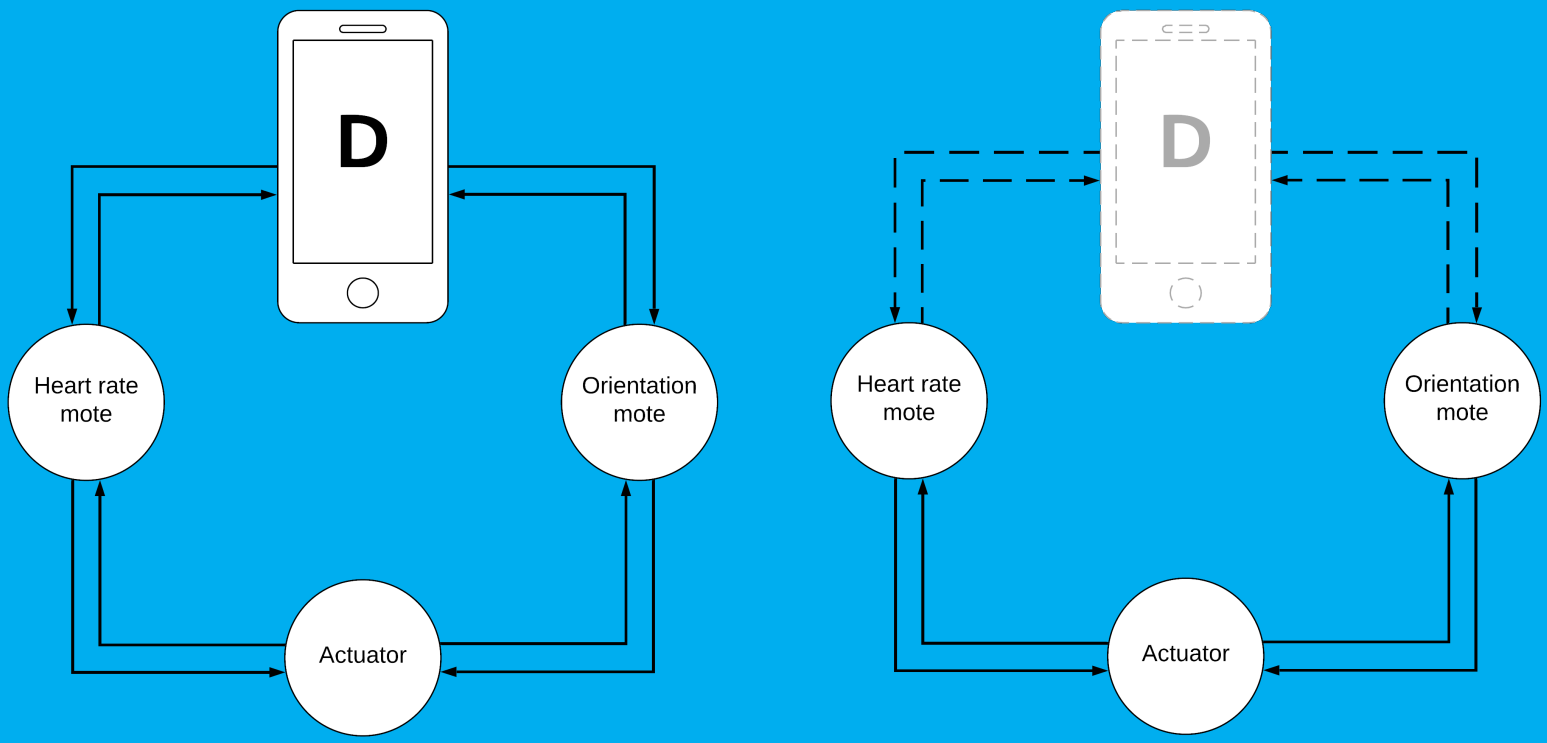
Bachelor Thesis

Seizure preventing MBAN

Joris Henstra
Pim Jansen



TU Delft

Bachelor Thesis

Seizure preventing MBAN

by

Joris Henstra
4500954

Pim Jansen
4537165

Commissioned by the Department of Neuroscience - Erasmus MC .

Supervised by Dr. ir. Arjan van Genderen.

For the bachelor graduation project EE3L11

First version - December 2019.

Revised - January 2020.

TU Delft

Abstract

Many people worldwide suffer from epileptic seizures and not all of them can be prevented using medicines, this thesis is being done for seizure prevention. This is based on implementing a medical body area network (MBAN) that takes sensory recordings across the whole body. This is needed as the research group that proposes this project found from previous research that this is needed for timely seizure prevention. In this report the design, implementation and results of building a prototype MBAN using Bluetooth low energy (BLE) is discussed. The result is a prototype that can measure heart rate when Bluetooth is turned off, but does function with Bluetooth when the heart rate sensor is replaced with mock data. Recommendations are made as to how to resolve the issues that arose during the implementation and as to which topologies should be implemented in the future.

Contents

1 Introduction

1.1. Motivation for research

Worldwide more than 50 million people suffer from epileptic seizures. An epileptic seizure is caused by an abnormal amount of activity in the brain and results in excessive shaking, uncontrolled muscle movements or even loss of consciousness [20]. Stafstrom and Carmant said: "Epilepsy is one of the most common and disabling neurologic conditions, yet we have an incomplete understanding of the detailed pathophysiology and, thus, treatment rationale for much of epilepsy."

1.2. Previous research

Nowadays medication is available that is able to reduce the amount of seizures or even prevent them. Unfortunately the existing medications do not work for 20-30% of patients [10]. Therefore alternative methods are researched to prevent seizures without the use of just medication. This is the reason that a project was commissioned by the department of neuroscience at Erasmus MC. In the commission they wrote the following: "Epilepsy is a medical condition which is identified as a period of health symptoms due to abnormally excessive or synchronous activity in the brain. Outward effects vary from uncontrolled shaking movements involving much of the body with loss of consciousness (tonic-clonic seizure), to shaking movements involving only part of the body with variable levels of consciousness (focal seizure), to a subtle momentary loss of awareness (abscence seizure). Most of the time these episodes last less than 2 minutes and it takes some time to return to normal." The neuroscience research group has done research on seizure prevention in the past. For one of their experiments they concluded that timely seizure prevention could be possible if sufficient sensory recordings are present. As epilepsy manifests itself across the whole body, the need for sensory data of multiple body functions is essential. Some of these body functions are: sweating, elevated heart rate, rapid muscle tone, synchronized neuron firing.

1.3. Proposed solution

Because of the need for sensory data of multiple body functions, the department of neuroscience commissioned a proof-of-concept implementation of a secure and reliable Medical Body Area Network (MBAN) for seizure prevention. The implementation of an advanced algorithm for seizure prevention itself is not an objective for this project. This project focuses on implementing the prototype for the MBAN. A Body Area Network (BAN) is composed of multiple nodes sharing information with each other. The range of the Body Area Network should be small or the BAN should be in close proximity of the body with little to no outward transmissions. Body Area Networks have different uses. One of these uses is monitoring of a patient's health it is then usually called a Medical Body Are Network (MBAN). An MBAN needs to be comfortable because it needs to be used for long periods of time in the order of days or weeks for the purpose of monitoring sensor data. Because of this most studies focus on applying Wireless Body Area Networks (WBANs) [11][13][18]. The BAN also needs to work with implantable and wearable sensor nodes and shall therefore be called an MBAN. The MBAN commissioned needs to work and simulate both wearable devices and implants so no manual input can be done after the "implantation". These sensor motes measure the body values of the patient. With these body values it is possible to predict when a epileptic seizure is going to happen. When the prediction

of a seizure is done it is possible to prevent this seizure with a brain implant that gives electric signals to the brain. Some challenges are: power consumption, connectivity and security.

1.4. Focus of the thesis

The project is divided in two subgroups. One subgroup will focus on the implementation of the of an Android app to use with the MBAN, this is called the application subgroup. The other subgroup is called the Network subgroup. This subgroup will focus on how to set up a simple MBAN with sensor and actuator nodes for seizure prevention based on a orientation monitor and a heart rate monitor. Additionally, the network subgroup will try to build the network in a robust and energy efficient way and allow for a connection with the Android phone. This will be done using the app made by the other subgroup. In this thesis the connecting, storing, measuring and processing of the data needed for a simple seizure prevention algorithm will be discussed. This thesis will also have a discussion on the implementation. Also will its results from the proof-of-concept prototype be presented. In the end a conclusion and recommendations based on the discussion will be made.

1.5. Research needed

In order to make such a network, multiple topologies have to be analysed to achieve energy efficient WBAN's [3]. Also to be taken into consideration is how data will be transmitted between the motes. Some previous work has been done in analyzing different transmission protocols and data gathering [9]. These protocols have to allow for secure transmission since the data is very sensitive and can cause harm if it were to be easily accessible to everyone.

2

Program of Requirements

2.1. Introduction

As stated in Chapter 1 it is very clear that currently there is no fitting solution available on the market for predicting and preventing epileptic seizures. There are some solutions available on the market, but most of these are medicine and these do not always work optimal for every patient. It can be concluded that these do not qualify as the most optimal solutions. An improvement on the current solutions would be a MBAN system with seizure prevention. The requirements for the prototype of such a system are discussed in this chapter.

2.2. Functional requirements

When one wants to market a new product on the market it should be economically affordable. Before a product is ready to be sold on the market, many design iterations should be carried out. The BAP project can be seen as the first iteration in the design process. The requirements, see Table 2.1, apply on the product built within the BAP.

Number	Name	Description
[F1]	Comfort	The patient should be able to wear the product without restraining him or her from doing their daily tasks.
[F2]	Scalability	The network should be expandable with extra sensor motes.
[F3]	Memory	The network should be able to store data in case the phone is not connected.
[F4]	Robustness	The network should be able to measure and intervene independently of the phone.
[F5]	Data transmission	The sensor motes should be able to send sensor measurements to the phone.
[F6]	Seizure detection	The actuator can be triggered by another mote.

Table 2.1: Functional requirements

2.3. System requirements

The system requirements are requirements that concern the main features with corresponding performance indicators. These are stated in Table 2.2.

Number	Name	Description
[S1]	Battery lifetime	A wearable battery should be able to supply the sensor with energy should last for at least 120 hours.
[S2]	Life expectancy	The product should at least have a life expectancy of 5 years.
[S3]	Heart rate	The heart rate sensor should measure the heart rate within an accuracy of 5 BPM.
[S4]	Orientation	The orientation sensor should give the angle with the ground to track the orientation of the with an accuracy of 5 degrees.

Table 2.2: System requirements

3

Hardware and software

3.1. Hardware choices

Before starting the implementation of the system in Chapter 6 some choices are made regarding the hardware and software to be used. The first to choose is the communication protocol as it gives restriction on what System on Chip (SoC) can be used. The chosen SoC on its turn decides which development board and sensors can be used. Finally the decision of software and libraries can be made by taking into account the previously mentioned aspects of the implementation. All hardware considered is small in order for it to become wearable in a future prototype (Program of Requirements Table 2.1 - F1).

3.1.1. Communication protocol choice

The communication protocols that were considered for this project are listed below. There are currently more communication protocols available than these three but the ones listed below link up the best with this project regarding their ease of implementation and hardware availability.

- ZigBee
- Bluetooth
- BLE

ZigBee is a communication protocol that is very easy to implement and work with. A very big disadvantage of this protocol however is that is consumes a lot of energy, something that is not permitted for the scope of this project [2][9]. Compared to the other two it is also not integrated in modern smart phones making it impossible to communicate with a phone without adding additional hardware to it (Program of Requirements Table 2.1 - F5). Another communication protocol that was considered is Bluetooth. This protocol is more difficult to work with but uses less energy than ZigBee. A protocol that uses even less energy is Bluetooth Low Energy. This is a protocol that supports sending up to 20 bytes of data per packet. This data size per package is enough for this project. Also BLE has been adopted by a lot of hardware nowadays which is a big advantage. This means that the sensor motes could communicate easily with a smart phone, something which would not be possible if ZigBee is used. Furthermore BLE is able to have up to seven devices connected at the same time making it expandable (Program of Requirements Table 2.1 - F2).

3.1.2. System on Chip

The SoCs listed below are the ones that were considered for this project. In this section these will be discussed.

- ESP32-WROOM-32D
- ESP32-WROOM-32U
- ESP32-WROOM-32
- ESP32-SOLO-1

- Laird BL652 series

All SoC's listed above are open-source, which saved a lot of time in the project. Also they have a lot of I/O pins and support a variety of bus protocols, which gives more freedom in sensor types. They also have ultra-low power capabilities (Program of Requirements Table 2.2 - S1).

SOLO-1 vs WROOM 32

First the main difference between the ESP32-SOLO-1 and the ESP32-WROOM-32 series will be discussed. The ESP32 series is from the company Espressif. The SOLO-1 is a single-core chip, whereas the WROOM-32 has a dual-core. Since the power usage of the SOLO-1 was similar to that of the ESP32-WROOM-32D when doing the same amount of computations, this was not a negative aspect for the latter [7][6]. The SOLO-1 was not chosen because it was not yet known how much computing power would be needed by each node and some room in this aspect could be useful.

Laird

Also from the company Laird the SoC BL652 should be considered. This is a very nice SoC since it also has an NFC component built in. Additionally this could possibly be used as a way to do key sharing for encryption. This SoC was not chosen because its development board was far too advanced [14]. Also it was not yet decided which way of encryption would be used or if it would be done at all, so the advantage of a built-in NFC component was not of crucial importance. With too many options that were not required for this project and less documentation it was discarded as an option.

ESP32-WROOM-32

In the end the ESP32-WROOM-32D was chosen, because the WROOM-32U does not have an antenna built in, which is very inconvenient for this project due to time restrictions. The ESP32-WROOM-32 is not chosen either, because it is an older version of the ESP32-WROOM-32D of which its antenna is worse. Also the antenna of the ESP32-WROOM-32D would lead to better RF performance [7][8]. The chip has flash memory which can also be used to store measuring data so it can either save data or send it via BLE (Program of Requirements Table 2.1 - F3 & F4).

3.1.3. Development board

Concerning the development boards none were compared. The choice made was to go with the ESP32-DevKitC. This development board has a build in DC/DC converter and its power supply is by use of a Micro USB [4]. Some big disadvantages of this board is that no debugging is possible, which was discovered only later on and there is no easy way to add a SD card for extra data storage. The ESP-WROVER-KIT does have these capabilities and would later be found to have been a better option. Unfortunately the number of other capabilities was somewhat overwhelming and deemed unnecessary thus it was not chosen [5].

3.1.4. Orientation sensor

Below two motion sensor ICs are listed that can be used for determining the orientation. These are the only ones that were considered for this project since these were the only two affordable and relatively easy to implement in the network.

- MPU-6000
- MPU-6050

These are motion sensors with a 3-axis gyroscope and a 3-axis accelerometer. This gives freedom to go for simple or advanced motion analysis. The MPU-6000 uses the SPI protocol, which uses less energy than the I2C protocol that is implemented on the MPU-6050. But since SPI is more susceptible to noise on a breakout board this is not a viable option for the prototype. But for a final product the MPU-6000 should be chosen. For the moment being the choice was made to order the MPU-6050 mounted on a breakout board because of the expected ease of use.

3.1.5. Heart rate sensor

The heart rate sensors that were considered for this project are listed below.

- AD8232 (ECG)
- ADS1293 (ECG)
- MAX30100 (PPG)

For the heart rate sensor both electrocardiogram (ECG) and photoplethysmogram (PPG) were considered. The former measures the hearts muscular activity by detecting the electric discharges the heart produces each heart beat. The latter represents blood volume changes by measuring changes in light absorption. For this project an ECG measuring sensor seems more appropriate to use, since it has lower power consumption and signal noise. But on the other hand it is more of an obstructing instrument to the patient's lifestyle whereas the PPG is a lot smaller and thus less intrusive. But also gives a less detailed view on how the heart is functioning and the measurements are more difficult to process onto a heart rate. When discarding the possibility of using a PPG due to previously mentioned disadvantages, a sensor that outputs an ECG the AD8232 seems more appropriate since it uses less power (on full operation) then the ADS1293 does. They use 170 μA for the AD8232 and 960 μA for the ADS1293 [1][21].

3.2. Software choices

As programming environment Eclipse was chosen in combination with Arduino libraries on top of the tool chain that comes with the ESP32. The reason Eclipse was chosen is that this was the recommended programming environment from Espressif and gives more functionality and Git integration than the Arduino IDE. In general all source code for such chips is most efficient if it is written in the programming language C, but because there were a lot of Arduino libraries that encapsulate the full functionality of the chip the decision was made to use these instead. Since these are also written in C++ it was concluded that this was an easier way to program, furthermore if it came to light that C++ did not have enough capabilities it would still be possible to switch to the C programming language gradually in the future. A disadvantage of using the Arduino libraries is that they are not very energy efficient compared to using bare C.

4

BLE Network

In this chapter the network topology will be discussed. During the project multiple possible topologies have been explored. These will be elucidated in Section 4.1. Next the implementation of how the sensor network currently functions will be explained in Section 4.2. The results will be shown in Section 4.3.

4.1. Design

Several topologies have been developed and in this section these will be illustrated. The four designs shown in Figure 4.1 have been examined. These topologies will be referred to as topology A, B, C and D respectively. The reason these topologies were examined is, because in each of these topologies the sensor motes have as few connections as possible. The advantage of this is that these are the motes most accepted fail. In Figure 4.1 the setup is shown of every topology with (top illustrations) and without the phone connected (bottom illustrations). Some basic information about BLE is provided in Section 4.1.1.

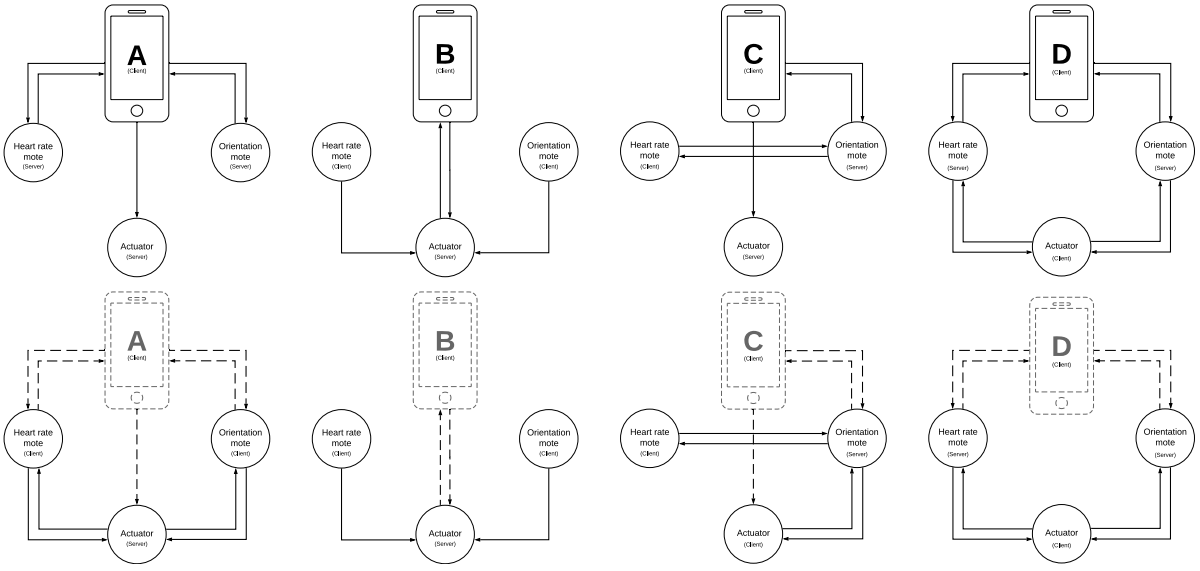


Figure 4.1: Overview of the considered network layouts

4.1.1. Basic knowledge BLE

Some basic knowledge is required to understand what is required to setup a BLE network. The most simple BLE network consists of a server and a client. The server will advertise data which can be read or written on by a client. A server is grouped into services, each of which can contain zero or more characteristics. These characteristics may consists of zero or more descriptors. A visualised example of how this looks is given in Figure 4.2. A service for example could be a battery service. Within this service several characteristics can

exist like a characteristic for the battery level, another characteristic for the number of total cycles of the battery and so on. These characteristics can have a descriptor, which can give extra information on the data that is sent on the characteristic. For example it can be a flag, which can enable clients to be able to observe server initiated updates. To summarise, a server can advertise data to clients. A client itself can not advertise data. It can however connect to a server and then read the data that a server advertises. The client can also write another value on a characteristic of a server, if this is allowed by a server. This is a setting in the setup of a server. In all of the possible topologies discussed in this report the phone functions as a client in the network. This is because the phone is not able to setup a connection with the role as server.

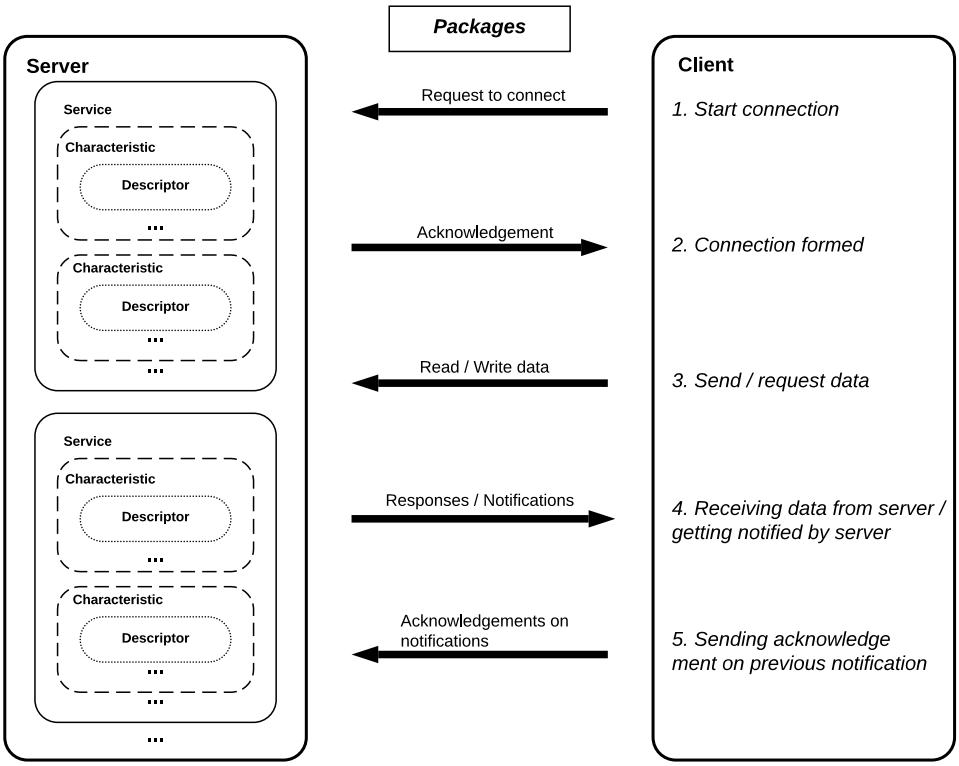


Figure 4.2: Hierarchy of a BLE server

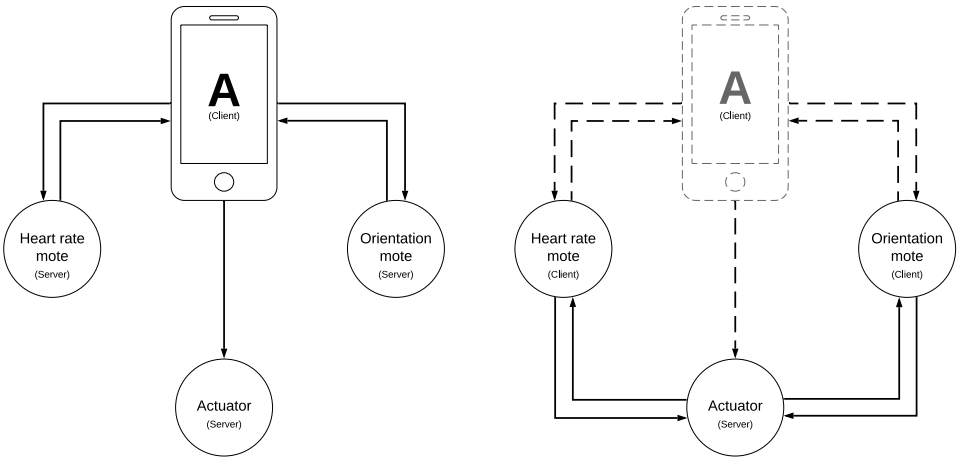


Figure 4.3: Network topology A

4.1.2. Topology A

In topology A the phone is directly connected to both sensors. In this topology both sensor motes will be a server. The actuator will also be a server when the phone is connected. However, when the phone is disconnected the actuator will switch from being a server to a client. Why this is, will be elaborated on in the next paragraphs.

Phone connected

When the phone is connected to the network it reads the sensor data of both sensors which is set on the corresponding characteristic by each of the sensor servers. The phone analyses the data and when the data of both sensors is critical it will send a flag to the actuator. This will only be possible if the actuator functions as a server on this moment, since if the actuator is a client, no data can be written to the actuator. When the phone first connects to a sensor it will write to the sensor that made the connection, in this case the phone. This is needed so that the sensor knows how to process and send its data.

Phone disconnected

When the phone is disconnected from the network the actuator will take on the task of analysing the data from the phone. Before this is possible the actuator has to switch its role within the network from being a server to being a client. Now it can connect to both sensors. In comparison to the phone, the actuator does not continuously read the values of the sensors. The sensors use an option in BLE called notify. In order to use the notify function within BLE, a descriptor has to be added to the corresponding characteristic. Now the actuator does not continuously have to 'read' the characteristic on which the sensor data is available, but the actuator can simply 'listen'. When a critical value is measured by a sensor, the sensor will notify the actuator and send the corresponding sensor value. Now the actuator can know when both sensor values are critical and undertake action if desired. When the actuator first connects to a sensor it will write to the sensor that made the connection, in this case the actuator. This is needed so that the sensor knows how to process and send its data.

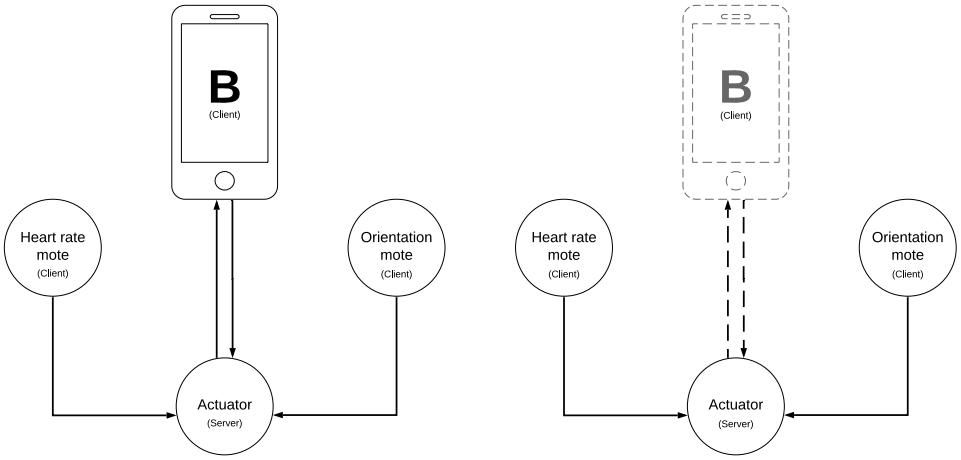


Figure 4.4: Network topology B

4.1.3. Topology B

In network topology B the only mote with which the phone is connected is the actuator. The actuator will be a server in this topology. This server would have at least two services, one on which the heart rate mote could write data and one on which the orientation mote could write data. Both of the sensors would be clients of the actuator. Since they are clients instead of servers in this topology it is not possible for the sensors to have a direct connection with the phone. A big advantage of this topology is that the sensor motes would never have to advertise data themselves, which saves a lot of energy on the sensor motes. All they have to do is establish a connection with the actuator and write their data to the actuator. However, a big disadvantage of this topology is that when the battery of the actuator is dead or if the mote gets broken the entire network will no longer be functional.

Phone connected

When the phone is connected to the network its only connection will be with the actuator mote. Both sensor motes will write their data on the corresponding characteristic of the actuator. The actuator then sends this data to the phone. The data is processed on the phone. When this data proves to be critical a signal will be sent to the actuator by the phone. The actuator can then act on the critical data.

Phone disconnected

In comparison to topology A when the phone is disconnected from the network, Section 4.1.2, all the roles within the network stay the same, meaning none of the mote changes from being server to client or vice versa. The data will now be processed on the actuator, with a more basic algorithm. In contrast to topology A, the data will always be written on the actuator by both sensors. The sensor data, from both sensor motes, will be saved on the actuator mote. If the phone comes back online in the network all this data will be send to the phone. When the data would be saved on the sensor motes instead of the actuator, this data then first has to be sent to the actuator, be processed here, and then forwarded to the phone which would take too much time.

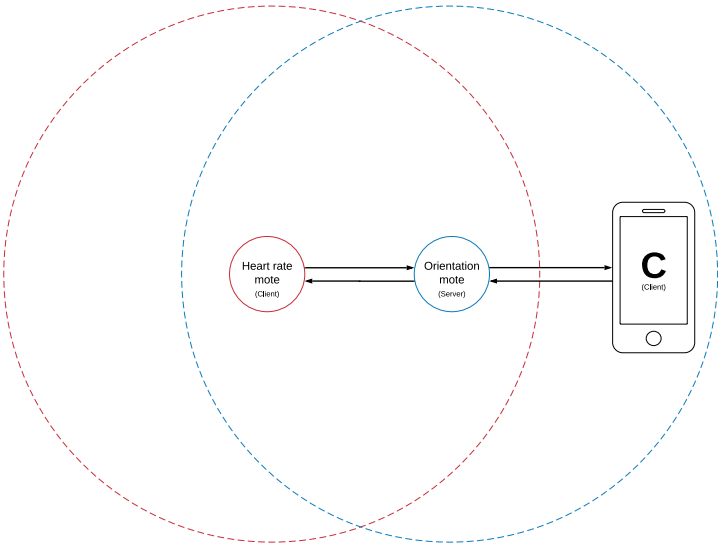


Figure 4.5: Relay motes

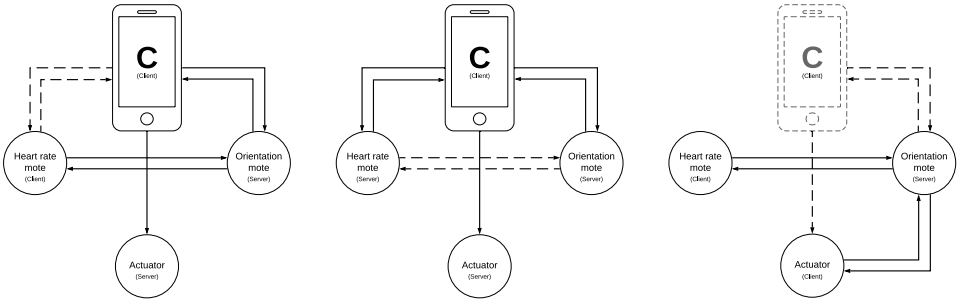


Figure 4.6: Network topology C

4.1.4. Topology C

In this topology both sensor motes and actuator will switch between acting as server and client. This topology uses the principle of relay motes. A relay mote is a mote that forwards the data from another mote. By forwarding the data of a mote, the transmission power can be reduced without loss of functionality of the network [3]. As can be seen in Figure 4.5, the heart rate sensor is not able to reach the phone directly. However, the orientation mote is able to have a connection with the phone because it is in range. The heart rate mote is able to connect to the orientation mote and send its data via the orientation mote to the phone, using

the orientation mote as a relay mote. The heart rate mote uses the orientation mote only when it is out of range from the phone. When the phone is back in range of the heart rate mote a direct connection will be established. This can be seen in the middle illustration in Figure 4.6. In this topology it is also possible for the heart rate mote to act as a relay mote, but to keep things clear this is not shown here.

Phone connected

When the phone is connected it receives the data from both sensors, processes it and sends a signal to the actuator if the data is critical. The phone may receive the measurement data from both motes from only one mote. As can be seen in Figure 4.6, the phone receives the measurement data from the heart rate mote also from the orientation mote. The heart rate mote sends its data to the orientation mote, which on its turn forwards this data to the phone.

Phone disconnected

In case the phone disconnects from the network and the heart rate mote is out of range for the actuator, the actuator will make a connection to the orientation mote. Otherwise the actuator connects to both motes. The measurement data will only be sent by the motes when it is critical. The motes however do save their data When the phone reconnects to the network this data is sent to the phone. Either this is done in a direct connection or a relay mote is used.

4.2. Implementation

4.2.1. Network topology

After reviewing all advantages and disadvantages of every of the previously mentioned three topologies A, B & C. The topology that was found best was topology A. For this topology however the sensor motes should be able to switch functionality within the network from server to client and the other way around. Due to time restrictions it was not possible to implement this. Therefore a different topology was made in which this was not necessary. This is topology D, this topology can be seen in Figure 4.7. In this topology both sensor motes act as servers in the network and the actuator acts as a client. The phone will have a direct connection to both sensor motes. Furthermore, both sensors will have a connection with the actuator.

Phone connected

When the phone is connected to the network it first receives stored data from the sensor motes. When the maximum number of samples are saved (which is 682), it takes about 4,4 seconds to send all samples to the phone. When a sensor mote has sent all its stored data it will start sending the current measurement values. The phone processes this data and when the data appears to be critical it will send a flag to the actuator. This is done by sending a message via the heart rate mote or the orientation mote to the actuator. The reason the phone does not send a message directly to the actuator is because the actuator functions as a client in the network. Since the phone is also a client they can not establish a connection.

Phone disconnected

When the phone is disconnected both sensors save their data and only send data to the actuator when it crosses a certain threshold. The actuator then processes the data with a very simple algorithm. When both sensors appear to have critical data the actuator will turn on an LED.

4.2.2. Business rules of the network

Actuator mote

The behaviour of the actuator mote will be discussed in this section. The flowchart of the actuator can be seen in Figure 4.8. The beginning of the flowchart is the circle on top with the text 'Actuator powered on' in it. After the actuator has been powered on it will immediately start scanning the environment to see if the heart rate mote and/or the orientation mote are powered on. If one of the two motes is powered on the actuator will make a connection to this mote and start data collection from this mote. The actuator will then try to scan three times for the remaining mote. If this fails the actuator will stop scanning for the remaining mote for 5 minutes. After this time it will restart the scanning for three times and so on. However, if before these 5 minutes have passed the end user has for example replaced the battery in the yet unconnected mote and he or she wants the actuator to connect to this mote directly it is possible to press a button on the phone which then will send a signal to the actuator, via one of the sensor motes, to start the scanning before the 5 minutes

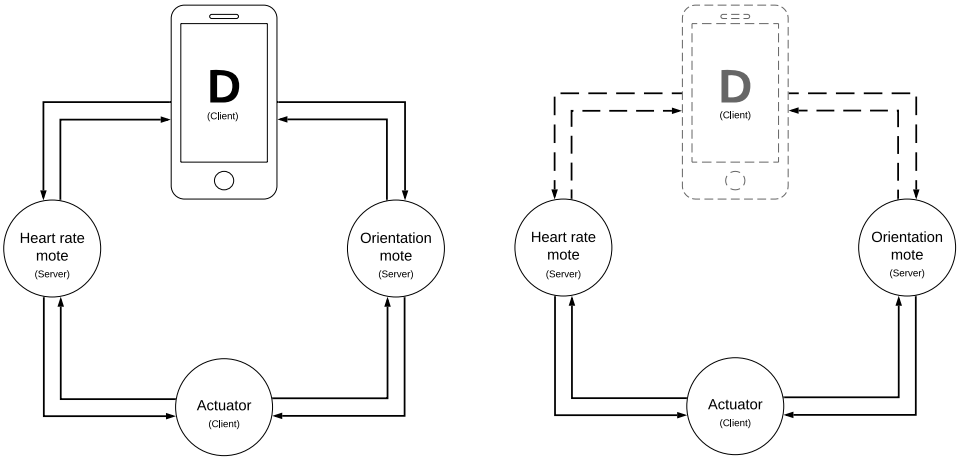


Figure 4.7: Network topology D

have passed.
When both sensors are connected and they both send critical data, the actuator will turn on a LED. This LED will turn off when the data of one of the sensor motes is no longer critical. If one sensor disconnects, the actuator will end the connection with both sensor motes. The actuator now tries to scan and connect to both motes again. The reason why the actuator disconnects from both motes when only one connection is lost was not clear. A lot of time had been spent on finding out why this is the case. But within the time space available for this project, the choice was made not to investigate this any further.

Sensor server

Figure 4.9 provides an overview of the server. As can be seen from this figure, the sensor server consists of three different services. The device information service is used for sending information about the device, within this service there exist two characteristics.

1. The first characteristic is for transmitting the device name of the sensor. This is done so other devices know with what device they have connected. Also when a device connects to a sensor server from the network it writes on this characteristic who he is. For example when the phone connects to the heart rate mote, the phone will read on this characteristic the value 'Heartrate Sensor'. The phone now knows it has connected to the heart rate mote. To let the heart rate sensor know who has connected to him the phone writes 'Phone' on this characteristic. Now the heart rate mote knows that the phone has connected. The characteristic is now reset to 'Heartrate Sensor' by the heart rate mote. In case the sensor server has a connection with two clients and a connection is lost. The sensor mote will ask the client which is still connected who he is. This is done by using a notify message. A notify message is like a normal message when it is send to a client. Normally the client has to continuously monitor a specific characteristic whether its value has changed. However when a notify message is used the client does not have to continuously monitor the characteristic, but gets a 'Heads up' from the server stating that the value of the specific characteristic has changed. In order to receive these notifications the client does have to register for them. This is done by sending a message to the server when the connection is first established.
   Within the network the actuator is registered for notifications, a big advantage of this function is that the actuator does not have to read the characteristic continuously, which saves energy and processing power. But it only reads the characteristic when it receives a notification. As can be seen in Figure 4.9 this characteristic has a descriptor. This descriptor is needed to send notifications to clients.

2. Another characteristic within the 'Device information service' is the 'Connect characteristic'. As explained in the subsection 'Actuator mote' in Section 4.2.2 the actuator tries to find all sensor motes for three times. If it fails to find these sensor motes within these three times the actuator will not try again to find the remaining sensor motes for five minutes. This is quite inconvenient for the end user of the product, since when a battery has to be replaced he or she has to wait for five minutes before the sensor will come back online in the network. To prevent the sensor from being offline for these five minutes, the user can go into the app and press a button. Since the phone can not communicate directly with

the actuator mote, the communication will go via the heart rate mote. When this button is pressed the phone will write 'TRUE' on the 'Connect characteristic' of the heart rate mote. The heart rate mote notices this and forwards this message to the actuator, this is done via a notify message.

The next service is the 'Sensor measurement service' this service only has one characteristic. This service is used to send the sensor measurements. The characteristic of this service has the same descriptor as the 'Connect characteristic' so the mote is able to send notifications. This is because when the phone is not connected to the network the actuator will only receive data from the sensor motes when their data is critical. So now the actuator does not have to read this characteristic continuously.
The last service that is used within the sensor server is the 'Time service'. This service has one characteristic which is the 'Current time characteristic'. When the phone connects to a sensor mote it will write the current time and date on this characteristic. This is used by the sensor motes to set their system time correct. And also to append timestamps to the data that has been saved on the sensor motes. Since every second a measurement is saved on the mote these timestamps can be appended using backwards iteration.

Sensor motes

Within this section the behaviour of the sensor motes in the network will be discussed. The sensor motes behave as the flowchart in Figure 4.10 shows. The beginning of the flowchart is the circle that states 'Sensor powered on'. When the sensor is powered on some initialisation procedures take place, before any connection can be established. These initialisations conclude the creation of a BLE server, with its services, characteristics and descriptors, see Section 4.1.1 for explanation of these terms and read the subsection 'Sensor server' in Section 4.2.2 to see how the server is implemented in this network. Once these services and characteristics have been created the default transmission power is set to its lowest possible value. The sensor server has now been fully initialised and starts advertising. When the server is advertising, it sends packages which can be received by for example the phone or the actuator. Now the phone and actuator now that the sensor mote is alive and react to these messages, by for example asking to establish a connection.
Almost always a connection will not immediately be established with a client. The sensor mote will then start saving data. This is done in the following order of events.

- First the sensor mote will get the sensor data from the sensor, this is done every second.
- When there are no connections or when the actuator is connected but there is no critical data present the measured data will be saved in the flash memory of the ESP32. However when the phone is connected or when the actuator is connected and there was critical data present less than five measurements ago the data will be written on the corresponding characteristic.
- If the phone is connected this is great since then the data does not have to be saved on the sensor mote. However if the phone is not connected it was the actuator who is connected and the actuator has to be notified that new data is present.
- Also the data still has to be saved. Now the system waits until a second has passed since the last measurement data has been taken. When this second has passed the sensor mote takes a new data measurement and starts the circle over again.

In case a device disconnects it depends on how many connections there are still left before it is decided what is done next. If there is one connection left this means there previously were two connections. The sensor mote now needs to know with whom it is still connected, so it asks for a device name update from the device which is still connected. This is done by writing 'Heartrate Sensor Update' or 'Orientation Sensor Update' on the characteristic for the device name. This is done by also sending a notification, since when the actuator is connected is does not continuously have to check whether the characteristic has changed. First the advertising will be restarted, because previously two connections were present, which is the maximum number of connections a mote can have in this network. When a connection however is lost, the advertising should be restarted again. Since this is needed to make a new connection. After this is done the sensor mote will check again with which client it is still connected, since now the update has been given by the corresponding client, and goes on from there. When there are no more connections left in case of a disconnect event the sensor will directly start saving the sensor data.
It can happen that a sensor mote receives a connect flag from the phone. This flag is meant for the actuator mote and has to be forwarded to the actuator. This is done by notifying the actuator mote of this. When this is done the sensor mote continuous as it would normally operate.

4.3. Results

4.3.1. Actuator

In Appendix A.1 a piece of the actuator terminal prints have been added. Here can be seen how the actuator functions as a result of the business rules of the network, which was explained in Section 4.2.2. As can be seen in the terminal prints of the actuator first the connections are established. Then the actuator receives critical data from both sensor motes and detects that there is a seizure present. When the data of one of the two sensor motes is no longer critical the actuator notices this and prints 'SEIZURE IS OVER'.
In the second terminal print list which is added in the Appendix A.1 it can be seen that the actuator fails to find the orientation mote within the first three times. Then receives the flag from the phone to try to connect again to the orientation mote and succeeds to do so.

4.3.2. Sensor motes

In Appendix A.2 & A.3 the logs for both sensor motes are shown. As can be seen both sensor motes work as they are supposed to according to the business rules of the network, to see the business rules of the network see Section 4.2.2. Both with the heart rate mote and orientation mote the data is correctly stored and not sent until the phone is connected. Then this data is transmitted with their correct timestamp to the phone. In the terminal prints from the heart rate mote it can be seen that it has received a connect flag from the phone which means the heart rate mote should notify the actuator of this event. Which is also correctly done.

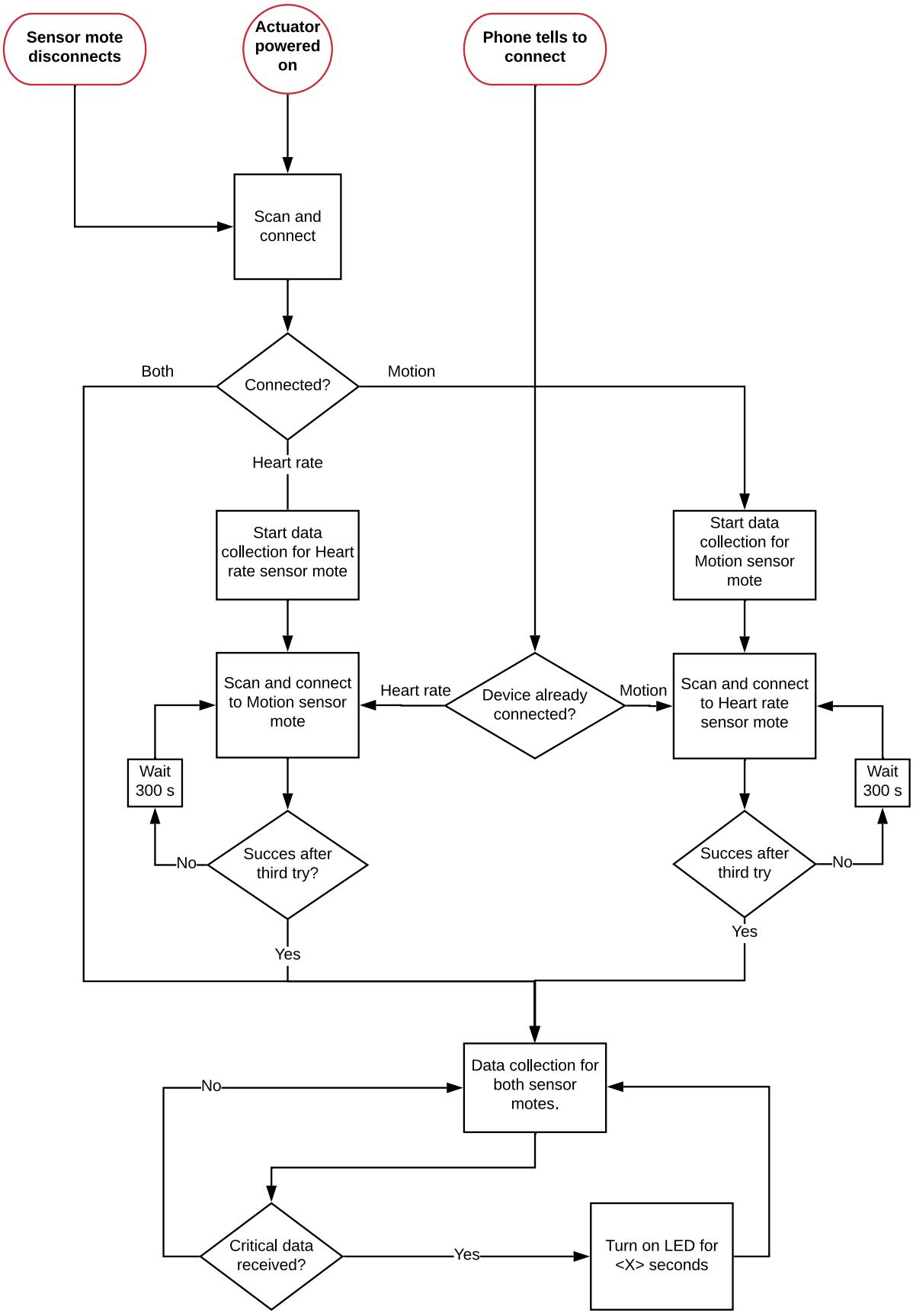


Figure 4.8: Flowchart actuator mote

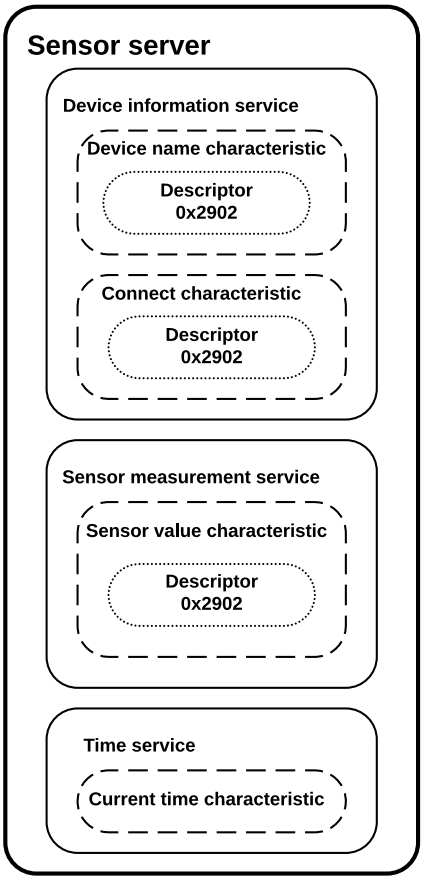


Figure 4.9: Hierarchy of a sensor server

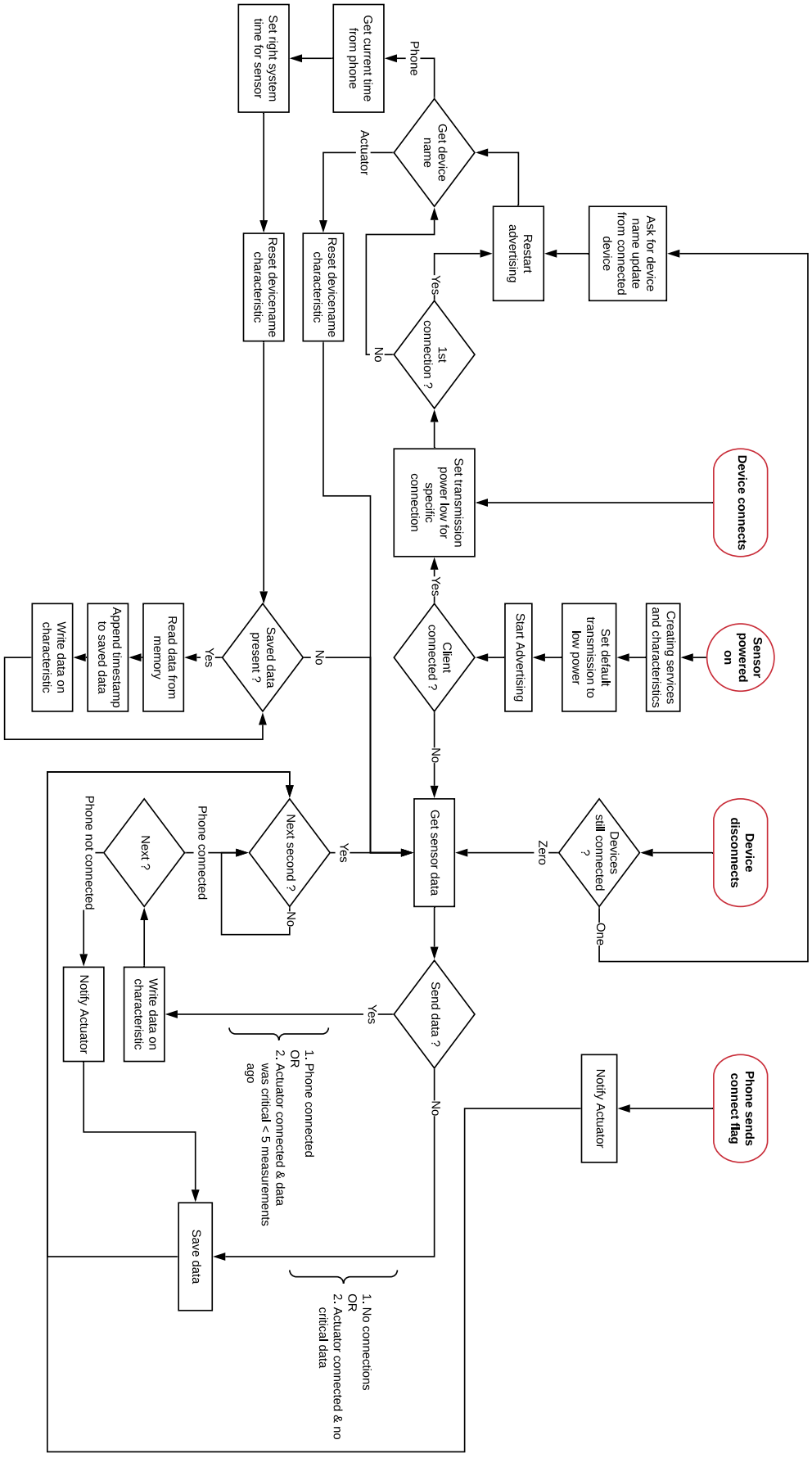


Figure 4.10: Flowchart sensor mote

5 Sensors

5.1. Design

The sensors will consist of a heart rate sensor and an orientation sensor measuring the orientation of the sensor. The heart rate sensor will have to detect when the users heart rate is elevated. For this system a heart beat of 120 beats-per-minute (BPM) is considered as elevated therefore a resolution of 5 BPM can be considered as sufficient. The orientation sensor will have to detect when the sensor is close to horizontal position and thus a resolution of 5 degrees could be considered sufficient. For both sensors some measurements from the output of its respective breakout board have to be done and then processed to give the desired measurement values as shown in Figure 5.1. These values are the heart rate in BPM and the angle with the ground in degrees.

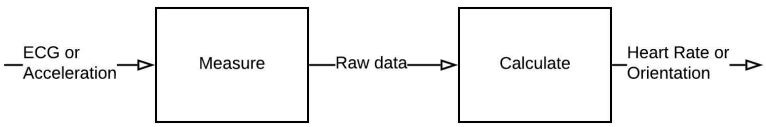


Figure 5.1: Data flow chart for a sensor mote.

5.1.1. Heart Rate

For measuring the heart rate an electrocardiogram (ECG) is used which is measured using electrode pads placed on the body as described in the documentation supplied with the breakout board [19]. This gives an electric potential difference of a few millivolts on the placed electrodes which represents the electric activity of the nerves in the heart. The AD8232 breakout board then amplifies this signal while rejecting dc offset[21]. The ECG is a function of voltage versus time often used for characterising cardiac activity like heart rate or rhythm abnormalities. It contains some characteristic wave shapes that represent the polarization and depolarization of different parts of the heart. The QRS complex is one of the features of the ECG and it represents the depolarization of the ventricles which are the large chambers of the heart and are the largest muscle masses of the heart and thus giving the R-peak of the QRS complex greatest amplitude of the ECG. This peak lends itself best for detecting a heart beat. The waveform of an ECG and the R-peak distance is shown in Figure 5.2. A frequency detection algorithm is designed which will use a transformation from the ECG in the time to the frequency domain and detecting the heart rate frequency based on its outcome. The calculation is done one the ESP32 and is thus written is C/C++. The reason for using this algorithm as opposed to other existing algorithms is to make an attempt at making an energy efficient yet continuously measuring sensor that is more robust to noise. Also this algorithm could implement real-time evaluation of the found value by comparing the magnitude of the frequency component with the rest of the components of the frequency spectrum. The algorithm demands that the ESP32 will make use of exact periodic sampling of the analog signal and an algorithm to detect the heart rate frequency component which corresponds to the heart rate frequency.

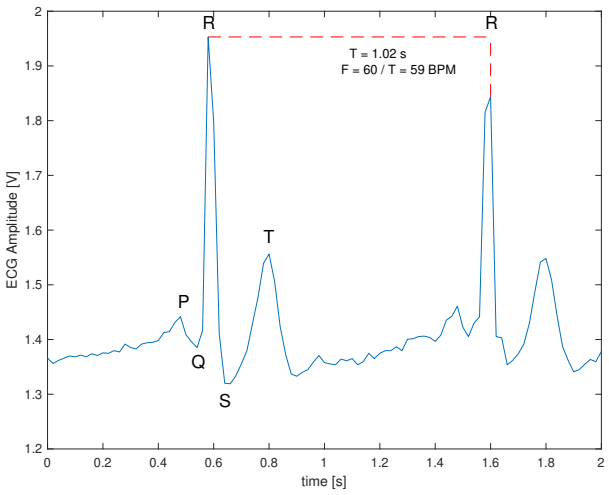


Figure 5.2: ECG waveform measured by ESP32 and AD8232 with R-peak detection heart rate.

Filtering and Amplification

By inspecting the AD8232 datasheet [1] "Cardiac Monitor Configuration" on page 25 and the Schematics [12] for the SparkFun AD8232 breakout board it is clear the board is an implementation of that circuit. This also means that it will not compensate for movement artefacts. The breakout board passes the ECG signal to its output following the power frequency response in Figure 5.3. The amplification of interest in the ECG signal is in the range of 50 bpm (beats per minute) or 0.8 Hz to 25 Hz being half the sampling frequency. In this range the power amplification is greater than 60 dB and the voltage amplification is thus greater than 1000. As a result the output voltage is comparable to the analog-to-digital (ADC) converting range of 3.3 V of the ESP32 [7] and thus making optimal use of its measuring resolution.

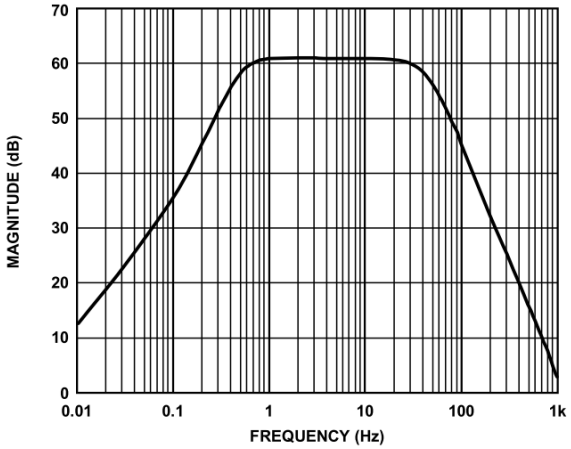


Figure 5.3: Frequency response from input to output of AD8232 in Cardiac Monitor Configuration.

Sampling

The chosen sampling frequency was to be as low as possible to reduce the amount of data measurements, storage and processing to achieve low power consumption. As most of the signal power of an ECG is located in the higher harmonics of the signal [17] a sampling frequency of 50 Hz is used which is sufficient to distinguish the heart rate even between adjacent heart beats [15]. This also satisfies the Nyquist sampling theorem for detecting a signal up to 250 BPM or 4.2 Hz which requires a sampling frequency of at least 8.4 Hz.

Calculation

The data is sampled periodically and thus it represents a time series of measurement values. When converting this to the frequency domain using the Fast Fourier Transform (FFT) this will result in a frequency domain dataset. In this dataset the magnitude values will be projected onto a frequency axis and the magnitude is the total energy of the signal in a bin (interval around that frequency). The bin size in this frequency axis will be determined by the number of samples used for the FFT algorithm and the used sampling frequency. This bin size will therefore be the frequency resolution (FR) at which the signals frequency components can be distinguished from each other and is determined by Equation 5.1. To acquire a resolution smaller than 5 BPM or 0.083 Hz a measurement time of at least 12 seconds is calculated using Equation 5.1 This equation can be rewritten to $T_m = \text{FR}^{-1} = \frac{1}{5/60} = 12\text{s}$.

$$\text{FR} = \frac{F_{\max}}{N(\text{bins})} = \frac{F_s/2}{N(\text{samples})/2} = \frac{F_s}{F_s \cdot T_m} = T_m^{-1}[\text{Hz}] \tag{5.1}$$

where:

$F_{\max}$ = highest frequency component present in sampled signal.
F_s = sample frequency
$N(\text{bins})$ = number of bins in the FFT result
T_m = duration of time measured in seconds.

5.1.2. Orientation

The orientation sensor will use the accelerometer function of the MPU6050. This gives a measurement on the acceleration in the x-, y- and z-axis. This will be transformed to the orientation of the sensor in degrees in relation to the relative position with the floor surface at the time it was calibrated.

Orientation calculation

The measurements made are represented as a vector $\hat{a} =< a_x, a_y, a_z >$ with a_x, a_y and a_z being the accelerations in direction x, y and z respectively. As these components are a projection of the vector $\hat{a}$ on the respective axis this means that the magnitude of the y component a_y contains the information on the angle with the ground and is found by using Equations 5.2 and 5.3. Using this algorithm does not take movement into account in the axes but only rotation around its origin. As movement would give additional acceleration on the axes it means that more advanced motion analysis has to be done to compensate for them.

$$\rho = \|\hat{a}\| = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2} \tag{5.2}$$

where:

ρ = Magnitude of the acceleration of the sensor [ms^{-1}].

$$\alpha = \arccos(-a_y/\rho) - 90° \tag{5.3}$$

where:

α = Angle relative to the floor [°].

5.2. Implementation

5.2.1. Heart Rate

Sampling

To achieve precise periodicity for the sampling timer, interrupts were used instead of using delays. One set to go off every 20 ms. It will then have to store the analog value measured by the 12-bit analog-to-digital converting (ADC) GPIO pin on the ESP32 to measure the output signal of the Cardiac Monitor breakout board. The other will go off every 1024 measurements so had to be set to 2048 ms and triggers processing of one data window.

Fourier Analysis

Memory space is limited as it is desired to leave as much memory for local data storage. So a Radix-2 FFT algorithm is used as it is most memory efficient. Also there is an open source library for it that can be used with the used Arduino libraries and C/C++ cross compiler as it was written in C. Disadvantage will be that it takes more than twice as much calculations and consequently also more energy than the fastest alternative, the Fast Hartley Transform [16]. The algorithm requires that the number of samples used for the algorithm should be a power of two. As interpolation would take more energy and time, the measuring time and frequency were tuned so as to the most usable samples in as little time. Therefore the measurement time is set to 20.48 seconds which is larger than the 12 second minimum given in Section 5.1.1. With the sampling frequency set to 50 Hz this gives 1024 samples per calculation window for the algorithm and a resolution of 2.93 BPM (Equation 5.1).

Processing

The heart rate is calculated by using the most recent 1024 samples every 1.28 seconds or 64 samples fitting sixteen calculation intervals (blocks) in the 20.48 seconds of measurements used per calculation. A data array is filled by pushing measurements on top of the older measurements and when it is full it starts at the bottom again. It has a size 64 samples larger than 1024 samples making it possible to store new measurements in one of the blocks of 64 samples while at the same time using the other 1024 samples for FFT calculation. The to be used samples for calculation are selected and inserted in the FFT object by shifting the calculation window through the data array for which a example is shown in Figure 5.4. In essence this enables for a FIFO data array enabling continuous calculation and measuring and preventing data collisions between the sampling and processing functionality. This implementation implies that the first 16 calculations or 20.48 seconds after starting up are not reliable as most of the data is not initialized yet. After the FFT calculation is executed, the greatest magnitude frequency component is detected and the beats per minute value is calculated. Real-time evaluation on the validity of the found value by comparing the magnitude of the frequency component with the rest of the components of the frequency spectrum is not implemented in the final prototype due to time restrictions.

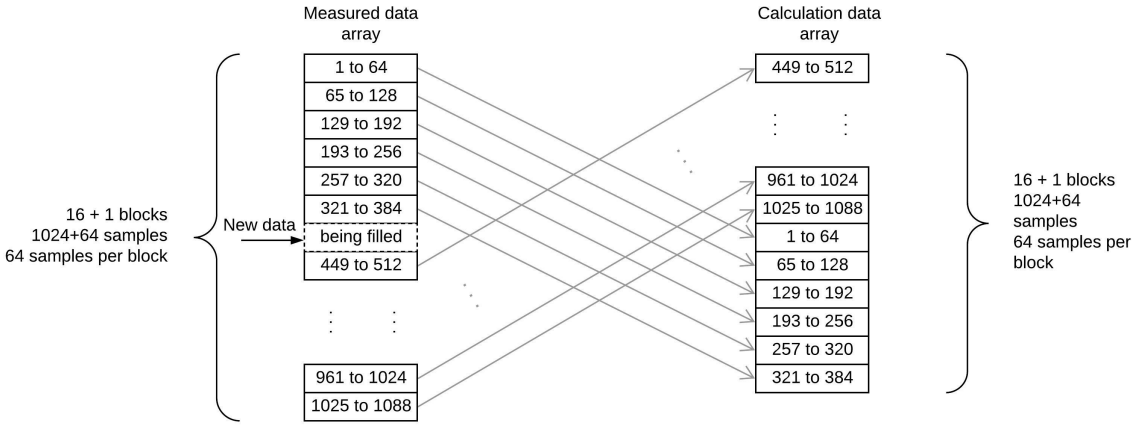


Figure 5.4: Shifting register for when the pointer to the top of the data array is between 384 and 449.

5.2.2. Orientation

An open-source library is used to implement the connection with the board and value readouts and interpretation from read out bytes to acquire sensor values.

Connection

The board uses I^2C serial protocol to transfer data back and forth between the ESP32 and MPU6050. So in order for the ESP32 to make sure it has a working data connection with the MPU6050 it first scans for all device addresses and then uses the address of a found working connection from that point on.

Calibration

When the breakout board is in neutral position the gravitational pull of the earth will give a measurement of 9.83 ms^{-1} on one of the axes. By inspection of the board this is determined to be the y-axis. So when in neutral position the acceleration values should be 0, 9.83 and 0 ms^{-1} for the x-, y-, and z-axis respectively as the y axis of the sensor is pointing upwards. The device starts by taking multiple measurements and averaging these to prevent noise from influencing the calibration. Then a calibration offset is stored for all axes which is used to correct the later measured values.

5.3. Results

For the heart rate monitor the algorithm was compared to R-peak detection heart rate calculation in a MATLAB simulation found in Appendix B.2. For the orientation sensor a simple test was done to verify realistic measurements. The simulation executes the same algorithm as done on the ESP32 when in operation and is done only to evaluate the performance of the algorithm compared to R-peak detection heart rate calculation.

5.3.1. Heart Rate

The MATLAB simulation uses measured data with the ESP32. The measurement is done after doing a few heart rate elevating exercises like squatting repetitively. It tracks for 55 seconds at the same sampling frequency as used in the algorithm when the user is not moving and therefore excluding motion artefacts from the analysis. The data was imported into MATLAB to simulate the algorithm and compare it to a rudimentary R-peak detection which was manually checked for errors and corrected if needed. The R-peaks are then used for determining the time in between heart beats and the heart rate is calculated from it. Both algorithm results are displayed in Figure 5.5. It is clearly visible that the first 20 seconds are indeed not corresponding to the physical heart rate due to the data array not being filled with measurements yet. After that it deviates less than 10 BPM from the R-peak algorithm and the average error is 5 BPM. The changes in amplitude and fluctuation seen in the ECG and R-peak detected heart rate respectively are due to breathing.

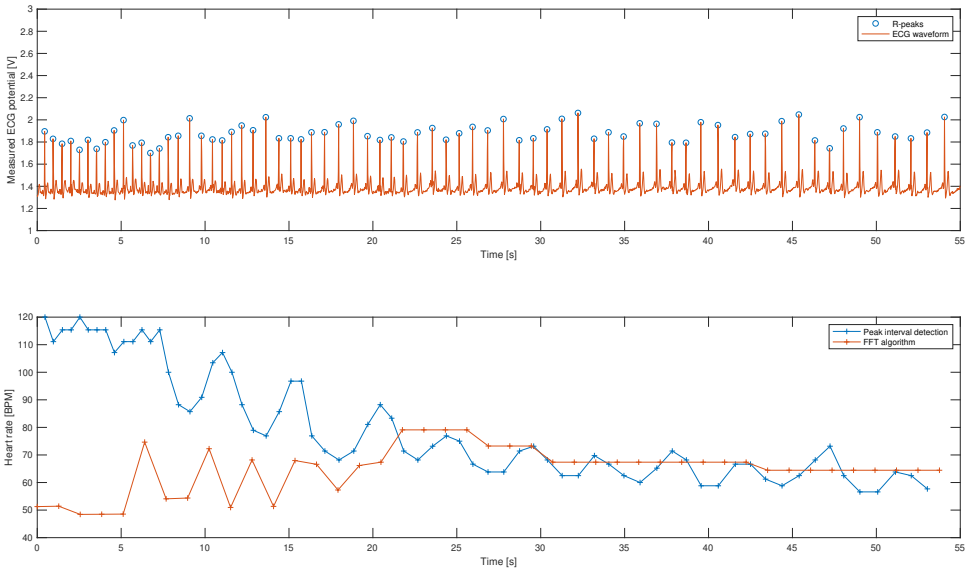


Figure 5.5: ECG waveform measured by ESP32 and AD8232 with R-peak detection heart rate compared to FFT algorithm.

5.3.2. Orientation

After starting up and thus automatically calibrating the sensor, the readings stayed on exactly 90 degrees as expected. When putting the module on its side by rotating exactly 90 degrees on an axis in the x-z plane checking with a protractor triangle the readings showed no greater deviation than 2 degrees from the expecting 0 degree when the sensor was stationary. This test was done to examine if the sensor would correctly

measure the angle between the sensors y-axis and ground.

6 MBAN System

In this chapter the merging of the BLE Network with heart rate and orientation monitor functionality is described. After explaining the design and implementation of the motes as a whole, the working and lacking functionalities are discussed and results are shown concerning the power usage.

6.1. Design

For combining BLE with heart rate and orientation monitors, the variables and functionality of the monitors are encapsulated as much as possible in separate classes which can be reused. This is done to make parallel development of both separate functionalities of the Mote possible and only combining it when both were sufficiently developed. A description of what processes are executed when and where in parallel when only one sensor mote and actuator are powered up is shown in Figure 6.1. It shows how both sending combined with storing and acquiring measurement data is separated as much as possible and that one does not rely on the other to function. When two sensor motes are in the network the functionality of the processes is in essence the same as in Figure 6.1 but for simplification it is not shown here.

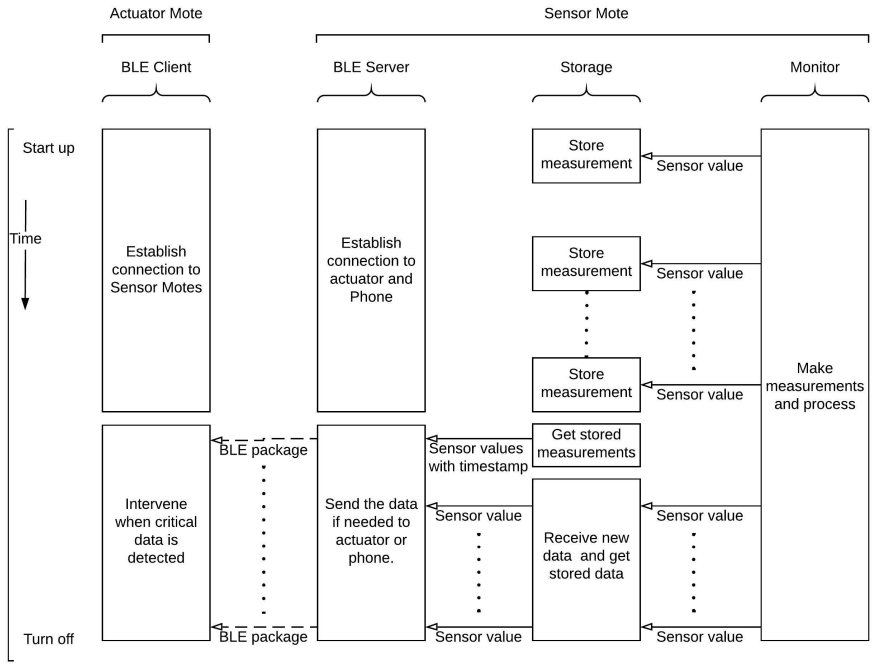


Figure 6.1: Overview of the execution of the network motes' processes over time from start up to turning off the devices.

6.2. Results

Combining

The combination of BLE Network functionality with the orientation monitor gave no problems and both worked exactly like before combining albeit for a minor delay for the BLE server in the sensor mote because it waits for the orientation monitor to be calibrated. When combining the BLE Network with the heart rate it crashes as the hardware access, calling a readout on the ADC, is too slow. This means that the BLE processes can not run while a measurement is made this results in a crash.

Power consumption

For testing the system as a whole, the motes' power consumption is measured using a multimeter in current measuring mode and the results are displayed in Table 6.1. It shows the largest current measured, this is always during starting up. It also shows the current which is achieved after a few seconds which stays relatively constant for the rest of the operating time. An estimation is given on the energy capacity of the batteries that are needed to run the device for 5 days. The energy capacities are comparable to that of a big smartphone battery for the motes which have BLE implemented and a lot smaller when this is turned off.

Mote type	**Current max**	**Current continuous**	**Energy battery**
Actuator	±300mA	±30mA	18 Wh
Orientation	±300mA	±30mA	18 Wh
Heart rate without BLE	±80mA	±10mA	6 Wh
Heart rate with BLE and mock data	±300mA	±30mA	18 Wh

Table 6.1: Table with results on current measurement and battery energy estimation for 5 days.

7 Discussion

This chapter discusses the results from the previous Chapters 4, 5 and 6.

7.1. Evaluation

The network is implemented using BLE and using the chosen hardware the motes can be made wearable as they are small. By expanding the written code, multiple other sensors can be added to the network but this is not shown in the results due to time constraints. The code can be found in the repository used, a link to this repository is stated in Appendix B.1. The BLE network implements one of the possible topologies and is proven to be able to connect, disconnect and transmit data as needed to have a function MBAN prototype. Transmission power can be reduced but this could not be combined with different topologies to research which topology would be most efficient in transmitting data for the MBAN. In Table 7.1 the requirements of the project are elucidated, these were introduced in Chapter 2.

Number	Name	Completed	Achieved
[F1]	Comfort	No	The prototype can not be worn yet.
[F2]	Scalability	Fully	The network is expendable with extra motes.
[F3]	Memory	Fully	The network can store data in case the phone is disconnected for 682 measurements.
[F4]	Dependability	Fully	The network is able to intervene, if critical data is present, independently of the phone.
[F5]	Data transmission	Fully	The network is able to send sensor measurements to the phone.
[F6]	Seizure detection	Fully	The actuator can be triggered by another mote.
[S1]	Battery lifetime	Partially	If a large enough battery is chosen the sensor will last for 120 hours this battery would be inconveniently large.
[S2]	Life expectancy	Partially	The expectation is that the product has a life expectancy of 5 years. However, due to limited time no validation could be done.
[S3]	Heart rate	Fully	The heart rate sensor is able to measure within an accuracy of 5 BPM.
[S4]	Orientation	Fully	The orientation sensor is able to give the angle with the ground with an accuracy of 2 degrees.

Table 7.1: Evaluation of requirements

7.1.1. BLE Network

Concerning the network topology some improvements can be made. The final network topology is D, see Section 4.2.1, as it was easiest to implement. The code written is not flexible enough to easily change this topology.

7.1.2. Sensors

The heart rate monitor is able to measure the heart rate using an FFT algorithm which gives readings within 10 BPM and on average 5 BPM of the real value which is simulated in MATLAB. The orientation sensor is able to calibrate itself and make measurements that are within 2 degrees of the real value and can therefore be used to track the position of the user when stationary.

7.1.3. MBAN System

When combining the measuring and BLE functionality the orientation sensor works as needed but the heart rate sensor does not. This is due to the way the measurements is done and causes a crash of the ESP32. The power consumption is larger than expected and if the sensors were to be used as is they would require a battery with the size of that of a smartphone. It would therefore be portable but not convenient.

8 Conclusions

By researching the way to implement an MBAN to prevent seizures an almost completely working prototype is achieved except for integrating the heart rate measuring in the BLE Network.

8.1. Predictions

The code is not written as efficient as possible so improvements can be made on that and also the sleep mode and more advanced BLE functionalities are not used. It would also be possible to use a second processor or parallel multi-core processing to split the BLE and monitoring functionality which would lead to less conflicts in memory and possibly an overall lower power consumption. Unfortunately the team was not experienced enough to implement this and it took a lot of time to set up the source code. Because debugging was not possible a few small workarounds were done to achieve functionality of the BLE which reduced the efficiency of the system as well.

8.2. Advice and future research

For future development of this system using BLE it would be more convenient to use a different development kit which allows for debugging. Instead of the Arduino Libraries the program could be written in C allowing for preciser tuning of the resources of the SoC. Different and existing algorithms that are proven to give exact measurements should be used which could also reduce power consumption of the system as a whole. By making the actuator efficiently switch its role within the network from acting as a server (when the phone is connected) to a client, different topologies and switching between topologies can be implemented to achieve the greatest efficiency in transmission power. Something to keep in mind when building topology A is, that a mote can function as a server and client at the same time. However one should avoid that both use the antenna at the same time, as this would make the system crash.After implementing ultra low power motes batteries can be integrated in the system making it truly possible and measurements can be done in a more realistic environment to gather data and feedback on the system's flaws and inefficiencies computation-wise. In order to lower the power usage of the independent motes it is strongly recommended to write the code not with Arduino libraries. This because these libraries are not optimised and use 2 cores to run on the chips. Therefore it is not possible to, for example use one core for maintaining the BLE connection and one core for making measurements and doing necessary computations.
Also some further development should be done for the heart rate sensor. The heart rate sensor is now not able to give correct measurements when a patient moves. How to compensate for motion artefacts is described in the MPU6050 datasheet and requires additional hardware.

Bibliography

[1] Analog Devices. Ad8232 Data Sheet - Single-Lead, Heart Rate Monitor Front End., 2012. URL https://cdn.sparkfun.com/datasheets/Sensors/Biometric/AD8232.pdf.

[2] Huasong Cao, Victor Leung, Cupid Chow, and Henry Chan. Enabling Technologies for Wireless Body Area Networks: A Survey and Outlook. (December):84–93, 2009. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.338.1526{&}rep=rep1{&}type=pdf.

[3] Jocelyne Elias. Optimal design of energy-efficient and cost-effective Wireless Body Area Networks. pages 1–30, 2013.

[4] Espressif. ESP32-DevKitC V4 Getting Started Guide, 2019. URL https://docs.espressif.com/projects/esp-idf/en/latest/hw-reference/get-started-wrover-kit.html{#}overview.

[5] Espressif. ESP-WROVER-KIT V4.1 Getting Started Guide, 2019. URL https://docs.espressif.com/projects/esp-idf/en/latest/hw-reference/get-started-wrover-kit.html{#}overview.

[6] Espressif Systems. ESP32-SOLO-1, 2019. URL https://www.espressif.com/sites/default/files/documentation/esp32-solo-1_datasheet_en.pdf.

[7] Espressif Systems. ESP32-WROOM-32D and ESP32-WROOM-32U modules specifications, 2019. URL www.espressif.com/en/subscribe.

[8] Espressif Systems. ESP32-WROOM-32, 2019. URL https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.

[9] Emmanouil Georgakakis, Stefanos A Nikolidakis, Dimitrios D Vergados, and Christos Douligeris. An analysis of bluetooth, zigbee and bluetooth low energy and their use in wbans. In *International Conference on Wireless Mobile Communication and Healthcare*, pages 168–175. Springer, 2010. URL https://link.springer.com/chapter/10.1007/978-3-642-20865-2_22.

[10] Marvin M. Goldenberg. Overview of drugs used for epilepsy and seizures: Etiology, diagnosis, and treatment. *P and T*, 35(7):392–415, 2010. ISSN 10521372. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2912003/pdf/ptj35_7p392.pdf.

[11] Emil Jovanov, Aleksandar Milenkovic, Chris Otto, and Piet C De Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and rehabilitation*, 2(1):6, 2005.

[12] Casey Kuhns. Datasheet Sparkfun Schematic Hearth Rate, 2014. URL https://cdn.sparkfun.com/datasheets/Sensors/Biometric/AD8232_Heart_Rate_Monitor_v10.pdf.

[13] Quang Duy La, Duong Nguyen-Nam, Mao V Ngo, Hieu T Hoang, and Tony Q S Quek. Dense deployment of BLE-based body area networks: A coexistence study. *IEEE Transactions on Green Communications and Networking*, 2(4):972–981, 2018.

[14] Laird. Data Sheet BL652-SA and BL652-SC Version 2.9, 2019. URL https://connectivity-staging.s3.us-east-2.amazonaws.com/2019-05/CS-DS-BL652v2_9.pdf.

[15] Shadi Mahdiani, Vala Jeyhani, Mikko Peltokangas, and Antti Vehkaoja. Is 50 Hz high enough ECG sampling frequency for accurate HRV analysis? *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2015-Novem(August):5948–5951, 2015. ISSN 1557170X. doi: 10.1109/EMBC.2015.7319746. URL https://tutcris.tut.fi/portal/files/3969576/Is_50_Hz_enough_sampling_frequency.pdf.

[16] Soni Manish and Padma Kunthe. A General Comparison Of Fft Algorithms Manish Soni, Padma Kunthe. 2011. URL https://pdfs.semanticscholar.org/0ae7/232f983a6356ebf10ccd44193b659dad6824.pdf?_ga=2.126010697.550767387.1576353786-1589233527.1571064604.

[17] J. McNames, C. Crespo, M. Aboy, J. Bassale, L. Jenkins, and B. Goldstein. Harmonic spectrogram for the analysis of semi-periodic physiologic signals. *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, 1(February):143–144, 2002. ISSN 05891019. doi: 10.1109/iembs.2002.1134427. URL https://www.researchgate.net/publication/3978230_Harmonic_Spectrogram_for_the_Analysis_of_Semi-periodic_Physiologic_Signals.

[18] E Monton, José F Hernandez, José Manuel Blasco, Thierry Hervé, Joseph Micallef, Ivan Grech, Andrea Brincat, and V Traver. Body area network for wireless patient monitoring. *IET communications*, 2(2): 215–222, 2008. URL https://www.academia.edu/16866194/Body_area_network_for_wireless_patient_monitoring.

[19] SparkFun. SparkFun AD8232 Breakout board - Hookup Guide. URL https://learn.sparkfun.com/tutorials/ad8232-heart-rate-monitor-hookup-guide?_ga=2.219858393.1181154071.1576182330-42680338.1571927261.

[20] Carl E. Stafstrom and Lionel Carmant. Seizures and epilepsy: An overview. *Epilepsy: The Intersection of Neurosciences, Biology, Mathematics, Engineering, and Physics*, pages 65–77, 2016. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4448698/pdf/cshperspectmed-BEP-a022426.pdf.

[21] Texas Instruments. ADS1293 Low-Power, 3-Channel, 24-Bit Analog Front-End for Biopotential Measurements, 2013. URL http://www.ti.com/lit/ds/symlink/ads1293.pdf.

A

Terminal prints

A.1. Actuator mote

```
Starting Arduino BLE Client application...
Try to connect to Heartrate sensor
Forming a connection to c4:4f:33:3e:c7:6b - Created client
 - Connected to server
We are now connected to the BLE Server :
Heartrate Sensor
Try to connect to Orientation sensor
Forming a connection to c4:4f:33:3e:c2:e3 - Created client
 - Connected to server
We are now connected to the BLE Server :
Orientation Sensor
Value ECG = 195
Value OR = 55
Value ECG = 209
Value OR = 49
Value ECG = 222
Value OR = 44
Value ECG = 241
Value OR = 39
Value ECG = 250
Value ECG = 250
Value OR = 31
Value ECG = 250
Value OR = 20
*** SEIZURE ***
Value OR = 16
Value ECG = 250
Value OR = 12
Value ECG = 250
Value OR = 10
Value ECG = 245
Value ECG = 241
Value OR = 9
Value ECG = 241
Value OR = 5
Value OR = 4
Value ECG = 233
Value OR = 4
```

```
Value ECG = 232
Value OR = 5
Value ECG = 229
Value OR = 10
Value ECG = 198
Value OR = 9
Value ECG = 185
Value OR = 12
Value ECG = 184
Value OR = 30
Value ECG = 182
Value OR = 39
Value ECG = 175
Value OR = 51
Value ECG = 156
Value OR = 65
Value ECG = 150
Value ECG = 150
Value OR = 74
Value OR = 79
Value ECG = 105
*** SEIZURE IS OVER ***
Value OR = 87
```

A.2. Heart rate mote

```
Starting BLE Heartrate Sensor
Transmission power changed
Waiting a client to make a connection...

Data to be saved : 54
 In register : 0
Data to be saved : 53
 In register : 1
Data to be saved : 55
 In register : 2
Data to be saved : 52
 In register : 3
Transmission power changed for connection 1.
Now there is 1 device connected
start advertising
In get devname : Actuator
Actuator connected
Data to be saved : 51
 In register : 4
Data to be saved : 55
 In register : 5
Transmission power changed for connection 2.
A device is connected
Now there are 2 devices connected
In get devname : Heartrate Sensor
In get devname : Phone
send_data = true
The local date and time is set on: Fri Dec 13 12:27:35 2019
txString to be send : 13122019122730-54
  Sending data from register 0
```

```
txString to be send : 13122019122731-53
  Sending data from register 1
txString to be send : 13122019122732-55
  Sending data from register 2
txString to be send : 13122019122733-52
  Sending data from register 3
txString to be send : 13122019122734-51
  Sending data from register 4
txString to be send : 13122019122735-55
  Sending data from register 5
Took 74 ms.
This is the txString : 50

*** Sent Value: 50 ***
This is the txString : 50

*** Sent Value: 50 ***
This is the txString : 50

*** Sent Value: 50 ***
Getting Connect Flag update
Connection Flag is TRUE
Notifying Actuator
This is the txString : 50

*** Sent Value: 50 ***
This is the txString : 50

*** Sent Value: 50 ***
This is the txString : 50

*** Sent Value: 50 ***
```

A.3. Orientation mote

```
Starting BLE Orientation Sensor
Transmission power changed
Waiting a client to make a connection...

Data to be saved : 90
 In register : 0
Data to be saved : 90
 In register : 1
Data to be saved : 90
 In register : 2
Transmission power changed for connection 1.
Data to be saved : 90
 In register : 3
Now there is 1 device connected
start advertising
In get devname : Actuator
Actuator connected
Data to be saved : 90
 In register : 4
Transmission power changed for connection 2.
A device is connected
```

```
Now there are 2 devices connected
In get devname : Phone
send_data = true
Current time is : 13/12/2019 13:35:56
Year : 2019
 Month : 12
 Day : 13
 Hour : 13
 Min : 35
 Sec : 56
The local date and time is set on: Fri Dec 13 13:36:08 2019

txString to be send : 1312201913363-90
  Sending data from register 0
txString to be send : 1312201913364-90
  Sending data from register 1
txString to be send : 1312201913365-90
  Sending data from register 2
txString to be send : 1312201913366-90
  Sending data from register 3
txString to be send : 1312201913367-90
  Sending data from register 4
Took 58 ms.
This is the txString : 90

*** Sent Value: 90 ***
This is the txString : 90

*** Sent Value: 90 ***
```

B

Code

B.1. Repository

The link stated below is a link to the repository used in this project. https://github.com/pimjansen98/MBAN

B.2. MATLAB Code

```
% Function takes file_path as argument of a measurement done with the ESP32
% in values ranging from 0 to 4095.
% The 60 second measurement is cut of at 55 seconds (skipping first 5)
% First it calculates FFT for every calculation window.
% From that it find the HR
% It then compares it to an R-peak detection algorithm and finds the average
 ↪  deviation.
function calc_HR(file_path)
close all;
clear all;
%prints all plots for every caluclation
print_all = false;

%default path
if nargin<1
file_path = '12dec_joris_varying.txt';
end
% Read textfile into array with value on new line
y = (dlmread(file_path))';
y = y(length(y)-length(y)/60*55:end)/4095*3.3; % Skip first 5 seconds of measurement
 ↪  and change to voltage

T_m = 55;  % Time length of Measured dataset in seconds
F_s = 50;  % Sampling frequency of dataset in Hz
T_ci = 1.28;  % Time length of Calculation Interval between two calculations in
 ↪  seconds
T_cs = 20.48;  % Time length of a Calculation Set in seconds
N_s_ci = T_ci * F_s;  % Number of Samples in a Calculation Interval between two
 ↪  calculations
N_s_cs = T_cs * F_s;  % Number of Samples in a Calculation Set

% Minimal and maximum to be detected heart rate and the corresp. freq.
HRmin = 50;
HRmax = 250;
fmin = HRmin/60;
```

```
fmax = HRmax/60;

% Loop over all calculation windows
i = 0;
while (i * T_ci <= T_m)
if print_all
    figure(i+1)
end
% Get calculation window for block i
begin = i*N_s_ci;
if begin - N_s_cs < 1
    x = [zeros(1, N_s_cs - begin) y(1 + begin:(begin + N_s_cs))];
else
    x = y(1 + begin - N_s_cs:(begin));
end
i = i + 1;

% Get time and frequency series
L = length(x);
T = 1/F_s;
t = (0:L-1)*T;
f = F_s*(0:(L/2))/L;

% Print calculation window
if print_all
    subplot(2,1,1);
    plot(t, x);
    xlabel("time [s]")
end

% Get FFT
Y = fft(x);
P2 = abs(Y/(2*L));
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);


imin = ceil(fmin*L/F_s);
imax = floor(fmax*L/F_s);

% Plot FFT
if print_all
    subplot(2,1,2);
    plot(f*60, P1);
    xlim([fmin fmax]*60);
end


% Find hr by taking maximum fft component
HR(i) = (f(find(P1(imin:imax)==max(P1(imin:imax)))+imin-1))*60;
end

%Algorithm finding peaks for the specific file '12dec_joris_varying.txt'
figure(length(HR)+1)
L = length(y);
T = 1/F_s;
```

```
t = (0:L-1)*T;
pks = (y>1.5).*y;
i_min = round(60/HRmax*F_s);
i = 1;
while i <= length(pks)
   if  pks(i) > 0 && any(pks(i+1:min([length(pks),i+i_min]))>0)
       i_max = i + find(pks(i:min([length(pks),i+i_min])) ==
       ↪  max(pks(i:min([length(pks),i+i_min]))), 1) - 1;
       val = pks(i_max);
       pks(i:min([length(pks),i+i_min])) = 0;
       pks(i_max) = val;
       i = i + i_min;
   end
   i = i + 1;
end
% Peaks found and transform to heart rates
I = 1:length(pks);
indices = I(pks > 0);
d_i = diff(indices);
d_t = d_i/F_s;
f_hr = 60./d_t;
subplot(2,1,2)
plot(indices(1:length(f_hr))*T, f_hr, '-+')
xlabel('Time [s]')
ylabel('Heart rate [BPM]');

% Plot ECG signal
subplot(2,1,1)
plot(t, pks, 'o')
hold on;
plot(t, y)
xlabel('Time [s]')
ylabel('Measured ECG potential [V]');
legend('R-peaks', 'ECG waveform')
ylim([1 3]);
xlim([0 55])
subplot(2,1,2)
hold on;
t = (0:length(HR)-1)*T_ci;
plot(t, HR, '-+')
xlabel('Time [s]')
xlim([0 55])
legend('Peak interval detection', 'FFT algorithm')

% Find average deviation from 20 seconds to end.
HR_interested_peak = f_hr(round(length(f_hr)/55*20):end);
HR_interested_fft = HR(round(length(HR)/55*20):end);
size(HR_interested_peak)
size(HR_interested_fft)
t_peak = (1:length(HR_interested_peak))/length(HR_interested_peak)*55;
t_fft = (1:length(HR_interested_fft))/length(HR_interested_fft)*55;
fft_peak = interp1(t_fft, HR_interested_fft, t_peak);
figure
plot(t_peak, fft_peak);
hold on;
plot(t_peak, HR_interested_peak)
```

```
hold on;
difference = abs(HR_interested_peak - fft_peak);
difference = difference(2:end);
d_avg = mean(difference);
plot(t_peak, difference);
end
```