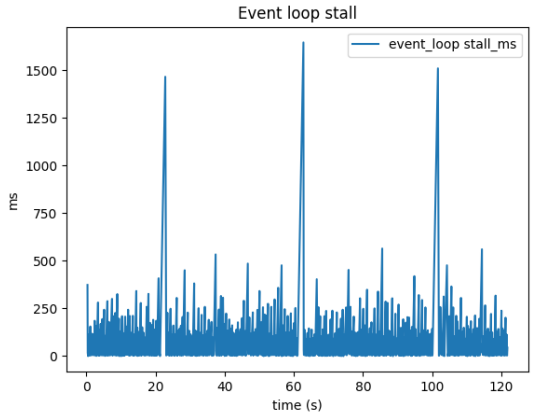
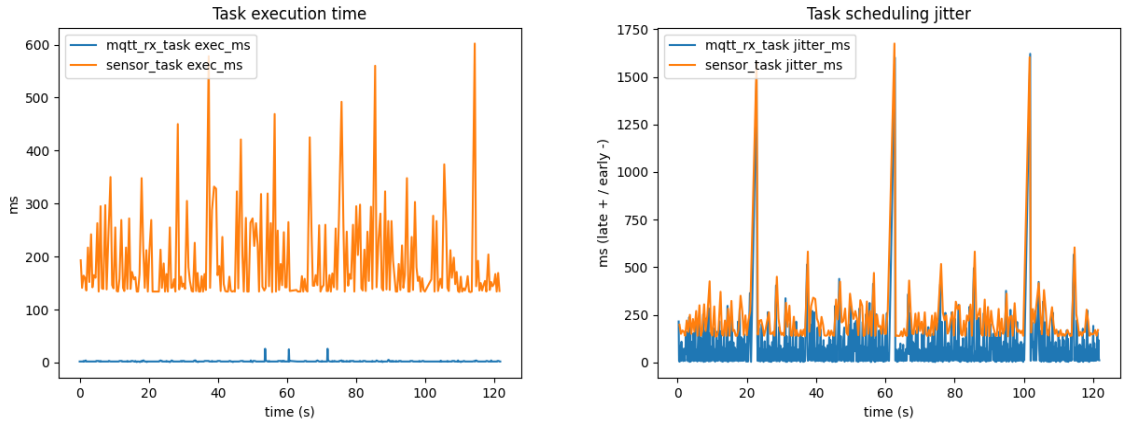


Figure 8.4: Graph showing the stalls of the main event loop during the performed test.

within single tasks. Figure 8.5a shows that the execution time is dominated by the sensor task during the test. The MQTT receive task is completed within a few milliseconds, except for some negligable spikes. The sensor task however can take up to 600 milliseconds to complete and these peaks delay the scheduling of other tasks and are the cause of some noticed stall and jitter peaks. The test indicates that the responsiveness of the sensor module's main event loop is limited by sensor data acquisition. This can be caused by stalls from the UART waiting for bytes to be received from the radar sensors, this is a blocking call. The three big spikes do not coincide with the delays caused by the large execution times of the sensor task. These are probably caused by the data logging of the implemented test itself. The test uses a logging task that logs all the data to a JSON file, in order to keep local memory available the logging tasks flushes itself every 20 seconds. This can create noticeable flush burst that create heavy stalls. The timing of the spikes coincide with these flushes occurring just after 20, 60 and 100 seconds.



(a) Graph showing the task execution time for two different tasks during the test.

(b) Graph showing the task scheduling jitter for two different tasks during the test.

Figure 8.5: Different performance measurements of the tasks in the main event loop during the test.

9 Casing Design

The final part of the hardware design is the design of the casings for the three devices. Since all three devices contain a PCB with different dimensions and some make use of a camera module, all three designs need to be unique to a certain extent. In the following sections, all of the designs will be reviewed and explained, starting with the Generic Device.

9.1. Generic Sensor Module Casing

The design of the casing starts by making sure that all components fit in the casing. For the generic device, this means that the casing should be able to contain the PCB with all the components soldered onto it. The next step is to ensure that all components and sensors are accessible from the outside without leaving them too exposed, resulting in an increased risk of damaging the electronics. To achieve this, the results in chapter 8.1 will be consulted. Using these results, the decision was made to cover the distance sensors as well as the environment sensors with a grill. This will provide sufficient protection against damage while handling the device, while not negatively influencing the sensor data acquisition. In addition to this, the area around the environment sensors will be walled off from the other components to reduce the influence of for example the heat generated by the other components on the PCB. This way, the sensor data will more accurately display the actual values of the atmosphere of the room as opposed to the atmosphere inside the device. Since the Generic device is powered via a USB-C cable plugged directly into the microcontroller, there should also be an opening in the case for this cable. Next up are the two openings for the indicator LEDs. Finally, in order to be able to make use of the buttons on the microcontroller, there should be openings in the casing in the areas where the two front facing buttons on the microcontroller are located as well as an opening where the reset button is located. To make these buttons even more accessible, extension buttons are printed separately that can be slotted into the open areas in the casing. To close the device up, a backplate will also be printed. This backplate will have screw-holes to make sure the device can be securely closed as well as easily reopened when necessary, for example, when components need to be replaced or repaired. The final result can be seen in figure E.1. The dimensions of the general casing can be found in table 9.1.

9.2. Bedroom/Mirror Sensor Module casing

Next up is the design of the casing for the device that will be used specifically in the bedroom above the bed or in the bathroom behind the mirror. This device is different from the Generic Device since the PCB is larger and, more importantly, the casing for this device should contain the camera module as well as a speaker and microphone module connected to the camera module. This means that there should be an opening in the casing for the camera lens, an opening for the speaker, and an opening for the microphone. Besides these changes, the casing design is very similar to the Generic casing design. The final result of the Bedroom/Mirror casing design can be seen in figure E.2. The dimensions of the Bedroom/Mirror Sensor Module casing can be found in table 9.1.

9.3. Surgeon Sensor Module Casing

The last design is the design for the device that surgeons will use to aid during surgeries.To make sure that this device does not impair on the ability of the surgeon to do their work properly, this device should be as small and as light as possible. This device should be a head mounted device to keep the hands of the surgeon free. It will include a camera module just like the Bedroom/Mirror device, but unlike that device the camera module will not be in the casing, but mounted to the front side a headband

while the casing will be mounted on the rear side. This device should be powered by a powerbank that provides enough power to last through a full surgery on one charge. The final design for the Surgeon casing design can be seen in figure E.3. The dimensions of the surgeion casing can be found in table 9.1.

Table 9.1: Table showing the dimensions of different sensor modules.

Type	**Height(cm)**	**Width(cm)**	**Depth(cm)**
Generic	7.75	6.10	2.40
Bedroom	8.48	11.43	2.55
Surgeon	7.75	7.25	2.40

10 Costs

To figure out whether or not the final product would be feasible to produce and sell, an overview needs to be made of the production costs of such a device. When doing this, it is wise to take into account both the costs of the concept device, as well as the costs of the device if it were to go into mass-production. An estimated cost overview of the proposed systems per individual component and the total cost can be seen in table 10.1. The main difference in costs between the concept versions and the mass production versions are the shipping costs of the components. It is feasible to assume that when buying components in bulk, agreements can be reached to further decrease the prices, but since it is hard to pinpoint exactly how much this would reduce the costs, this has not been taken into account when calculating the final costs.

Component	Concept	Mass-production
LilyGO T-QT Pro	€15.00	€10.00
C1001 mmWave Sensor	€29.00	€29.00
HMMD mmWave Sensor	€4.00	€0.80
BME280 Environment Sensor	€3.50	€0.20
ENS160+AHT21 ENvironment Sensor	€6.00	€3.50
GL5516 LDR	€4.00	€0.50
L7805 Voltage Regulator	€5.00	€0.75
LEDs	€4.50	€2.00
Camera	€20.00	€20.00
Miscellenious Components	€8.00	€2.00
PCB Production Costs	€30.00	€5.00
Casing PLA	€1.00	€1.00
Total costs Generic Sensor Module	€70.00	€21.50
Total costs Bedroom/Mirror Sensor Module	€122.50	€73.75
Total costs Surgeon Sensor Module	€93.50	€44.75

Table 10.1: Cost estimation for the production of device both as concept as well as in mass-production

11 Discussion

This chapter discusses the results found in chapter 8. The limitations of the tests and system will also be discussed.

11.1. Casing design

For the design of the casing the radar sensor was limited by the thickness of the PLA in front of the sensor. The total thickness that would detect motion was 1.0mm of PLA, but when the material in front of it included slits the thickness could at least be 2.0mm thick PLA. The acryllic mirror did not detect any motion for the HMMD sensor.

The results show that the casing of the sensor module can be atleast 1.0mm thick when designing the casing, but it could be 2.0mm thick when the slits are used for the eventual design of the case. Placing the HMMD sensor behind a mirror did not detect motion, thus this will not be a good sensor for placing it behind a mirror. The final casing designs used for prototyping included a front thickness of 1.5 millimetres with slits for the radar sensor and environment sensor. It also includes separate chambers for the environment sensors such that the heat of the components in the enclosed space interfere less with the environment sensor readings. The overall designed casing worked great and did not show any problems with interfering the sensors. The conclusive final designs are discussed in chapter 9. The casing for the surgeon was not extensively tested, thus no conclusions can be drawn for the functionality of that specific casing design.

C1001 sensor problems

The results from the C1001 sensor were redacted due to ongoing problems with the sensor output. It was observed that the sensor did not give accurate measurements, and would detect motion, presence, heart rate and breath rate even when no person was in the line-of-sight of the antenna array. This suggests that the C1001 sensor is too sensitive. The software of the C1001 mmwave sensor lacks documentation for hardware control on the sensor, thus it was not possible to change this in software. It was tried to limit the sensitivity of the sensor physically by placing differing objects in front of the sensor, yet it would still give inaccurate measurements. The term coined for these wrong output values are "ghost values".

11.2. Wi-Fi stability

The Wi-Fi tests of the sensor module indicate that the sensor module has a stable connection at distances from 1-10 meters, but it does display location based dips in connectivity. This is most likely the problem from multipath propagation that destructively interfere that causes these dips in RSSI. The TCP latency is consistent and is mostly governed by processes on the microcontroller. These results do not show how many bytes the connection is able to handle or anything about the performance of the connection, it shows that the connection created is stable.

The test was only performed in one location and indicates that the Wi-Fi connection is good for that location, but it was not tested under extreme conditions or with through-the-wall conditions. This limits the conclusion from the tests to only line-of-sight. Furthermore other metrics like package losses and dropouts could not be tested due to time constraints of the project.

11.3. The MQTT connection

The measurements for the MQTT connection display that the overall system is responsive and an expected delay below 0.1 seconds between sending the data and receiving it. The throughput of the system shows that it can send, in normal operation, between 700-840 bytes/s depending on the conditions of the network connection. That accounts to ~7-9 messages per second depending on the size of the messages. The performance of the system is good enough for its application, especially since nothing is sent when there is no change in the data.

11.4. The main loop delays

The test of the main loop of the deployment program displays that the sensor module takes a long time gathering data from the sensors, this takes between ~150-600 milliseconds. This is considered the largest delay in the system and it hogs most of the runtime with IO events for getting the sensor data. This is indicated for the Bedroom/Mirror Sensor Module and does not actually show the performance of the generic sensor module, but since this sensor module has one less sensor and has less readings in total that sensor module is expected to perform better overall.

Sensor module delay

The overall delay of the sensor module is mostly governed by the data acquisition plus the delay time of the MQTT connection. From the tests it can be concluded that the delay of the sensor module is between ~200-700 milliseconds.

12 Conclusion

In this chapter a conclusion will be drawn from the validation based on the initial program of requirements in chapter 4. Recommendations for future work will also be discussed.

12.1. Conclusion

The final sensor module functions as expected and the PCB works according to design with the code on the microcontroller. All the sensors can be connected and read by the microcontroller and only the changed data will be send to the SBC via a MQTT connection secured by mutual TLS over the LAN. The integration with the overall system is thereby complete for the deployment phase. The registration of the sensor modules works according to the registration protocol of the registration subgroup[13], thus the integration with the overall system is done. The delay of the system is within the stated limit of one second and all the sensor data can be read within these times. All the physical aspects on the final system are present. It has buttons and the correct LEDs present that function correctly. The total size of the sensor module depending on which type can fit in your hand with the Generic Sensor Module being the smallest at: H 7.75 cm X W 6.10 cm X D 2.40 cm, while the largest is the Bedroom/Mirror Sensor Module at: H 8.48 cm X W 11.43 cm X D 2.55 cm. The casings designed are fine for the concept created. The Bedroom/Mirror Sensor Module has a twelve volt connection for power that enables the mounting of a camera within the housing. All in all, the final product adheres to the program of requirements with some slight problems of the C1001 mmwave radar sensor and some delay issues in the main loop of the deployment program.

12.2. Future work

There were many things this project could not do in the limited available time. Thus here are some recommendations for future work that can be done to improve the sensor module in the future.

12.2.1. Software

The results show that the biggest task that blocks the main event loop is the code for data acquisition. In the future this code should be optimized more such that this task blocks the main loop for a shorter time. This can decrease the delay between sensing data and receiving it on the SBC. One of the biggest things that can still be done with the sensor module software in the future is to rewrite the code to C or embedded rust for greater control and optimization of the processes on the microcontroller. Rewriting it in rust is better since the industry is moving towards that direction, it is memory efficient and there is a lot of documentation available. During the project problems were encountered with running everything when the display tried to turn on the device disconnected from the PC. Everything for the display is included in the initialization happening in main but it is commented out.

12.2.2. Hardware

The biggest recommendation is to switch to soldered sensors on the board this can reduce the size of the sensor module significantly. It also ensures that mysterious hardware disconnects are not likely to happen due to badly soldered wires or just normal use. There are also many component upgrades that are recommended for future systems. These recommendations are mentioned in section 12.2.3. Mainly the processing power of the microcontroller limits the encryptions it is able to use. Furthermore, there should be accuracy tests done of the sensors compared with calibrated tools. The light sensor works, but should be calibrated to a fixed light source. And it should be researched why the microcontroller

disconnects when the display is initialized. It is probably a brown-out. The casing designed right now is good enough for prototyping, but research should be done on injection moulding of the final casing and the production possibilities of the different sensor modules.

12.2.3. Alternative Hardware Recommendations

For this project, most of the components were already provided. After working with these components, some recommendations for improvements can be given.

Microcontroller

The microcontroller that is used is the LilyGO T-QT Pro, as mentioned before. This microcontroller was chosen mainly due to the fact that it has a programmable LED display. However, this microcontroller sacrifices in other areas. One of the main issues encountered while using this microcontroller was its lack of processing power. Besides this, it became evident that this microcontroller is not widely used from the lack of documentation that can be found on the usage of this component. This scarce usage also contributes to reduced availability and therefore relatively higher cost.
Luckily there are plenty of alternatives available. We have two different recommendations for a replacement; the Raspberry Pi Pico 2 W or the Raspberry Pi Zero 2 W.
The Raspberry Pi Pico 2 W is a comparable device to the ESP32-based LilyGO T-QT Pro, but it has some advantages that would make it a better suited option for this specific use case. The Pico 2 W runs out of the box at a lower clock speed than the ESP32 does (133 MHz vs 240 MHz), but the Pico 2 can be overclocked to around 300 MHz which makes it a lot faster. Although that might not even be necessary since benchmark tests show that even at 133 MHz the Pico 2 W completes tasks faster than the ESP32 does. Besides this, the Pico 2 W has a significantly lower power consumption which makes it more suitable for battery powered applications, such as the Surgeon Device we designed. The Pico 2 W can be equipped with a display just like the T-QT Pro is, but it would also require a stand-alone camera module since it cannot process images with the resolution required for this project. The price point of the Pico 2 W is around $5, and with that lies at about half of the price of the T-QT Pro, which is also a major upgrade.
The Raspberry Pi Zero 2 W is a significant upgrade over both the Pico 2 W and the T-QT Pro in terms of processing power. This does, however, come at a price. The Zero 2 W costs about 3 times as much as the Pico 2 W and around 1.5 times as much as the T-QT Pro with a price of about $15. The dimensions of the Zero 2 W also force the device to be slightly larger with the length of the Zero 2 W at about 60 mm and the width around 30 mm. The reason we would be willing to sacrifice in these areas are the major advantages that the Zero 2 W brings. The Zero 2 W is a tier above the other two microcontrollers since it is closer to a small computer than a microcontroller. The Zero 2 W is equipped with a quad-core 64-bit ARM Cortex-A53 CPU clocked at 1GHz. This is processor has much more processing power compared to the other two. This allows for the code to run much faster and more efficient, while allowing for more robust encryption algorithms than currently possible as well as more room for future expansion of the device's functionality. Another major advantage is that a camera module can be directly connected to the Zero 2 W, which saves space, power, and costs. The downside is that the Zero 2 W is not suitable for battery powered applications due to its relatively high power consumption, so for the Surgeon Device another option would have to be picked.
Both of the suggestions we made are also more widely used options than the LilyGO T-QT Pro, so there is more documentation and information available which will help during the development of the devices. The lack of documentation on the T-QT Pro showed to be an issue at times, so this would also be a considerable upgrade.

Camera

For this project the camera that can be used should meet very specific criteria. Meeting these criteria while staying within a feasible price range was a big challenge that we could not fulfil to our desired level. Using a Raspberry Pi Zero 2 W, as stated in the previous section, could also solve this issue. This is because when the microcontroller is replaced by a Raspberry Pi Zero 2 W, a camera module can directly be connected to the microcontroller as opposed to functioning as a stand-alone module. This would solve the issue of specifically needing a camera module with Onfiv protocol T, simplify the power circuits, reduce the costs, and allow for the device to be smaller.

Display
When choosing a different microcontroller, the option will also be there to connect a display to it. The LilyGO T-QT Pro already has a display connected to the board, but this takes away any ability to change the display or upgrade to a better or larger display. Using a microcontroller without a built in display gives more freedom to chose what display to use, as well as where to place the display since it is not fixed to the microcontroller.

Final Verdict
Regardless of whether or not the decision will be made to switch to a Rapsberry Pi device as we suggested, we would at least advice to use a different ESP32 board since in our opinion the LilyGO T-QT Pro lacks documentation, is not very readily available, takes away the freedom of choosing a specific display and is overpriced for the value it offers.

Bibliography

[1] Centraal Bureau voor de Statistiek. *Prognose bevolking; Geslacht en Leeftijd, 2025-2070*. Dec. 2024. URL: https://www.cbs.nl/nl-nl/cijfers/detail/86041NED.

[2] M. Stevenson. URL: https://www.who.int/news-room/fact-sheets/detail/ageing-and-health.

[3] Lingran Xu et al. "A systematic review of digital literacy in Lifelong learning for older adults: Challenges, strategies, and learning outcomes". In: *Educational technology research and development* (July 2025). DOI: 10.1007/s11423-025-10530-w.

[4] Alexandros Moraitopoulos, Antonis Billis, and Panagiotis Bamidis. "Empowering Ageing with Ingenuity: A Scoping Review on the Methods and Technologies Used into Elderly Health Monitoring". In: *Studies in health technology and informatics* 316 (Aug. 2024), pp. 525–529. DOI: 10.3233/SHTI240465.

[5] Lidan Li et al. "Research on Improvement of Configurable I2C controller IP Core". In: *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*. 2023, pp. 484–489. DOI: 10.1109/ICCECT57938.2023.10141414.

[6] Dvijen Trivedi et al. "SPI to I2C Protocol Conversion Using Verilog". In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, pp. 1–4.

[7] Weilun Huang and Guolun Sheng. "Analysis and Research on UART Communication Protocol". In: *2024 4th Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*. 2024, pp. 768–771. DOI: 10.1109/ACCTCS61748.2024.00140.

[8] Jiajun Gong, Wenbo Guo, and Weijian Sun. "UART communication protocol frame format explanation and application". In: *Applied and Computational Engineering* 14 (Oct. 2023), pp. 47–56. DOI: 10.54254/2755-2721/14/20230757.

[9] M. Saravanan et al. "Implementation of 8-Bit SPI Protocol Using System Verilog and UVM". In: *2024 4th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS)*. 2024, pp. 1893–1898. DOI: 10.1109/ICUIS64676.2024.10867233.

[10] MicroPython project. *MicroPython*. Website. URL: https://micropython.org/.

[11] Gabriel Gaspar et al. "Development of IoT applications based on the MicroPython platform for Industry 4.0 implementation". In: *2020 19th International Conference on Mechatronics - Mechatronika (ME)*. 2020, pp. 1–7. DOI: 10.1109/ME49197.2020.9286455.

[12] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks". In: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*. 2008, pp. 791–798. DOI: 10.1109/COMSWA.2008.4554519.

[13] Simon van der Avoird Yannick Rijkers and Rutger van Leeuwen. "Hardware-Driven (De-)Registering Peripheral Devices for a Medical System". Report of the other subgroup. 2026.

[14] Mohammad Faiz Usmani. *MQTT Protocol for the IoT - Review Paper*. May 2021. DOI: 10.13140/RG.2.2.26065.10088.

[15] André Perez. "SSL, TLS and DTLS Protocols". In: *Network Security*. Wiley Data and Cybersecurity, 2014, pp. 109–132. DOI: 10.1002/9781119043942.ch5.

[16] Sean Turner. "Transport Layer Security". In: *IEEE Internet Computing* 18.6 (2014), pp. 60–63. DOI: 10.1109/MIC.2014.126.

[17] Iqbal Luqman Bin Mohd Paris, Mohamed Hadi Habaebi, and Alhareth Mohammed Zyoud. "Implementation of SSL/TLS Security with MQTT Protocol in IoT Environment". In: *Wireless Personal Communications* 132 (July 2023), pp. 163–182. DOI: 10.1007/s11277-023-10605-y. URL: https://doi.org/10.1007/s11277-023-10605-y.

[18] Ambarish G. Mohapatra et al. "IoT-driven remote health monitoring system with sensor fusion enhancing immediate medical assistance in distributed settings". In: *Alexandria Engineering Journal* 120 (2025), pp. 627–636. ISSN: 1110-0168. DOI: https://doi.org/10.1016/j.aej.2025.02.057. URL: https://www.sciencedirect.com/science/article/pii/S1110016825002340.

[19] Hamoud H. Alshammari. "The internet of things healthcare monitoring system based on MQTT protocol". In: *Alexandria Engineering Journal* 69 (Apr. 2023), pp. 275–287. DOI: 10.1016/j.aej.2023.01.065. URL: https://www.sciencedirect.com/science/article/pii/S1110016823000881.

[20] Surabhi Joshi and Sunil Joshi. "A Sensor based Secured Health Monitoring and Alert Technique using IoMT". In: *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*. 2019, pp. 152–156. DOI: 10.1109/ICCT46177.2019.8969047.

[21] Lorie Donelle et al. "Passive Remote Monitoring Technologies' Influence on Home Care Clients' Ability to Stay Home: Multiprovincial Randomized Controlled Trial". In: *JMIR Aging* 8.1 (Mar. 2025), e69107. DOI: 10.2196/69107. URL: https://aging.jmir.org/2025/1/e69107/.

[22] Eleonore Bayen et al. "Reduction of Time on the Ground Related to Real-Time Video Detection of Falls in Memory Care Facilities: Observational Study". In: *Journal of Medical Internet Research* 23.6 (June 2021), e17551. DOI: 10.2196/17551. URL: https://www.jmir.org/2021/6/e17551/.

[23] Nico Jähne-Raden et al. "INBED: A Highly Specialized System for Bed-Exit-Detection and Fall Prevention on a Geriatric Ward". In: *Sensors* 19.5 (Feb. 2019), p. 1017. DOI: 10.3390/s19051017. URL: https://www.mdpi.com/1424-8220/19/5/1017.

[24] Toshiyo Tamura and Ming Huang. "Unobtrusive Bed Monitor State of the Art". In: *Sensors* 25.6 (Mar. 2025), p. 1879. DOI: 10.3390/s25061879. URL: https://www.mdpi.com/1424-8220/25/6/1879.

[25] Hannelore Strauven et al. "Unobtrusive Nighttime Movement Monitoring to Support Nursing Home Continence Care: Algorithm Development and Validation Study". In: *JMIR Nursing* 7 (Dec. 2024), e58094. DOI: 10.2196/58094. URL: https://nursing.jmir.org/2024/1/e58094/.

[26] U.S. Food and Drug Administration. *De Novo Classification Request for Oxehealth Vital Signs: Decision Summary (DEN200019)*. Tech. rep. DEN200019. Decision granted (De Novo). Center for Devices, Radiological Health (CDRH), U.S. Food, and Drug Administration, Mar. 2021. URL: https://www.accessdata.fda.gov/cdrh_docs/reviews/DEN200019.pdf.

[27] Oxehealth Limited. *Oxehealth Vital Signs: Instructions for Use*. Version 9.0. EN-UK. Instructions for use; for use with Oxehealth Vital Signs versions 1.2 and 1.12. Oxehealth Limited. Oct. 2018. URL: https://cdn.prod.website-files.com/64070a459cda4dafc482acef/648ae4aef8a7ff8e3e39fef7_Vital%20Signs-IFU-Instructions%20For%20Use-EN-UK-9.0-1.2%20and%201.12.pdf.

[28] Yingxiao Wu et al. "Non-intrusive Human Vital Sign Detection Using mmWave Sensing Technologies: A Review". In: *ACM Trans. Sen. Netw.* 20.1 (Nov. 2023). ISSN: 1550-4859. URL: https://doi.org/10.1145/3627161.

[29] Vayyar. *Vayyar Care: Fall Detection Exclusively with Alexa Together*. Online documentation page. URL: https://vayyar.com/care-docs/b2c/.

[30] Essence SmartCare Ltd. *Care@Home™ Activity Monitoring*. White paper. Essence SmartCare Ltd., Mar. 2019. URL: https://catalogartifact.azureedge.net/publicartifacts/essencegroup1598265860129.essence_smartcare-cf3087d2-626f-4ea0-b75c-d2de529cb573/Artifacts/Documents/EssenceSmartCareWhitePaper.pdf.

[31] SafelyYou. *SafelyYou Safety AI™: How It Works*. Website. URL: https://www.safely-you.com/safelyyou-safety-ai/.

[32] Xinyuan-LilyGO. *Xinyuan-LilyGO/T-QT*. Aug. 2024. URL: https://github.com/Xinyuan-LilyGO/T-QT/blob/main/schematic/T-QT-Pro.pdf.

[33] Waveshare. *HMMD mmWave Sensor*. Waveshare Wiki. URL: https://www.waveshare.com/wiki/HMMD_mmWave_Sensor.

[34] DFRobot. URL: https://wiki.dfrobot.com/SKU_SEN0623_C1001_mmWave_Human_Detection_Sensor.

[35] DFRobot. *DFRobot/dfrobot humandetection*. URL: https://github.com/DFRobot/DFRobot_HumanDetection/tree/master.

[36] Banggood Ltd. *GY-BMP280-3.3 Module Datasheet*. PDF datasheet. BMP280 temperature and barometric pressure sensor module; I²C/SPI interface; 3V supply. 2017. URL: https://www.bosch-sensortec.com/bst/products/all_products/bmp280.

[37] Bosch Sensortec GmbH. *BME280 Data sheet: Digital humidity, pressure and temperature sensor*. BST-BME280-DS001-24. Document revision 1.24; Document release date February 2024; Sales Part Number (SPN) 0 273 141 185. Reutlingen, Germany, Feb. 2024. URL: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/.

[38] ASAIR (Aosong Electronics Co., Ltd.) *Data Sheet AHT21: Humidity and Temperature Sensor*. Version V1.0. Datasheet, 16 pages; www.aosong.com. ASAIR. May 2021. URL: https://www.aosong.com/.

[39] ScioSense B.V. *ENS160 Datasheet: Digital Metal-Oxide Multi-Gas Sensor*. Version 1.1. Release date: 2022-03-30; Document status: Production. ScioSense B.V. High Tech Campus 10, 5656 AE Eindhoven, The Netherlands, Mar. 2022.

[40] Shenzhen Senba Optical & Electronic Co., Ltd. *GL55 Series CdS Photoresistor Manual*. Datasheet/-manual for GL55-series photoresistors (includes GL5549); 6 pages; spectrum peak 540 nm; company website: www.sbcds.com.cn. Shenzhen Senba Optical & Electronic Co., Ltd. URL: http://www.sbcds.com.cn.

[41] STMicroelectronics. *L7800 Series: Positive Voltage Regulators*. Datasheet; Rev. 12; output current up to 1.5 A; fixed output voltages 5 V to 24 V. STMicroelectronics. Nov. 2004. URL: https://www.st.com/.

[42] Briana L. Moreland et al. "A descriptive analysis of location of older adult falls that resulted in emergency department visits in the United States, 2015". In: *American Journal of Lifestyle Medicine* 15.6 (Aug. 2020), pp. 590–597. DOI: 10.1177/1559827620942187.

[43] MicroPython Developers. *micropython-umqtt.simple (PyPI package)*. Version 1.3.4. Lightweight MQTT client for MicroPython. License: MIT. June 2017. URL: https://pypi.org/project/micropython-umqtt.simple/.

[44] Eric Bogatin. *Bogatin's Practical Guide to Prototype Breadboard and PCB design*. Artech House, 2022.

[45] Mark I. Montrose. *EMC and the Printed Circuit Board: Design, Theory, and Layout Made Simple*. 1st. A comprehensive reference on electromagnetic compatibility in PCB design, covering layout strategies, grounding, and signal integrity. Piscataway, NJ: IEEE Press, 2000. URL: https://ieeexplore.ieee.org/book/5265135.

A

Division of labour

Group

- Background research.
- Research into state of the art technologies.
- Overall design of the sensor module.
- Integration and validation.

Hugo Snelder
Researched, created or worked on:

- Micropython.
- Communication protocols for microcontrollers.
- state of the art technologies.
- MQTT and TLS.
- Software for the sensor module.
- integration of the sensor module registration protocol
- integration of software with hardware of the sensor module.
- Testing of the Wi-Fi connection.
- Testing of the MQTT connection.
- Testing of the main event loop.
- An exe tool for flashing and moving files to and from the sensor module.

Malik fernald
Researched, created or worked on:

- Background research Onvif Protocol
- Resarch and integration camera module
- Component research
- Circuit design
- PCB design, integration, and testing
- Casing design and testing
- Cost overview and estimation
- Future hardware alternatives research and recommendations

B

Configuration file example

This chapter shows the config.json file for clarity of thre reader and gives an indication for included parameters for people wanting to recreate the system.

configuration file of unregistered sensor module begin

```
{
  "wifi": {
    "ssid": "DeviceRegistrationNetwork",
    "LED_pin": 17,
    "password": "Device2025"
  },
  "device": {
    "location": "home"
  },
  "registered": null,
  "LABEL": null,
  "UUID": null,
  "i2c": {
    "i2c_sda": 34,
    "i2c_scl": 33,
    "i2c_id": 1
  },
  "mqtt": {
    "topic": "data",
    "BROKER": null,
    "PASSWORD": "new123",
    "debug": false,
    "port": 8883,
    "USER": "new_dev"
  },
  "uart": {
    "hmmd_rx": 48,
    "uart": 1,
    "hmmd_tx": 16,
    "C1001_rx": 48,
    "C1001_tx": 16
  },
  "environment": {
    "hum_res": 1,
    "temp_res": 0.5,
    "co2_res": 5,
    "pres_res": 5,
    "light_res": 5
  },
  "sensors": {
    "environment": null,
```

```
    "radar": null
  },
  "publish_interval": 0.1,
  "LDR_pin": 18
}
```


configuration file of unregistered sensor module end

configuration file of registered sensor module begin


```
{
  "wifi": {
    "ssid": "DeviceRegistrationNetwork",
    "LED_pin": 17,
    "password": "Device2025"
  },
  "device": {
    "location": "home"
  },
  "registered": true,
  "LABEL": "FLYL",
  "UUID": "c0423d6c-baa7-4545-a3ba-83f13a8c304c",
  "i2c": {
    "i2c_sda": 34,
    "i2c_scl": 33,
    "i2c_id": 1
  },
  "mqtt": {
    "topic": "data",
    "BROKER": "192.168.0.103",
    "PASSWORD": "new123",
    "debug": false,
    "port": 8883,
    "broker": "192.168.0.104",
    "USER": "new_dev"
  },
  "uart": {
    "hmmd_rx": 48,
    "uart": 1,
    "hmmd_tx": 16,
    "C1001_rx": 48,
    "C1001_tx": 16
  },
  "environment": {
    "hum_res": 1,
    "temp_res": 0.5,
    "co2_res": 5,
    "pres_res": 5,
    "light_res": 5
  },
  "sensors": {
    "environment": "BME280",
    "radar": "DFRobot"
  },
  "publish_interval": 0.1,
  "LDR_pin": 18
}
```


configuration file of registered sensor module end

C

Lilygo TQ-T pro ESP32S3

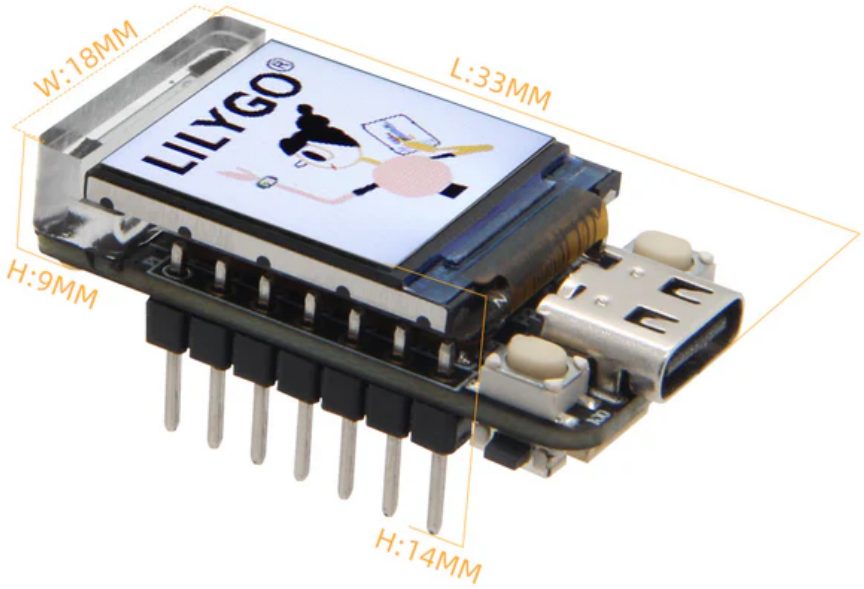


Figure C.1: Dimensions of the Lilygo T-QT Pro micro-controller taken from the Lilygo website.[32]

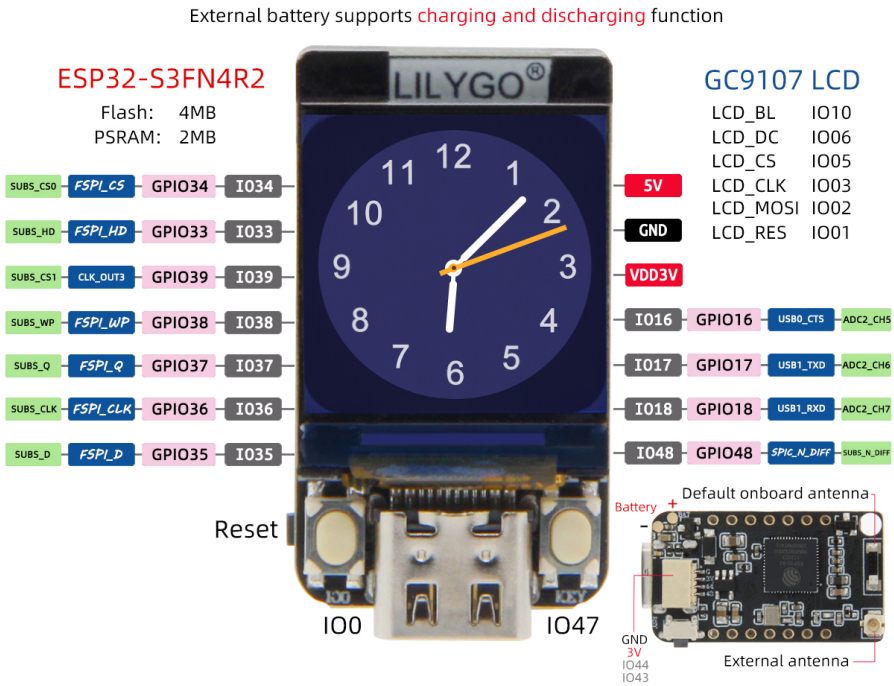


Figure C.2: pinout of the Lilygo T-QT Pro micro-controller taken from Xinyuan-Lilygo github repository.[32]

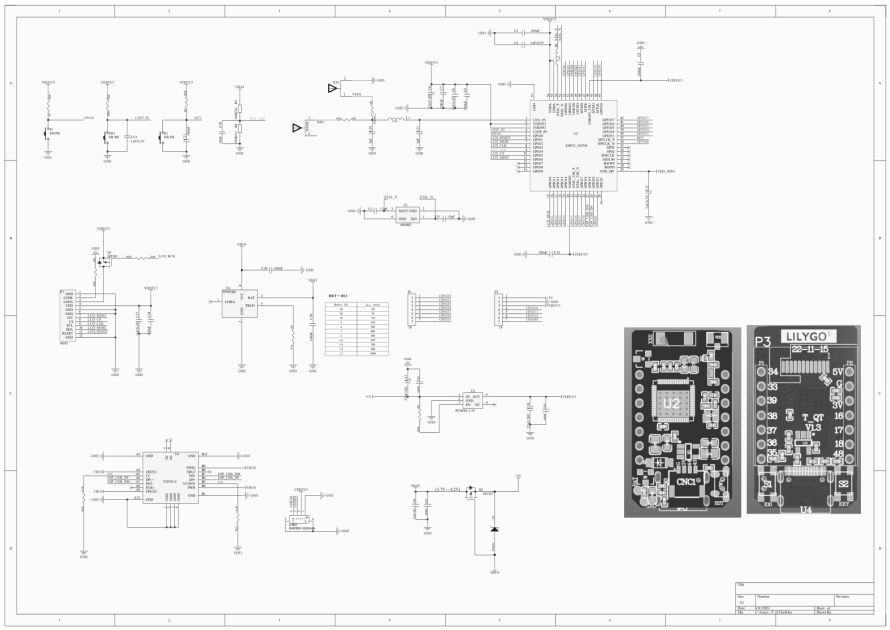


Figure C.3: schematic of the Lilygo T-QT Pro micro-controller taken from Xinyuan-Lilygo github repository.[32]

D

PCB Schematics

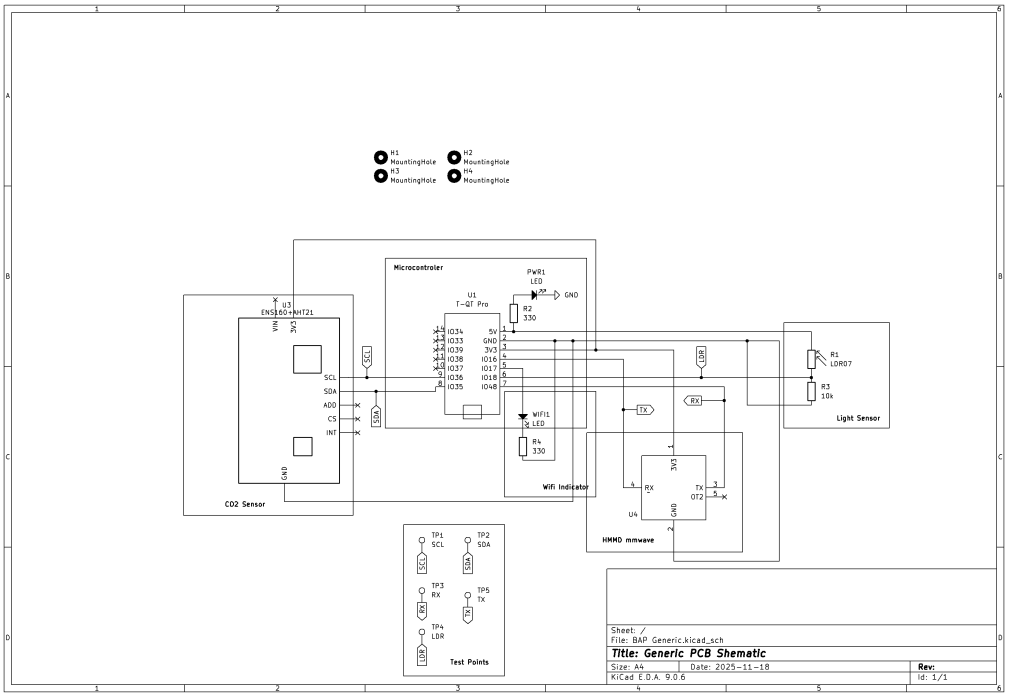


Figure D.1: The schematic for the generic PCB

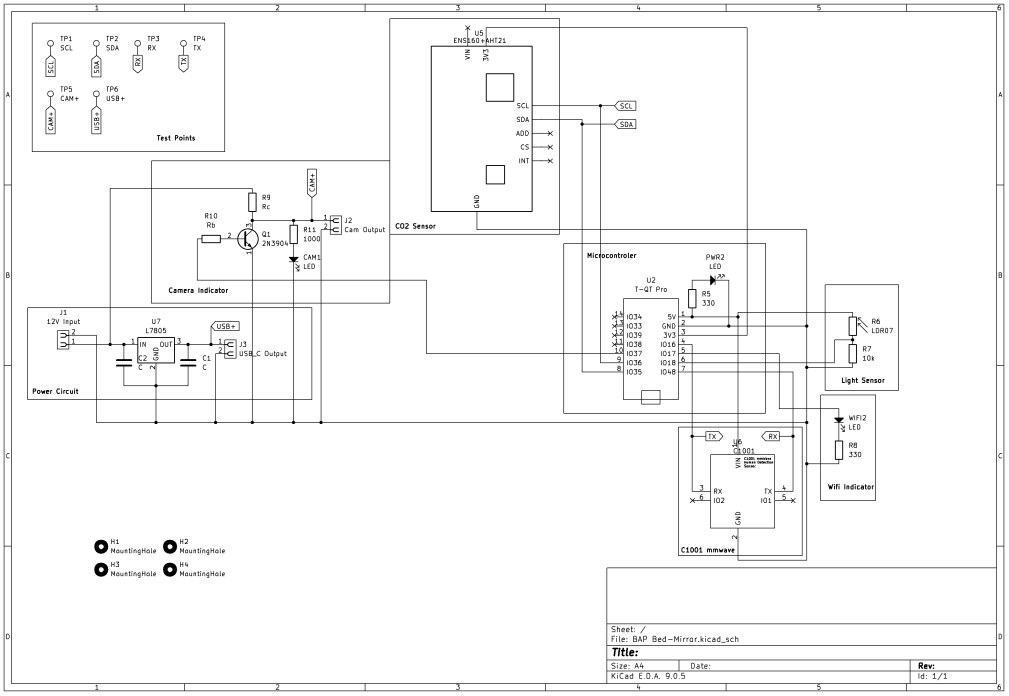


Figure D.2: The schematic for the bedroom/mirror PCB

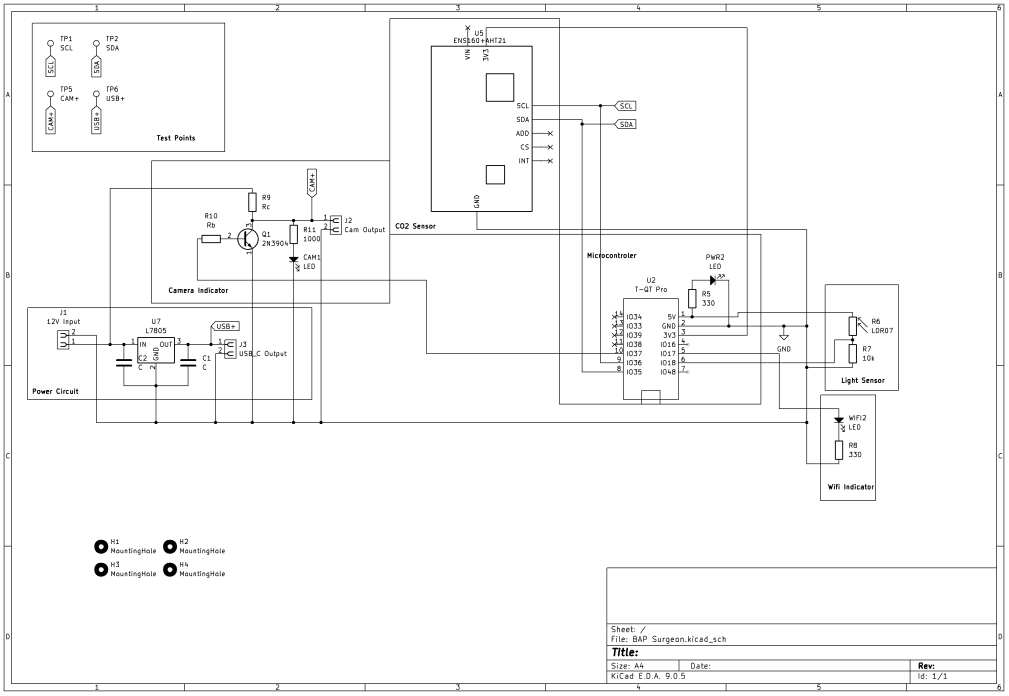
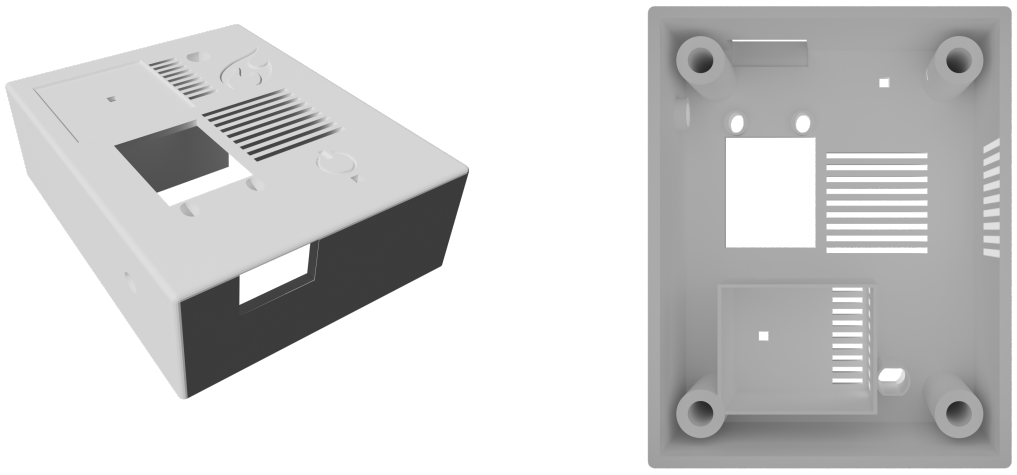


Figure D.3: The schematic for the surgeon PCB

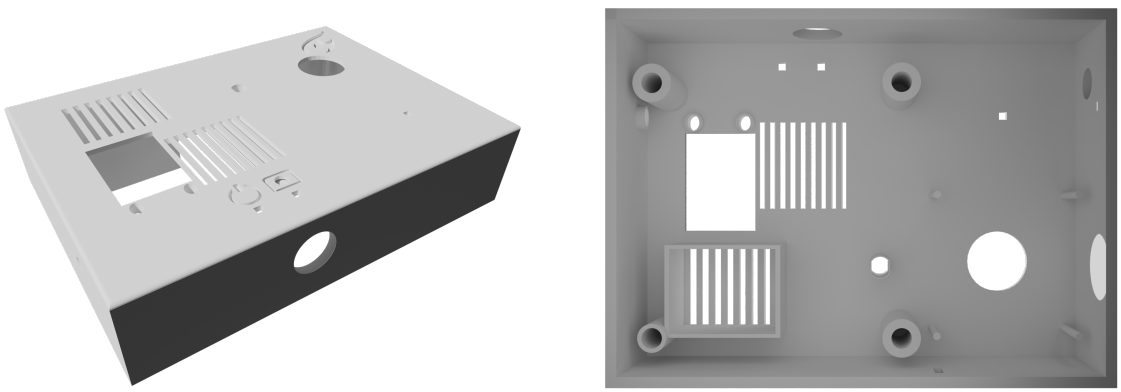
E

Casing Design Pictures



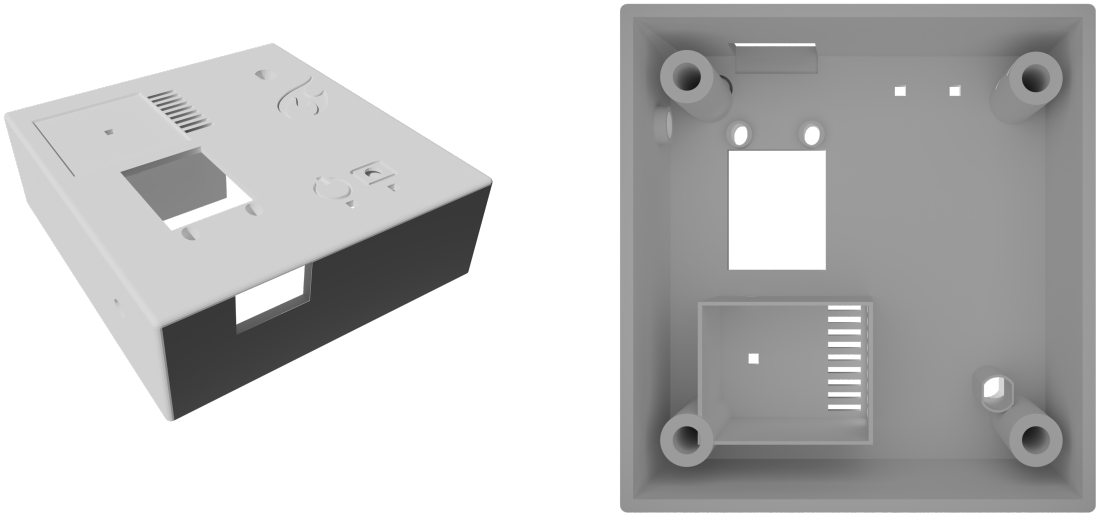
(a) Overview **(b)** Inside

Figure E.1: The casing design for the Generic device



(a) Overview **(b)** Inside

Figure E.2: The casing design for the Bedroom/Mirror device



(a) Overview **(b)** Inside

Figure E.3: The casing design for the Surgeon device