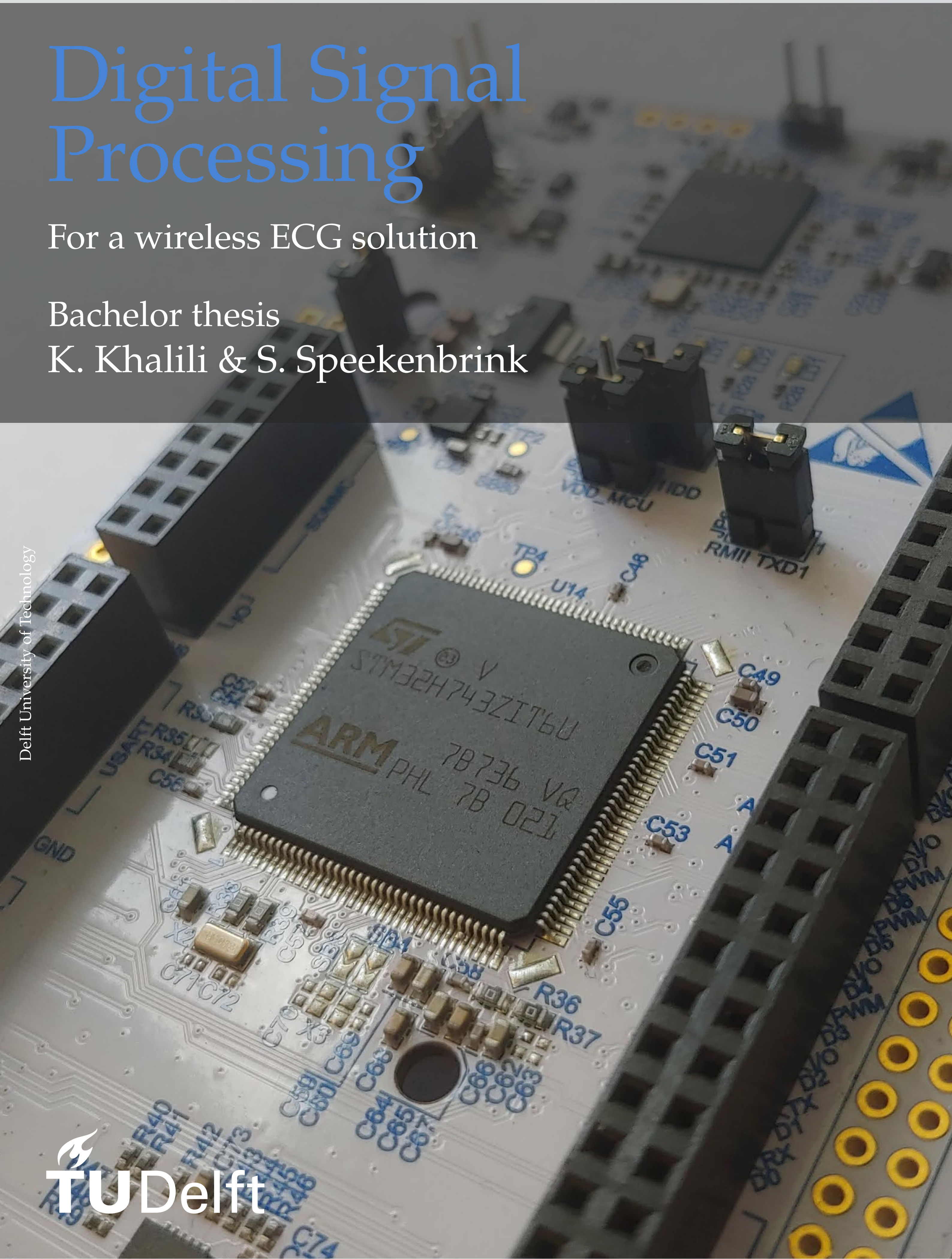


Digital Signal Processing

For a wireless ECG solution

Bachelor thesis

K. Khalili & S. Speekenbrink

Delft University of Technology

TUDelft

Digital Signal Processing

For a wireless ECG solution

by

K. Khalili & S. Speekenbrink

Student Name	Student Number
Keyvan Khalili	5118190
Sebastian Speekenbrink	5038324

Supervisor:	Prof. Dr. Ir. W.A Serdijn
Project Duration:	Apr, 2022 - Jun, 2022
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), Delft

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TUDelft

Abstract

The goal of the WiECG project is to create a prototype device that makes it possible to perform a 12-lead ECG measurement on patients without wires from the patient to a monitor. The solution consists of a transmitter and receiver, one of which is close or on the patients body and the other is connected to a monitor.

This thesis describes the design and implementation of a subsystem of the prototype device that performs digitization, digital processing and reconstruction of the measured 12-lead ECG signal. This concerns converting nine 0 to 3.3V analog signals to the digital domain by using Analog-to-Digital converters, real-time filtering of nine signals with multiple digital IIR filters and reconstructing nine digital signals to the analog domain using Digital-to-Analog converters. Furthermore, component selection, design decisions and the implementation process will be detailed in this document.

The subsystem proposed in this paper is able to successfully sample, efficiently filter and reconstruct nine signals in real time. Recommendations on improving the implementation to better adhere to the lower power requirements for a longer battery life are provided as future research prospects.

Preface

This thesis is written in context of the Bachelor Graduation Project. The project was proposed by S. Speekenbrink who originally got the idea for this project by working closely together with ambulance personnel during his minor project.

This thesis was completed in cooperation with two other groups, Hardware: R. van Krieken and P. Wiersma and Protocol: Geert Custers and Stefan Loen, all together under the supervisement of Prof. Dr. Ir. W.A. Serdijn.

As all different groups worked together very closely, documenting the work in a logical flow was quite the challenge. **It is therefore recommend to read the theses in the following order: Protocol, Digital Signal Processing and Hardware.**

We would like to thank our supervisor Wouter Serdijn for providing invaluable insights and great starting points, Asli Boru and Francesc Varkevisser for the incredible feedback and of course the other members of the WiECG group for their hard work in completing the project. Furthermore we also would like to thank Martin Schumacher for providing us with all the components needed to manufacture the WiECG system.

K. Khalili & S. Speekenbrink
Delft, July 2022

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
DMA	Direct Memory Access
DSP	Digital Signal Processing Subgroup
ECG	Electrocardiogram
EMG	Electromyography
EMI	ElectroMagnetic Interference
FFT	Fast Fourier Transform
FIR	Finite Impulse response
HW	Hardware Subgroup
I2C	Inter-integrated circuit
IFFT	Inverse Fast Fourier Transform
IIR	Inifinte Impulse response
LSB	Least Significant Bit
MCU	Microcontroller unit
PCB	Printed Circuit Board
PLI	Power line interference
PROT	Protocol Subgroup
PoR	Programme of Requirements
SPI	Serial Peripheral Interface
SPS	Samples Per Second
STM32 MCU	STM32H743ZI Microcontroller unit
WiFi	Name for international standard IEEE 802.11

1 Introduction

This thesis was written in collaboration with 6 people. The project was divided in three subgroups, each delivering their own thesis. For this reason some parts in this thesis will mimic those of other groups (e.g. General Introduction, problem statement, proposed solution and state of the art). To preserve the logical flow of the design, the order of reading should be as follows: Protocols [1], Digital Signal Processing and Hardware [2].

1.1. The WiECG Project

In the Netherlands alone, 1.3 million ambulance rides are made each year. Of these 1.3 million rides 76 % are urgent [3]. After interviewing Jim van Akkeren (Operational Head Witte Kruis Ambulance Zorg Den Haag) and Mirthe Ruijgrok (Ambulance operator) it was concluded that in 90% cases an ECG is connected to the patient being transported. An ECG is used in many cases to exclude a heart related problem as the treatment of such should happen as fast as possible. Furthermore ECGs are used when any form of anaesthesia or medicine is administered, to monitor the patient's reaction.

As the deployment of an ECG requires wires, problems arise for the ambulance personnel in applying them. The Wireless ElectroCardioGram project aims to replace these wires with a wireless solution.

1.1.1. Basics of ECG

An ECG is a visualization of the muscle contractions produced by a heart. This is done by measuring the vector projection of the hearts' electrical field on the chest of a patient. The measurement is done using electrodes located on the body of the patient, which measure potentials that are causing heart contractions. These potentials produced by the heart are then registered and visualized on a monitor.

There are two main ECG variations health workers employ [4]. One with 4 electrodes and one with 6 additional ones. The first configuration is called the extremity electrodes, which can give a general electrical overview of the heart functions. In the second configuration 6 other electrodes are added: the chest electrodes. These give more detailed information of the heart on which diagnoses can be made [5]. With these signals 12 signatures in total can be obtained.

1.1.2. Problem statement

As shown before, the usage of ECGs by ambulance personnel is crucial to ensure the well-being of the patients. ECGs are currently applied by usage of electrodes attached to the chest of the patient. For regular, non emergency use, only the extremity electrodes are used. In emergency situations a full 12 lead (10 electrodes/wires) configuration is used. Currently these electrodes consist of stickers connected to wires which are connected to a heart monitor. According to the interviewed ambulance personnel, the wires are very annoying to work with in emergency situations. The wires get tangled, dirty and in the way of the ambulance personnel as it obstructs the cabin.

1.1.3. Proposed Solution

The proposed solution is a device where the same monitor can be used as before, but where the cables have been replaced by a pair of wireless devices, a transmitter device and a receiver device. The transmitter device has 10 electrodes which have to be applied to the patient like before, requiring no extra actions for the operator. The receiver side can be plugged in to the monitor, also requiring no extra actions. This plug-and-play system can be used with any ECG monitor as long as the connector for that monitor is available. To make sure that the devices transmit and receive the right signal and not that of another pair, they can be paired easily by having the transmitter and receiver briefly connect with each other. This solution solves the problems mentioned in Chapter 1.1.2 in the following way:

- The short cables tend to tangle up way less compared to the longer ones.
- Because the transmitter device is located near the patient and the receiver lies next to the ECG monitor, no cables are suspended through the ambulance, greatly improving the comfort/workflow of the operator.

To achieve this goal, a self proposed electrical engineering bachelor graduation project was submitted to Delft University of Technology. This project will be executed with 6 people and is split up into three parts:

- Protocol (PROT)
  The group that deals with the wireless transmission of the ECG data
- Digital Signal Processing (DSP)
  The group that converts, processes digitally and reconstructs the signal and forwards it to the wireless module
- Hardware (HW)
  The group that prepares the measured signals for digital conversion and facilitates the aforementioned groups in creating a prototype device.

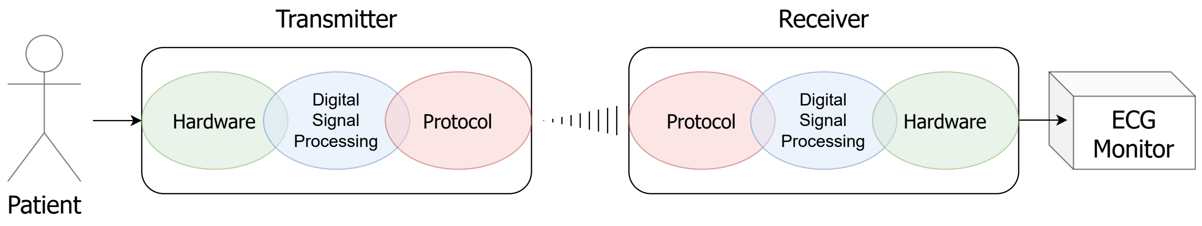


Figure 1.1: Brief overview of the solution and the separation of tasks

1.1.4. State Of The Art

Currently wireless heart monitoring (via ECG) is mainly used in several markets; medical and consumer. Consumer wireless ECG systems (e.g. KardiaMobile 6L™[6]) are usually limited to 6 leads, or only one in case of wearable devices (e.g. *Apple Watch™, Galaxy Watch™*), which outputs the data to a smartphone or display. Furthermore, the sensors and workflow used to obtain the ECG differ majorly from those that the medical personnel use. (No stickers but hand**held**/wearable sensor device).

The medical market is home to devices which are purpose-made for health monitoring on a diagnostic or treatment level. Currently, the Lifepak 15™[7] is in use in many ambulances to monitor the vitals of the patient. This device is part of a production line which is 12 years old and still in use - in Dutch ambulances - due to the others[1] not being reliable (slow to start up, easily breakable or plainly not working). The connections to the monitor of these currently used monitors are however wired, leading to the aforementioned problems.

[1]Jim van Akkeren stated that since 2017 the Philips Tempus ALS™[8] monitor was in use. But most ambulance regions stopped usage due to complaints of reliability

There are however medical wireless monitoring systems on the market that remove the wires between patient and monitor. Some examples are the ZOLL Heart Failure Management System (HFMS)™[9] and the Corpuls3™[10]. However, the HFMS focusses on detection of heartattacks in a non emergency setting and the Corpuls3 simply has a detachable wireless display (the relatively bulky defibrilator/patient box of around 3 kg has to be close to the patient). None of the devices researched focused on removing the medium of the signal like the proposed solution.

1.2. Subgroup Digital Signal Processing

The Digital Signal Processing (DSP) subgroup concerns itself with the realisation of a submodule that digitizes, digitally filters and reconstructs an analog signal, received from the Hardware group.

1.2.1. Background research

One of the most important aspects of the system is successfully capturing the important characteristics of the incoming ECG signal. Two important system features that have a big contribution to capturing the important characteristics of the incoming ECG signal are sampling resolution and sampling frequency. The research done into these two system features will be presented in this section.

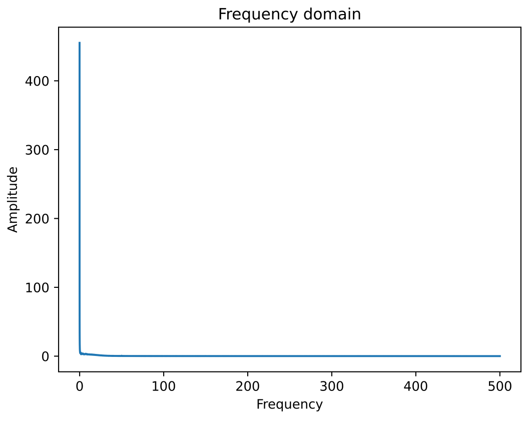
ECG Frequency contents

The maximum frequency of the input signal is an important metric, as this determines the sampling frequency. Article [11] states that QRS frequency content is composed of components from the minuscule to at least 700 Hz. Article [12] concludes that a 250 Hz sampling frequency would be acceptable for HRV analysis and 100 Hz would already be acceptable if frequency-analysis is not needed on the signal. Furthermore, Article [13] even states that virtually all of the power is contained in frequencies below 30 Hz with peak power occurring in the range of 4 to 12 Hz. On top of that, another article [14] states that a sampling frequency of 100 Hz or 250 Hz is sufficient if used by convolutional network-based deep learning models for QRS detection methods.

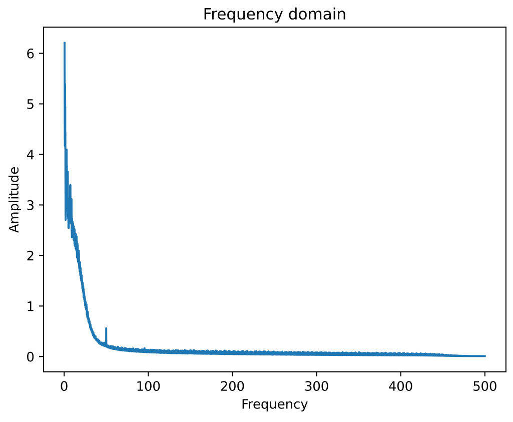
Analysis

In order to gain a better view on these papers, an analysis was performed on the PTB Diagnostic ECG Database [15] [16]. Fast Fourier Transform (FFT) was performed on all the patients in the data set and the absolute values of the frequency components were summed together to give a better understanding of what frequency components compose an ECG recording. Figure 1.2 shows the resulting value of the frequency spectrum of all the patients summed.

Very low frequency components (0-0.5 Hz) dominate the signal, as seen in Figure 1.2a. Therefore frequencies under 0.5 Hz were left out in Figure 1.2b to give a better visualisation of the amplitude of the different frequency components.



(a) Unfiltered frequency spectrum



(b) 0.5 Hz High-pass filtered spectrum

Figure 1.2: Frequency spectrum PTB Diagnostic Database

As seen in Figure 1.2b, mostly frequency components in the range from 0 to 50 Hz are present, with little to no energy in frequencies above 50 Hz. Furthermore, the small 50 Hz peak can be accounted for due to 50 Hz noise being present in the signal.

All in all, a sampling frequency of 500 Hz was chosen to include all frequencies up to 250 Hz, but also keep the data rate to an acceptable level.

ECG Resolution

The sampling resolution is in part based on Article [17]. This article states that a 16 bit resolution with the LSB $< 10\mu V$ is sufficient for the reliable analysis of the P-wave and will most likely suffice for the other parts of an ECG signal as well. Furthermore, an amplification [2] is performed on the original signal by the Hardware group, thus theoretically increasing the minimum voltage step per LSB and thus also the resolution. All in all, in order to stay ahead of the curve, to keep the programming simple (by choosing the bit resolution as a multiple of 8) and to not be dependent on the hardware subgroup (as this resolution is sufficient to not need an amplification circuit), a bit resolution of 16 bits was chosen as the sampling resolution, with the LSB comprising $5.0 * 10^{-5}$ V (see eq. 4.1). Note that the effective resolution could theoretically exceed the resolution proposed in article [17] because of the Hardware amplification circuit [2].

2 Program of requirements

Here the most important requirements are listed. Throughout the report, these requirements will get referenced to ensure that the design choices that were made have value for the end result of this project. The design described in this report and the reports of the other subgroups will attempt to completely fulfill the following list:

- **G1:** The device must not employ wires from patient to monitor.
- **G2:** The device must allow the user to perform a 12 lead ECG.
- **G3:** The device must be safe for the patient
- **G4:** The data handled by the device must be safeguarded
- **G5:** The device must not induce a delay of more than 5 seconds to the workflow of the user compared to a regular ECG wire.
- **G6:** The signal transferred by the device must be indistinguishable by the eye from the signal transferred by a regular ECG wire.
- **G7:** The device must have a battery life of 2 hours.
- **G8:** The device must not be bigger than 10cm x 20cm x 5cm (a smartphone-device)
- **G9:** The device must be lighter than 500 gram
- **G10:** A prototype device must be functional within 10 weeks from the start of the project.
- **G11:** The prototype must not cost more than 500 euros.

From this list, the following requirements were specified which specifically apply to the digital signal processing part of this project.

- **G2D:** The system must be able to sample nine signals
- **G3D:** Added noise must be removed to prevent misdiagnosis due to a distorted signal
- **G5D:** The signal processing must be done in real-time
- **G6D.a:** An input signal must be sampled at 500 Hz
- **G6D.b:** An input signal must be sampled at 16 bits resolution
- **G6D.c:** The digitized input signals must be reconstructed at 16 bit resolution
- **G6D.d:** The digitized input signals must be reconstructed at 500 samples per second
- **G7D:** Signal processing must be computationally efficient to be implemented in low power microcontrollers.
- **G10D:** The digital processing submodule must be developed within the given timespan

3
System Overview

The proposed solution in the introduction 1.1.3 can, in accordance to the PoR, be further specified as shown in figure 3.1. This also depicts a high-level overview of the contributions of each subgroup. In section 3.1.1 and 3.1.2, brief descriptions will be given of how the different parts of the transmitter and receiver attempt to fulfill the requirements given in chapter 2. Finally, in section 3.2, the complete system is shown where the detailed overviews are connected to the detailed overviews of other subgroups.

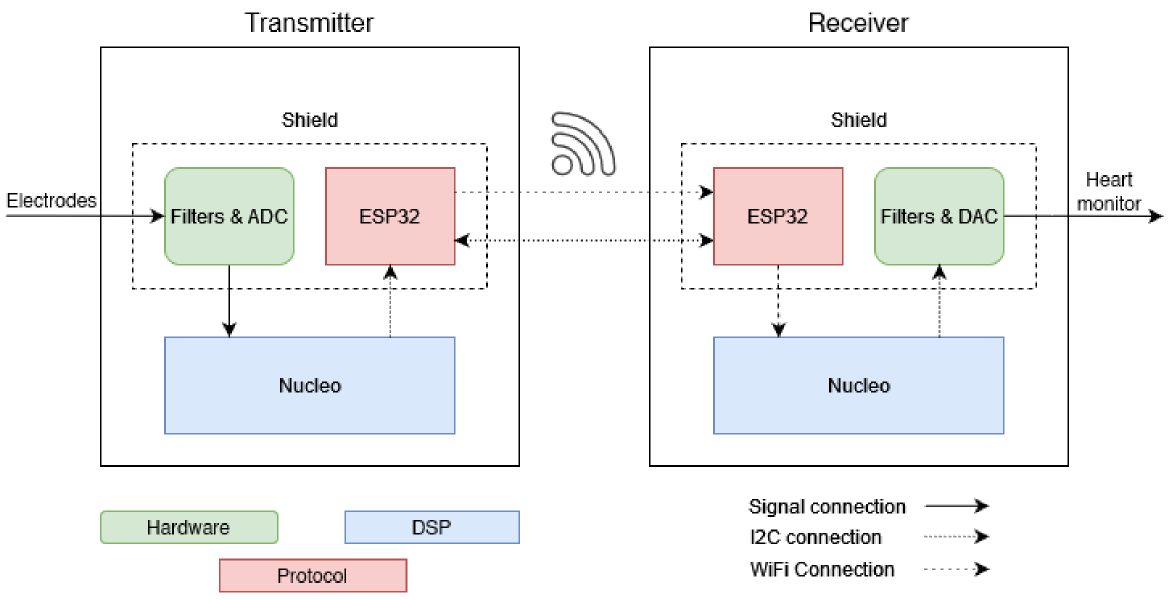


Figure 3.1: Block diagram of the general overview of the solution

3.1. Submodule Overview

In order to fulfill the requirements from chapter 2, two subsystems have been developed, a transmitter subsystem and a receiver subsystem. These subsystems have been realized on a Nucleo development board with a STM32H743ZI MCU, now referred to as STM32 MCU, together with extra peripherals. The subsystems are noted as the 2 Nucleo blocks in figure 3.1. The explanation for why this particular development board/MCU was chosen can be found in chapter 4.1.3. The detailed overview of both subsystems can be seen in figure 3.2. Both subsystems, transmitter and receiver, will be discussed in chapter 3.1.1 and chapter 3.1.2 respectively.

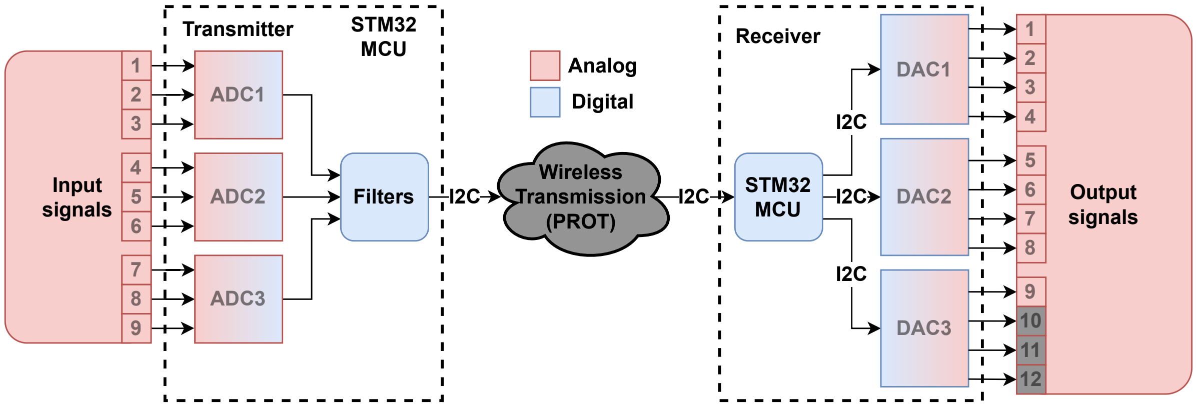


Figure 3.2: DSP System Overview

3.1.1. Transmitter

The transmitter subsystem prepares the analog signal for transmission by the Protocol subgroup. The subsystem converts the analog signal to a digital format needed for the wireless transmission and is completely realised on a STM32 MCU, placed on a Nucleo development board. The transmitter subsystem comprises 3 ADCs and a filter system. The ADCs take care of converting an analog signal to the digital domain. The design will be discussed in chapter 4.1.1 and the implementation will be discussed in chapter 5.1.
The filter system removes certain signal artifacts, with the filter design discussed in chapter 4.2 and the implementation discussed in chapter 5.2.

3.1.2. Receiver

The receiver subsystem reconstructs 9 analog signals[1] from a digital signal received from the Protocol subgroup. The receiver subsystem comprises a STM32 MCU, placed on a Nucleo development board, and 3 external DACs. The STM32 MCU takes in digital data from the protocol subgroup, modifies the data to be used by DACs and transmits the modified data to multiple DACs where the digital signal is converted to an analog signal.

3.2. Complete overview

Figure 3.3 shows the Protocol subgroup's responsibility placed in the complete overview of the system. For complete understanding of this figure, refer to the report of PROT[1] and Hardware[2]

[1]12 analog signals if the system is used as a stand-alone product.

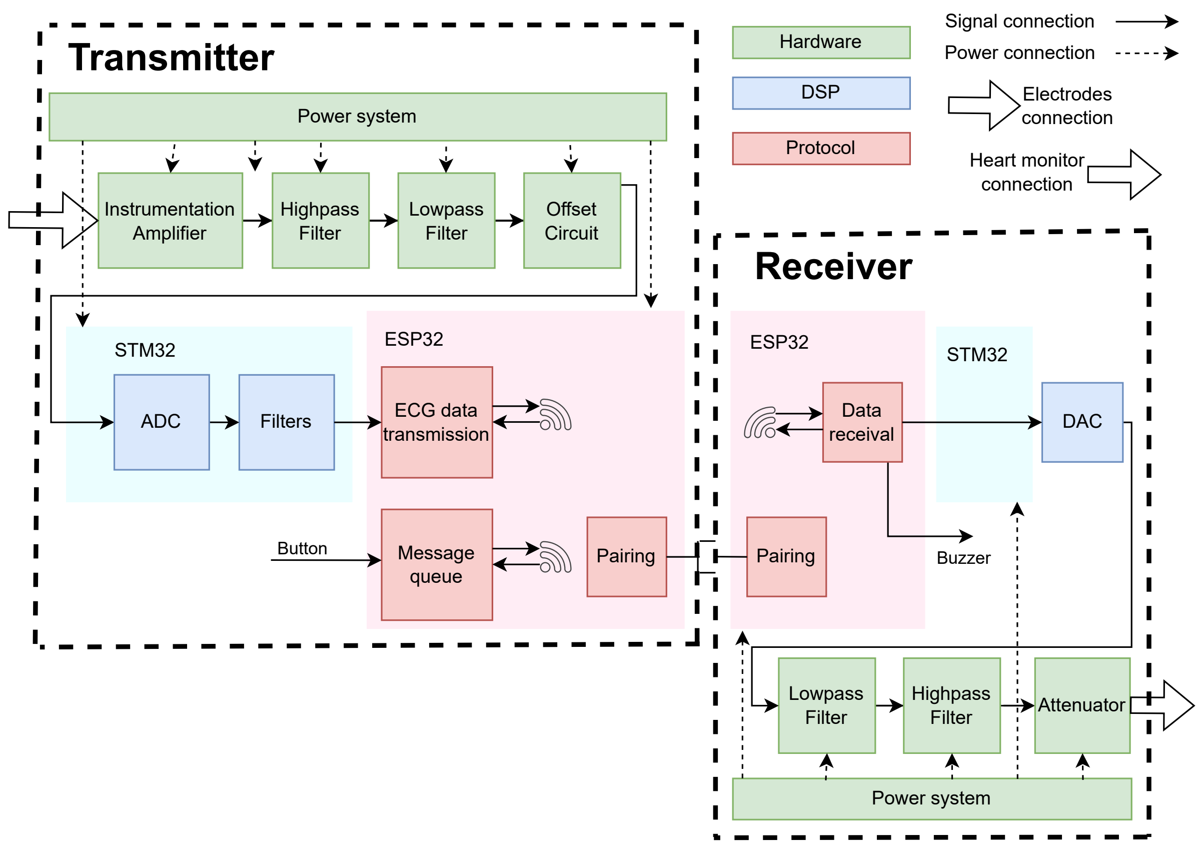


Figure 3.3: Complete overview of the WiECG project

4 Design

The objective of the design is to quantize the already amplified ECG signals, received from the Hardware subgroup, and process the signal so that it is ready for wireless transmission on the transmitter side. The second part is processing the received signal and reconstructing the signal so that it may be displayed on an external display, either an oscilloscope or an ECG monitor.

4.1. Hardware

The first aspects of the design that are considered include selection of capable hardware to assure high quality, real-time capable components. The hardware that is required to realize digital signal processing include Analog-to-Digital / Digital-to-Analog Converters (ADC/DAC) and a microcontroller (MCU).

4.1.1. ADC for digitization

Before any processing can be done, the signal has to be digitized. This is done by an Analog-to-Digital Converter. There are certain requirements that are taken into account when choosing an ADC when it comes to digitizing ECG signals. For working with ECG signals, the ADC has to be very accurate, i.e have high enough resolution (16-bit as in **G6D.b**) and have a fast sampling rate (minimum of 500 Hz as in **G6D.a**) as to not hinder real-time processing of the ECG signal or consume less of the time budget (**G10D**).

There are five different ADC types available, namely Successive Approximation (SAR), Delta-sigma ($\Delta\Sigma$), Dual Slope, Pipelined and Flash ADCs, two of which are suitable for data acquisition, Successive Approximation and Delta-sigma. These 2 technologies are able to provide an adequate resolution, sampling rate and price compared to the others and are therefore suited. The pros and cons of the possible technologies to be used is summarized in the following table [18]:

Table 4.1: Pros and Cons of ADC technologies considered

ADC Type	Pros	Cons	Resolution	Sample Rate
Successive Approximation	Good speed/resolution ration	No anti-aliasing protection	Lower	Higher
Delta-sigma	High dynamic performance, anti-aliasing protection	Hysteresis on unnatural signals	Higher	Lower

A 16-bit successive approximation embedded ADC was eventually chosen. Three of these ADCs are embedded into the STM32 MCU and were therefore chosen, as these ADCs satisfied the requirements (**G6D.a** and **G6D.b**) and are easy to interface with. These ADCs have a 16 bit resolution, quantifying from 0V to 3.3V, with the LSB comprising:

$$3.3/2^{16} \approx 5.0 * 10^{-6} \text{ V} \tag{4.1}$$

4.1.2. DAC for reconstruction

While selecting ADCs, its counter-part responsible for reconstruction of the digital data was also selected, a Digital-to-Analog Converter (DAC). Typical architectures of the DACs found included R-2R ladder and resistor string. The chosen MCU, explained in section 4.1.3, is equipped with a single internal DAC, capable of outputting two channels of data at a lower bit resolution of 12 bits. Therefore, reconstruction of the ECG data should be done using external DACs with a minimum resolution of 16 bits. The DACs that were considered for the prototype and their important characteristics are summarized in table 4.2.

Table 4.2: Multi-channel 16-bit Digital to Analog converters [19][20][21]

Property	TI DAC8554IPW	AD5675ARUZ	TI DAC8574
Supply Voltage	2.7-5.5V	2.7-5.5V	2.7-5.5V
Channels	4	8	4
Data interface	SPI/QSPI/Microwire	I^2C	I^2C
Integral non-linearity	±12LSB	±8LSB	±64LSB
Gain error	±0.15	±0.24	±1.0
Output settling time	10μs	8μs	10μs
Output slew rate	1.8$V/\mu s$	0.8$V/\mu s$	1$V/\mu s$

The chosen MCU is equipped with both Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I^2C) protocol which makes implementation of all the listed DACs possible. However, as reliability is an important aspect of the prototype, it is necessary to ensure that the data has successfully been received by the DACs. Normally, SPI is implemented with a 4-wire setup with bidirectional flow of information between the Master sender and the Slave receiver. However, in Texas Instruments DAC8554 SPI is implemented with a 3-wire setup with a one-way communication protocol, thus making confirmation of data arrival impossible. Therefore, DAC AD5675ARUZ by Analog Devices is the most suitable DAC for the project due to its low gain error, low integral non-linearity error and its I^2C communication interface.

Despite AD5675ARUZ being the most suitable DAC for the prototype, the prototype was made with TI DAC8574IPW due to lack of AD5675 availability, caused by the chip shortages. This compromise had to be made due to the requirement **G10** which dictates the time frame the prototype has to be ready in.

4.1.3. Microcontroller

The microcontroller is chosen based on its capability to perform intensive calculations needed for digital signal processing and its ability to store data (RAM/Memory). Unfortunately, due to the global chip shortage and the inability to solder a microcontroller by hand, it was decided to implement a development board into the system instead of using a standalone microcontroller placed on the PCB. Therefore, only development boards with a MCU implemented thereon will be considered as options for the microcontroller choice.

The microcontroller boards that were considered for this purpose include many different development boards. Table 4.3 shows the different development boards and their pros and cons. Simply by comparing the price against performance, all of the Arduino boards are too expensive for the performance they deliver compared to the alternatives presented. Furthermore, the Protocol subgroup tested the capabilities of an Arduino Mega for filtering and deemed this development board not powerful enough for the tasks it would need to perform. Therefore, it was decided against using any Arduino board to realize the DSP subsystem.

At last, the Nucleo-H743ZI2 was chosen as the development board in the DSP subsystem, as this development board already had adequate ADCs embedded into the chip present on the board, compared to the Teensy and the NUCLEO-F411RE. Performance-wise, the Teensy 4.1 and NUCLEO-F411RE would have been more than capable in performing the calculations needed, but were not chosen as external ADCs would have to been implemented and interfaced with, thus increasing the complexity of the system. Furthermore, the Nucleo-H743ZI2 makes use of the STM32H743ZIT6 MCU [22], a high-performance ARM Cortex-M7 chip, highly capable of the tasks it would need to perform.

Table 4.3: Development board specifications

Board:	Clock Speed: (MHz)	RAM/ROM: (KB)	Memory: (KB)	Extra:	Price: (EUR)
Arduino Uno Rev3	16	2/1	32	8 channel, 10-bit ADC	22 (Arduino) [23]
Arduino Mega 2560 Rev3	54	8/4	256	8/16 channel, 10-bit ADC	38.50 (Arduino) [24]
Arduino Due	84	96	512	12-bit ADC 12-bit DAC	38.50 (Arduino) [25]
Portenta H7 Lite	480	8196	16384	3× 16-bit ADC (36 channels, 3.6 MSPS) 2x 12-bit DAC DSP Extension	63 (Arduino) [26]
Teensy 4.1	600	1024	7936	12-bit ADC DSP Extension Not available	33.28 (Mouser) [27]
NUCLEO-F411RE	100	128	512	1× 12-bit ADC (16 Channels, 2.4 MSPS) DAC not available DSP Extension	12.84 (Mouser) [28]
NUCLEO-H743ZI2	480	1024	2048	3× 16-bit ADC(36 channels, 3.6 MSPS) 2x 12-bit DAC DSP Extension	26.68 (Mouser) [29]

4.2. Noise filtering

There are different filtering approaches available, each suited for removal of a particular type of noise. They can be divided into a few categories[30]:

1. Spatiotemporal domain method such as classical Wiener filters, median filters and adaptive filters. The disadvantage of this approach is that they are no suitable for non-stationary signals such as the cardiac signal. However they can be used for removal of frequency varying PLI[31].
2. Frequency domain methods such as band-pass filtering and wavelet-based denoising technique. While this can not remove noise overlapping with the useful frequency of the ECG signal, it can be used for removal of out of band noise.
3. Statistical methods such as independent component analysis (ICA) and principle component analysis (PCA) which can be used in removing in-band noise, but might not be feasible for real-time systems.

There are different noise sources that contaminate the ECG signal. These noise sources include [32][33][34]:

- Power line interference which consists of 50Hz and its higher harmonics.
- Electrode contact noise due to improper contact between patient and measuring system
- Motion artifact due to change in electrode-skin impedance caused by electrode motion
- Muscle contraction or EMG interference (DC-10kHz)
- Baseline wander due to respiration or movement of patient (0-0.5 Hz)

Baseline wander noise is filtered in the amplifying stage by the Hardware subgroup before being digitized and thus does not need to be filtered digitally. The remaining noise sources that have to be removed are EMG interference and PLI plus its higher harmonics. In figure 4.1, the contribution of noise sources at low frequencies, where most of the essential information is found, can be seen.

As noises originating from bad electrode contact noise and motion artifacts are preventable and might not occur under use of trained medical personnel, their removal is not attempted. The bulk of the noise produced by EMG is removed in the analog domain by a low-pass filter that also acts as an anti-aliasing filter. The lower frequency components of EMG overlap with the desired frequency band and thus its removal requires use of techniques such as PCA and ICA for in-band noise removal. However, as it can be seen in figure 4.1, it almost has a constant amplitude and thus its removal is not essential.

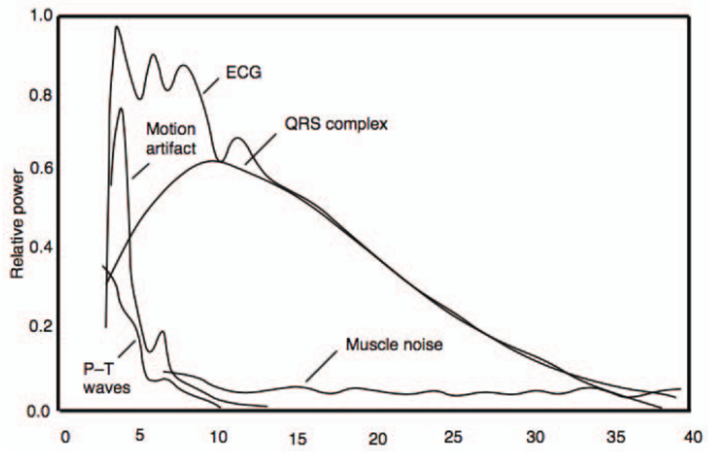


Figure 4.1: Contribution of noise sources in an ECG signal [35] in sub 40Hz range, excluding PLI noise

4.2.1. Power line interference

With the baseline wander already removed, the PLI and its harmonics are the main noise sources in the ECG signal. The PLI interference is mainly 50Hz with some deviations depending on the power usage across the grid. The grid frequency is very stable in West-Europe with average frequency deviations below ±0.2Hz[36][37]. Additionally, extreme deviations of ±2.5Hz results in an automatic disconnection of all connected generation and devices from the grid[38]. Therefore, the frequency band of 49-51Hz is the bandwidth where the PLI resides and should be removed in the system, with its higher harmonics being an integer multiple.

There are two main digital filtering categories that can be used for filtering, namely Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). IIR filters have the advantage of being less computationally expensive and requiring less memory, but have non-linear phase characteristics. This is undesired as the original shape of the input signal may not be retained[1].

FIR filters on the other hand introduce no distortion in the time-domain but introduce a constant group delay equal to half the filter order. Additionally, they require a higher order filter to achieve the same level of attenuation as IIR filters and thus require more memory and computation time. Important aspects differentiating FIR and IIR filters are summarized in table 4.4[2].

Table 4.4: Comparison of FIR and IIR filters

Aspect	FIR	IIR
Efficiency	Lower	Higher
Usage	Difficult	Convenient
Stability	More	Less
Memory usage	More	Less
Controllability	Easier	More difficult
Sensitivity	Less	More

4.2.2. IIR filters

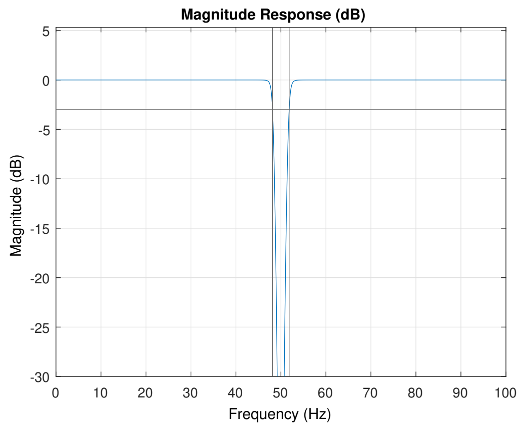
The IIR filters that can be implemented for PLI removal include band-stop and notch filters. Removal of higher harmonics can be achieved by using a comb filter or cascading multiple band-stop or notch filters. First, different band-stop filters are investigated as there are different IIR filters that may be used for the band-stop filter.

- Butterworth
- Chebyshev type I/II

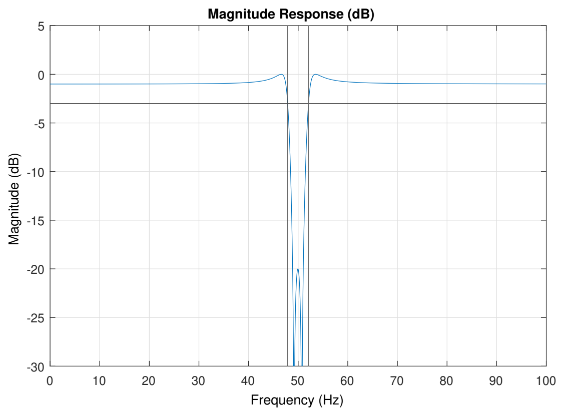
[1]source: https://www.advsolned.com/linear-phase-iir-filters-analysis-design/
[2]source:https://circuitglobe.com/difference-between-fir-filter-and-iir-filter.html

- Elliptic

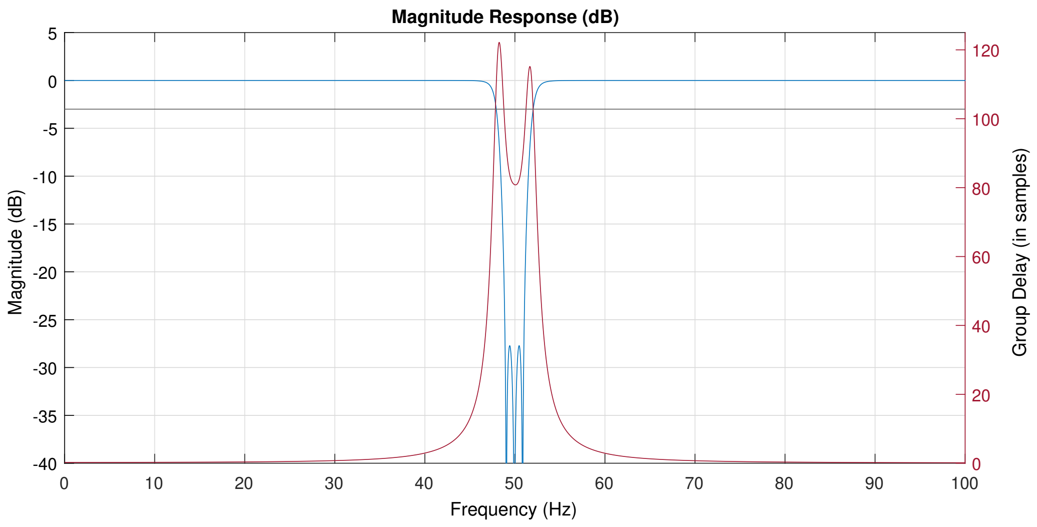
PLI consists of a very narrow band at around 50Hz and thus requires a very steep roll-off. Among these filters, Elliptic filters provide the steepest roll off but produce ripples in the pass-band. This unfavorably changes the important components of the ECG signal. Given same reasoning, the Chebyshev type I filter is also not suitable. However, both Butterworth and Chebyshev type II filters have a flat pass-band. Given that a Chebyshev type II filter has a faster roll-off than a Butterworth filter at the transitions and the ripple occuring at the stopband makes it a suitable candidate for implementing the band-stop filter. The advantage that a Chebyshev type II filter has over a Butterworth filter is that the same roll-off speed can be achieved with a lower order filer, making the filter computationally less intensive. Given that the ripple in the stop-band doesn't have any significance, the choice for the band-stop filters goes to Chebyshev type II. The magnitude and frequency response of these IIR band-stop filter designed with a stopband of 49-51Hz with attenuation of 20dB can be seen in figures 4.2 and A.1 respectively.



(a) 8th order Butterworth



(b) 4th order elliptic



(c) 6th order Chebyshev type II

Figure 4.2: Magnitude response of different IIR band-stop filters

As seen in Figure 4.2, for the same attenuation in the stop-band the Butterworth filter has a higher order than an elliptic filter or a Chebyshev type II filter. Additionally, the elliptic filter has a ripple around the corner frequency as expected and can achieve the same level of attenuation due to its high roll-off.

The main disadvantage of an IIR filter is its frequency dependent group delay. The group delay of the Chebyshev Type II filter is added to its magnitude plot in order to estimate whether the introduced group delay will be detrimental to the time domain signal. As it can be observed in figure 4.2c, the group delay introduced to frequencies 45-48Hz and 52-55Hz is substantial and is in the order of 10 – 200ms.

Given that the time interval between the P wave, QRS complex and the T wave is around 40ms [39], the filtered signal is likely to suffer from distortion. However, simulations and measurements need to be done to further investigate this assumption.

The MATLAB filterDesigner application was used to make an IIR notch filter by finding an optimized transfer function that fulfills the required specifications. A notch filter is then created with a notch at 50Hz and a stopband of 47.5 – 52.5Hz. Creating a pure notch filter, while being seen as less stable, is much simpler and can be realized with only a second order transfer function and the same roll-off as the designed band-stop filters. The disadvantage of the notch filter is that the variation PLI will not be removed as effectively as that of the band-stop filter. The more complex band-stop option might be a better choice if there are noticeable variations in the PLI frequency. However, this might remove useful components of the ECG signal as well.

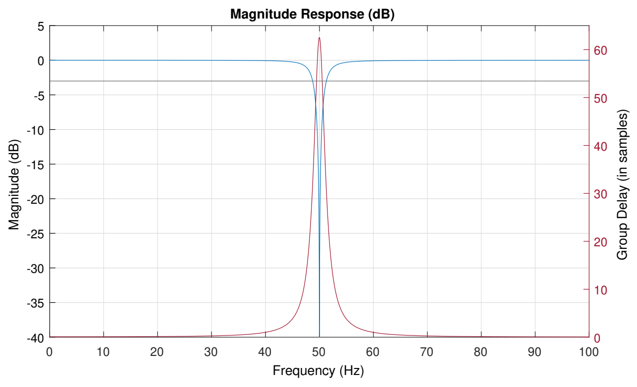
The general transfer function of the notch filter is [33]:

$$H(z) = k\frac{1 - 2cos(\omega_0)z^{-1} + z^{-2}}{1 - 2rcos(w_o)z^{-1} + r^2z^{-2}} \tag{4.2}$$

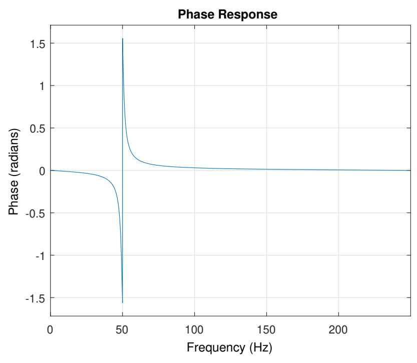
The derived transfer function with sampling frequency of 500Hz and a notch at 50Hz is as follows:

$$H(z) = \frac{0.98426 - 1.59257z^{-1} + 0.98426z^{-2}}{1 - 1.59257z^{-1} + 0.96852z^{-2}} \tag{4.3}$$

The magnitude, group delay and the phase characteristics of the designed filter can be seen in figure 4.3. As it can be seen in figure 4.3a, the notch filter has a slightly lower bandwidth but similar 50Hz attenuation as the Chebyshev Type II bandstop filter, but it has much lower group delay of 10 – 60ms. This may cause less catastrophic distortion than the bandstop filter. This, together with the notch filter's computational efficiency, makes a notch filter a more suitable IIR filter for this system.



(a) Magnitude response and group delay



(b) Phase response

Figure 4.3: Important characteristics of the designed notch filter

The last IIR filter that is considered is the comb filter. The advantage that this filter has is its ability to remove PLI as well as its higher harmonics. The comb filter acts the same as cascaded notch filters and requires a 10^{th} order filter to implement, but is theoretically much more efficient as its coefficients are mostly zero. The transfer function of the comb filter with the same bandwidth as the notch filter is:

$$H(z) = \frac{0.85812 - 0.85812z^{-10}}{1 - 0.71625z^{-10}} \tag{4.4}$$

The magnitude response of the comb filter can be seen in figure 4.4. The problem with the comb filter is that it will remove the essential components that are found in $0.5 - 2Hz$ as well. One way to alleviate this issue is to reduce the bandwidth of the comb filter as to not filter this component with the downside of worse PLI rejection. As the main goal of the filtering is to remove the PLI, the comb filter is not applicable and is therefore not implemented in the final prototype.

However, the same functionality of the comb filter without removal of these low frequencies can be achieved by means of cascading IIR notch filters with each designed filter having a notch at the frequencies of the PLI harmonics. Given the computational efficiency of the notch filter, this filter type is implemented in the final prototype.

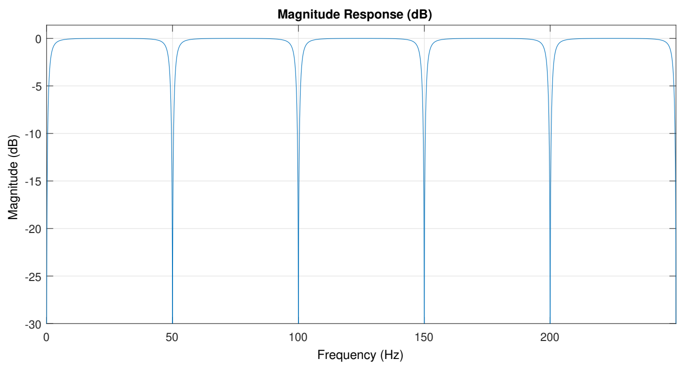


Figure 4.4: Magnitude response of the Comb filter

4.2.3. FIR filters

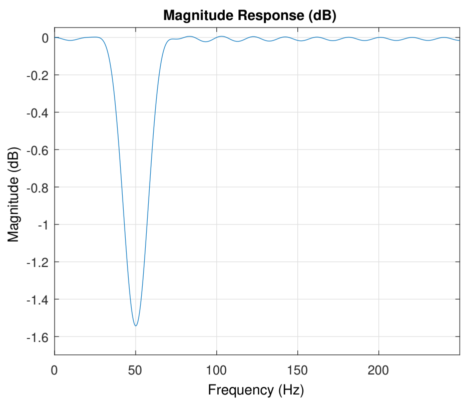
The non-linearity of IIR filters in the pass-bands may introduce distortion in the time domain. Therefore it is necessary to furthermore consider FIR filters and analyze their feasibility in real-time signal processing. Here we mainly look at Hamming and Hann FIR windows which can be used for PLI removal [34].

Using the MATLAB filterDesigner application, FIR band-stop filters are designed. In figure 4.5, the magnitude response of the Hamming and Hann window with different filter orders are shown. One of the first limitations that can be noticed is the need for a much higher order of filters to ensure the same bandwidth and roll-off as the equivalent IIR filter. A 250^{th} order is only capable of 10dB attenuation. As we do not want to remove information carrying frequencies, the filter has to have a very sharp notch, i.e. small band-stop, with high attenuation at 50Hz. Therefore, for reaching the same level of attenuation as the IIR filter as seen in figure 4.3a, an even higher order filter is required. This indicates that the amount of memory used for FIR filtering is considerably higher than IIR filters.

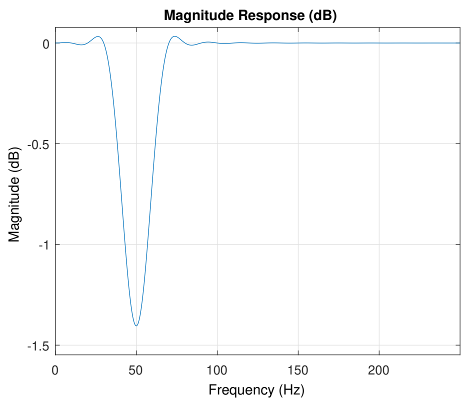
A positive aspect with respect to the FIR filter is its linear phase response. The advantage of having a linear phase response is constant group delay over the frequency range of interest. This ensures that the time domain response of the filtered signal remains distortion free.

Applying an FIR filter is not as straight forwards as an IIR filter for real time filtering. The IIR filters can be efficiently applied using convolution in the time domain, as its complexity is $O(n)$, but the same does not apply to a FIR filter as the order of filter increases. At some point it becomes too computationally expensive to do convolution in the time domain and more efficient to do the filtering in the frequency domain. The complexity of convolution and FFT is $O(m \times n)$ and $O(n \log_2 n)$ respectively.

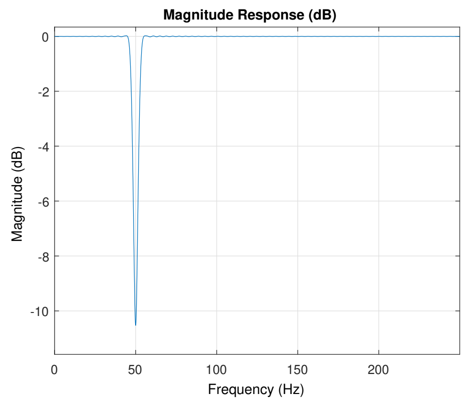
Graphically, it can be seen in figure 4.6 that the crossover point is around 40 samples, given a FIR filter of order 250. However, the execution time will differ per hardware used, as instructions per clock cycle and clock rate will differ. Therefore, this analysis has to be carried out on the chosen hardware, i.e. the microprocessor/microcontroller, as well. If an FIR filter is to be implemented on a less power demanding MCU for real-time filtering at a later stage, the MCU has to be able to perform Fast Fourier and inverse Fourier Transform. However, the need for a large memory would still persist.



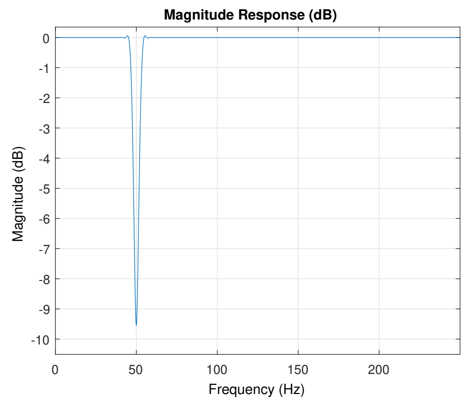
(a) 50th order



(b) 50th order



(c) 250th order



(d) 250th order

Figure 4.5: Magnitude response of Hamming window (left) and Hann window (right)

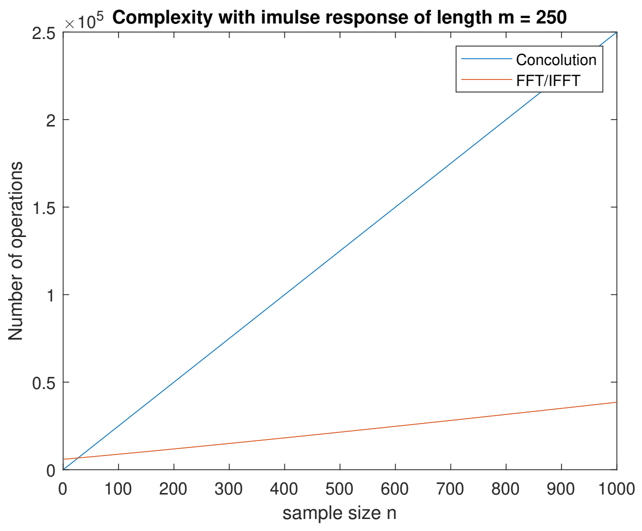


Figure 4.6: Complexity of convolution $[O(m \times n)]$ & FFT $[O(n \log_2 n)]$ based on sample size n

5 Implementation

All the tasks outlined in this report are to be implemented by two STM32 MCUs, one at the transmitter and one at the receiver. The tasks the are implemented in the prototype are as follows:

1. **Transmitter:**
   (a) Sample 9 signals
   (b) Apply digital filters to the sampled signals
   (c) Relay filtered data to an ESP32 MCU to be transmitted wirelessly
2. **Receiver:**
   (a) Receive data from an ESP32 MCU for further processing [1]
   (b) Reconstruct digital data using DACs

Development tool
In order to program/debug the STM32 MCU, the STM32CubeIDE [22] development platform will be used. STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation and debug features for STM32 microcontrollers and microprocessors. This program has everything needed to configure/program the STM32 MCU and is therefore chosen as the prefered development tool for this project.

Device configuration tool
The STM32CubeIDE comprises a GUI for configuring the STM32 MCU. Herein, an user is able to configure settings for peripherals, clocks, timers and much more. These settings are then automatically converted to C code. If it is stated that settings are configured, it means that the settings are configured in the device configuration tool and then automatically converted to C code.

Direct Memory Access
The STM32 MCU contains multiple Direct Memory Access (DMA) controllers. Direct Memory Access controllers are used to provide high-speed data transfers between peripherals and memory and between memory and memory without any CPU action. Therefore, DMA is the preferred method for any data transfer to take as much load off the CPU as possible.

5.1. Analog to Digital Conversion

The performance of the whole subsystem relies on the performance of the ADCs. If the ADCs digitize the signal poorly, the rest of the system will use the poorly digitized signals, resulting in a sub-optimal output. Therefore it is very important that the ADCs are set up well. This section outlines the method used to configure and run the ADCs.

[1]Processing in the prototype is minimal

5.1.1. Settings

In order to to use the the ADCs, they must be configured with the right settings to convert 9 signals from the analog to the digital domain. This can be done by programming the STM32 to turn on the 3 different ADCs with the following (non-default) STM32CubeIDE settings:

- Resolution: 16-bit
- Scan conversion mode: Enabled
- Conversion data management: DMA Circular Mode
- Number of conversions: 3
- External trigger conversion source: Timer 1 Trigger Out event
- External trigger conversion edge: Rising edge
- Sampling time: 810.5 cycles
- DMA: Mode circular
- ADC Clock: 150 MHz
- Other settings are left default

Taking a look at the sub module overview (Figure 3.2), every ADC needs to digitize 3 different signals out of the 9 total signals. In order to facilitate this, 3 different inputs were enabled as Single-ended. Furthermore, the number of conversions was set to 3, to notify the ADC it needs to digitize 3 different signals. This in turn enabled Scan conversion mode automatically, telling the ADC to multiplex between the 3 different input signals. Moreover, the ADCs were set up to start the conversion with the Timer 1 Trigger Out event. Timer 1 is a timer that raises an interrupt every 2 ms, thus creating a 500 Hz timer. This makes sure the ADC samples the input signals at 500 Hz (**G6D.a**). On top of that, the conversion data management was set to DMA circular mode. The DMA controller will fetch the conversion result from the ADC and put it in an array. This array will be overwritten every single time the ADCs digitize the signal. It is therefore crucial to gather the result in time before it is overwritten by a new sample. When the DMA controller is done with retrieving the conversion result, an interrupt is generated, so the data can be processed. At last, the Sampling time is set to 810.5 cycles, resulting in a sampling time of:

$$\frac{1}{150 * 10^6} * 810.5 = 5.40\mu s$$

This sampling time makes sure the capacitor of the sample and hold circuit will be adequately charged before the ADC will start digitizing the capacitor voltage. The full derivation for the minimum sampling time can be found in the Hardware report[2].

5.1.2. Code

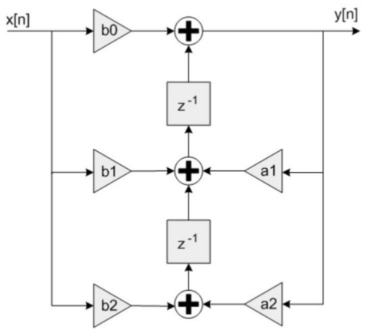
The functions of interest that are causing the ADC to work properly can be seen in the following code block:

```
    // Calibration function
    HAL_StatusTypeDef        HAL_ADCEx_Calibration_Start(
        ADC_HandleTypeDef *hadc, // Handle to the ADC
        uint32_t CalibrationMode, // Select calibration mode, usually ADC_CALIB_OFFSET
        uint32_t SingleDiff // Selection of single-ended or differential input
    );

    // Function to start the ADC in DMA mode
    HAL_StatusTypeDef        HAL_ADC_Start_DMA(
        ADC_HandleTypeDef *hadc, // Handle to the ADC
        uint32_t *pData, // Pointer to the array to store ADC results
        uint32_t Length // Length of the array used to store ADC results
    );

    // DMA Callback function
    void HAL_ADC_ConvCpltCallback(
        ADC_HandleTypeDef* hadc // Handle to ADC
    );
```

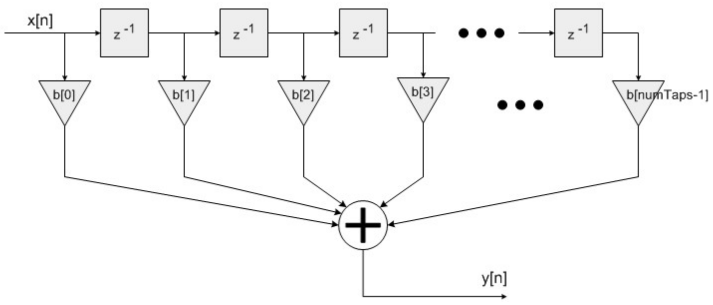
The *HAL_ADCEx_Calibration_Start* function is used to self-calibrate the ADC to get a more precise reading. When an ADC has not been used in a while or on startup, it is recommended to call this function.
The *HAL_ADC_Start_DMA* function starts an ADC in DMA mode. This is a non-blocking mode, meaning no CPU clocks are used to gather the results from the ADC, as the DMA controller, explained in Section 5, takes care of moving the data from the peripheral to memory. When the DMA controller is done transferring the data from ADC to memory, the callback function *HAL_ADC_ConvCpltCallback* is called. This callback function can then be used to process the data for example.

5.2. Digital Filters

The performance of the filters depends strongly on the type of filter chosen, precision of coefficients, method of filtering and the microprocessor used. While a general function could be written to implement the digital filters, the chosen microprocessor has an extensive DSP library that can perform the filtering operation more efficiently than general code would as they are written with the available hardware in mind and use their full capability. The IIR filters are implemented using a Biquad filter in direct form II. Higher order filters can be implemented as well by means of cascading Biquad filters. The topology of the FIR and IIR filters implemented in the CMSIS DSP library can be seen in figure 5.1.



(a) Topology of the Biquad IIR filter



(b) Topology of the FIR filter

Figure 5.1: Topology of FIR and IIR filters implemented in CMSIS DSP Library[40]

The functions of interest, which perform filtering by convolution in the time domain, found in the CMSIS DSP library and used in prototyping can be seen in the following code block:

```c
typedef struct
{
    uint8_t numStages;          /**< number of 2nd order stages in the filter. */
    float64_t *pState;          /**< points to the array of state coefficients. */
    const float64_t *pCoeffs;       /**< points to the array of coefficients. */
} arm_biquad_cascade_df2T_instance_f64;

void arm_biquad_cascade_df2T_f64 (const arm_biquad_cascade_df2T_instance_f64* S,
                                   const float64_t* pSrc,
                                   float64_t* pDst,
                                   uint32_t blockSize
                                   );

// For FIR Filtering
typedef struct
{
    uint16_t numTaps;      /**< number of filter coefficients in the filter. */
    float32_t *pState;     /**< points to the state variable array. The array is of length
 numTaps+blockSize-1. */
    const float32_t *pCoeffs;    /**< points to the coefficient array. */
} arm_fir_instance_f32;

void arm_fir_f64 ( const arm_fir_instance_f64*  S,
                const float64_t*     pSrc,
                float64_t*     pDst,
                uint32_t     blockSize
                );
```

The full description of the functions and how they work can be found in the CMSIS DSP Software Library[40]. In short they use normal convolution to apply filter "**S**" to input data "**PSrc**" of size "**blockSize**" and then store the results in "**pDst**".

Additionally, for a more efficient implementation of FIR filters in frequency domain, the CMSIS library includes Fast Fourier Transform functions which are included for completeness but are not implemented or tested due to time constraints outlined by **G10**.

```c
    typedef struct
    {
        uint16_t fftLen;                   /**< length of the FFT. */
        const float32_t *pTwiddle;         /**< points to the Twiddle factor table. */
        const uint16_t *pBitRevTable;      /**< points to the bit reversal table. */
        uint16_t bitRevLength;             /**< bit reversal table length. */
    } arm_cfft_instance_f32;
    typedef struct
    {
        arm_cfft_instance_f32 Sint;      /**< Internal CFFT structure. */
        uint16_t fftLenRFFT;             /**< length of the real sequence */
        const float32_t * pTwiddleRFFT;        /**< Twiddle factors real stage  */
    } arm_rfft_fast_instance_f32 ;

    arm_status arm_rfft_fast_init_f32 (
        arm_rfft_fast_instance_f32 * S,
        uint16_t fftLen);

    void arm_rfft_fast_f32(
        arm_rfft_fast_instance_f32 * S,
        float32_t * p, float32_t * pOut,
        uint8_t ifftFlag);
```

5.3. Digital to Analog Conversion

As seen in table 4.3, there is no MCU that possesses nine DAC channels. Therefore, regardless of the chosen MCU, the prototype requires external DACs. In figure 5.2, a simplified overview of component connections in the receiver prototype, where signals are reconstructed, can be seen[2]. The Nucleo would receive the wirelessly transmitted data from the ESP chip via an I^2C line. This mode of communication between the STM32 MCU and the ESP chip was discussed with PROT subgroup and chosen as the most suitable communication protocol.

At the receiver, the Nucleo has the option to store a buffer of the received data to perform further signal processing. However, as there is minimal data processing taking place on the Nucleo on the receiver module's prototype, no buffer is held. The immediate data is prepared to be sent via a single dedicated I^2C line to the three DACs. The complete communication with the DACs are documented in appendix B.2.

The I^2C protocol is implemented using the Nucleo's built-in "Hardware Abstraction Layer" (HAL) library. Using this library puts emphasis more on fast prototyping rather than the most efficient implementation, as the built-in functions are more complex and contain many checks to catch most of the unexpected outcomes. However, they do not cause any performance overhead in the prototype. The process of receiving data from the ESP32, preparing the data and transmission to the DACs is implemented using the following functions and simplified sequence [41][2]:

```c
    // Timer interrupt to initiate reception & transmission process
    void HAL_TIM_PeriodElapsedCallback(timer)

    // Sets I2C line 1 in reception mode
    void HAL_I2C_Slave_Receive_IT(hi2c1, pData, Size);

    // Reception is complete call back function.
    // Data is processed here as preperation for DACs
    void HAL_I2C_SlaveRxCpltCallback(hi2c1)
    {
        // Transmission to DACs is initiated
```

[2]Detailed workings of these functions can be found in [42], chapter 47

```
        void HAL_I2C_Mem_Write_IT(hi2c, DevAddress, MemAddress, MemAddSize, pData, Size);
    }

    // Transmission complete call back function
    void HAL_I2C_MemTxCpltCallback (hi2c2){
    {
        // Retranmission of next channels of data until all channels are updated
        HAL_I2C_Mem_Write_IT(hi2c2, DevAddress, MemAddress, MemAddSize, pData, Size);
    }
```

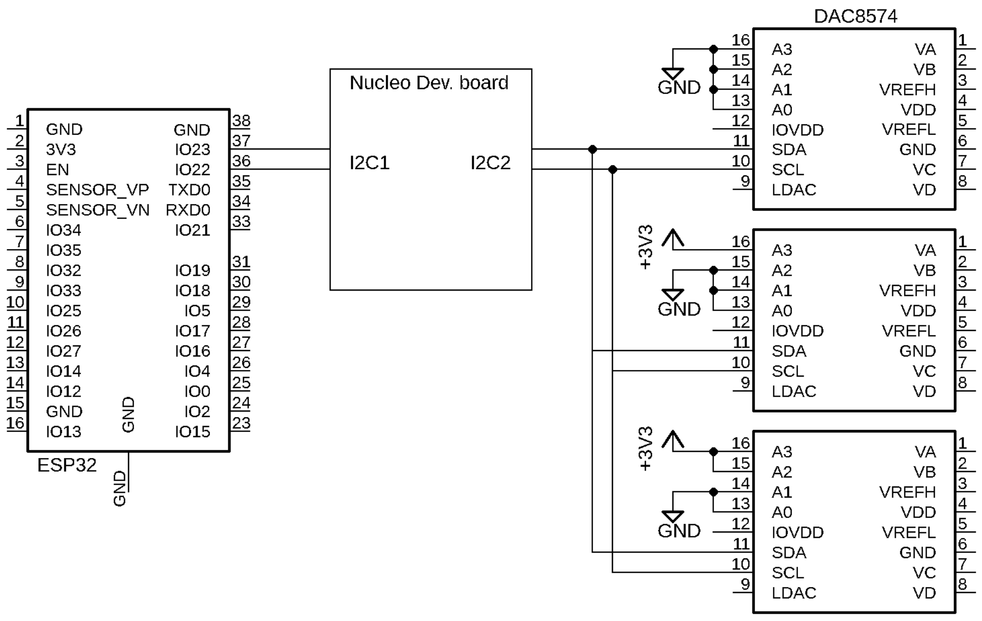


Figure 5.2: Simplified schematic of the receiver module

6 Measurements

In this chapter, the important characteristics of the designed modules are benchmarked to ensure that they follow the requirements outlined in chapter 2. These benchmarks include both theoretical simulations and practical measurements.

6.1. Execution time

6.1.1. Filtering time

The benchmarks are performed on a STM32H743 board, with the STM32H743ZI2 MCU. STM32 boards include CPU cycles counter as part of the Debug Watch and Trace (DWT) module[43]. This counter can be easily used to measure the execution time of a given code. The cycle counter can be enabled and used as indicated in the code block below[43]:

```
#define  ARM_CM_DEMCR      (*(uint32_t *)0xE000EDFC)

#define  ARM_CM_DWT_CTRL   (*(uint32_t *)0xE0001000)

#define  ARM_CM_DWT_CYCCNT (*(uint32_t *)0xE0001004)

  // Enabling Counter
  if (ARM_CM_DWT_CTRL != 0) {         // See if DWT is available

      ARM_CM_DEMCR      |= 1 << 24;  // Set bit 24

      ARM_CM_DWT_CYCCNT  = 0;

      ARM_CM_DWT_CTRL   |= 1 << 0;   // Set bit 0

  }

  start = ARM_CM_DWT_CYCCNT;

  // Code to measure

  stop  = ARM_CM_DWT_CYCCNT;

  delta = stop - start;
```

Then the execution time can be derived from the recorded counter and the SystemCoreClock. The filtering performance of Nucleo-H743ZI2 and Nucleo-F411RE on a 250 sample data is summarized in tables 6.1 and 6.2 respectively. The results show that both microcontrollers are suitable for single precision real-time filtering. However, should double precision be used, only the MCU equipped with Cortex Arm M7 is suitable.

Furthermore, it can be seen that, on both MCUs, the execution time of the FIR filter is around 50 times slower than the IIR filter while having lower attenuation and a considerably higher stop-band bandwidth which is not desirable.

Table 6.1: Performance of **STM32H743ZI2** with clock rate of 480MHz

Filter type	Filter order	Precision	execution time [μs]	Clock Cycles
IIR Notch	2nd	single	17	8 000
5× IIR Notch	10th	single	85	39 600
IIR Comb	10th	single	438	210 000
FIR Bandstop	250th	single	952	453 000
IIR Notch	2nd	double	32	15 500
5× IIR Notch	10th	double	159	77 300
IIR Comb	10th	double	454	218 000

Table 6.2: Performance of **STM32F411RE** with clock rate of 100Mhz

Filter type	Filter order	Precision	execution time [μs]	Clock Cycles
IIR Notch	2nd	single	47	4 700
5× IIR Notch	10th	single	233	23 300
IIR Comb	10th	single	476	47600
FIR Bandstop	250th	single	2100	210 000
IIR Notch	2nd	double	1960	196 000
5× IIR Notch	10th	double	9700	972 000
IIR Comb	10th	double	1600	160 500

The FIR filter performance can be improved significantly if the filtering is done in the frequency domain using FFT/IFFT as seen in figure 4.6. By using FFT/IFFT, the complexity of the calculations will be drastically reduced to $O(n \log_2 n)$.

The 250 samples used represent half a second of data, thus the FIR filter can theoretically be used for real time signal processing of nine channels on both the ARM CORTEX M4 and M7, but this would leave less performance overhead for other functions of the microprocessor, i.e. sampling, transmission and possibly re-transmissions that ensure high reliability. This leads potentially to forming a bottleneck in the system. Additionally, this might prevent the transition of the prototype to an even more efficient MCU to fulfill requirement **G7D**.

Even though using FFT/IFFT would improve performance of the FIR filter, it would not reduce the memory usage. Additionally, it will be still more computationally intensive than the IIR filter while having a larger notch and less attenuation as seen in figures 4.3a and 4.5.

The only advantage that the FIR filter has is its constant group delay which prevents distortion in the time domain. The group delay in the IIR filter occurs around the notch which is attenuated. This indicates that the distortion and hence the illegibility of the ECG signal in time domain will likely be limited. The group delay of the IIR and the FIR filter can be seen in figure A.3.

Having a very steep notch does have a disadvantage that is present in both IIR and FIR filters and that is the time it takes for filter to reach the designed attenuation level. As it was seen in figure 4.5, a very high order FIR is needed for a top notch which results in a high group delay. While the IIR filter does not have a group delay in its pass-band due to having less delay elements, it too needs prior samples in order to properly filter out the 50Hz noise.

This delay was seen in MATLAB simulations as well as when a 50Hz sinusoidal signal was passed through the IIR filter implemented on the STM32 MCU. It was observed that the attenuation would converge after some delay, seen in figure A.4, which is inversely proportional to the sampling rate. This is further investigated and documented in section 6.2

6.1.2. DAC Reconstruction time

The chosen DAC, namley Texas Intruments DAC8574IPW, is capable of operating with wide ranges of I^2C bus rates ranging from 100kb/s upto 3.4Mb/s, with the upper limit of the chosen Nucleo board being 1Mb/s. Based on the data sheet of TI DAC8574IPW, updating each output channel requires four bytes of data and an additional four is required as a broadcast signal for synchronous updating of the DAC's output channel. This results in a minimal transfer speed of ~ 205kb/s in ideal conditions. As some overhead has be left in transmission, the highest transmission speed of 400kb/s is chosen, which does not alter the "Standard/Fast Mode" transmission protocol of TI DAC8574IPW[21].

Tests performed and documented in table 6.3[1] are done on the receiver prototype, designed and made by the Hardware group [2]. If real-time reconstruction of ECG signals is to be guaranteed, the time needed for receiving data from the ESP32 and transmitting to DACs must be below 2ms, as outlined by **G6D.d**. It can be observed that the sequential reception and reconstruction of the nine signals are done successfully under 1.3ms in real-world operating conditions.

Table 6.3: Communication time from/to the Nucleo MCU

	No. of Samples	I2C bus rate $[kb/s]$	execution time $[ms]$	Effective bits per sample
3× DAC8574	12	400	1.43	47
3× DAC8574	9	400	1.06	47
ESP32	9	1000	0.20	22

6.2. Filter Performance

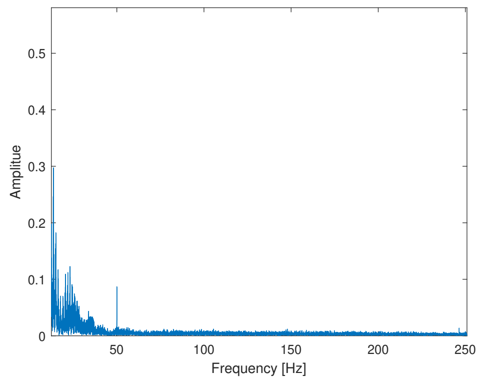
This section describes the performance of the filter. The theoretical performance of the filter will be looked at, realized in Matlab and on the STM32 chip, and the real-life performance will be tested on a real-life ECG signal.

6.2.1. Theoretical

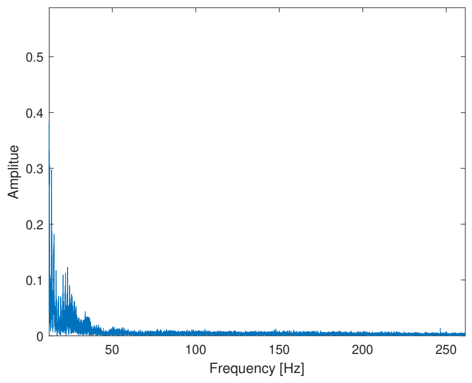
In order to test the (semi-)theoretical performance of the filter, the filter performance is tested in different environments. First, the performance of the filter is tested in Matlab. Subsequently, the performance of the filter, realized on the STM32 MCU, is tested with a pure 50 Hz sine wave. At last, the filter performance is tested on an ECG signal with 50 Hz added, output by a function generator.

Matlab

Using the MIT-BIT database [44][16], the frequency content of the ECG signal is looked at. Firstly, a great peak is seen at the lower frequencies, which are due to the baseline wander noise. However, this noise would be removed in an earlier stage with analogue circuitry and is thus not the focus of this analysis. Applying the designed notch filter to the data, shows that PLI is effectively removed. The power spectral density of the ECG signals before and after filtration can be seen in figure 6.1



(a) Without Notch filter



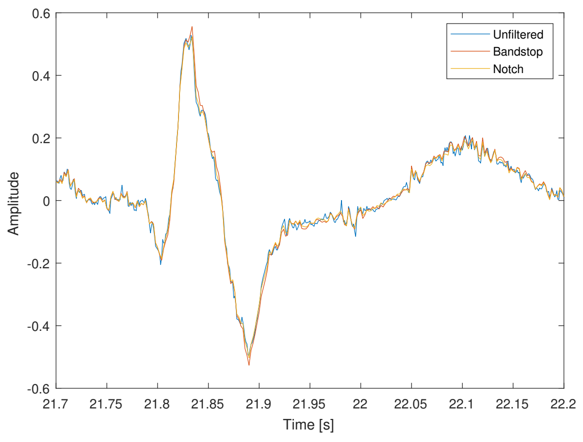
(b) With Notch filter

Figure 6.1: The power spectral density of the ECG signal from MIT-BIT data base

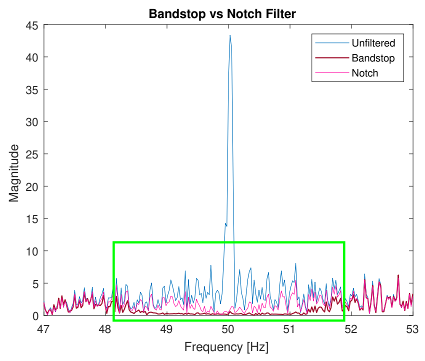
Testing the bandstop Chebyshev Type II filter on the same data and zooming at frequencies around

[1]12-channel is included in case the prototype is used as a standalone ECG monitoring device.

50Hz shows the difference in filtering capabilities of the designed filters. It can be observed in figure 6.2b that the bandstop filter is much more suited for removing a wider band as expected. Initially, a high level of distortion was expected due to a large group delay. However, testing on an ECG data signal from MIT-BIT data base did not show distortion. This indicates that despite having a large group delay, the Chebyshev bandstop filter may be used for filtering the PLI as well.



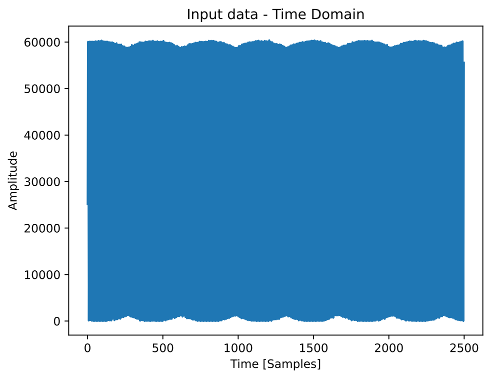
(a) ECG signal in time domain



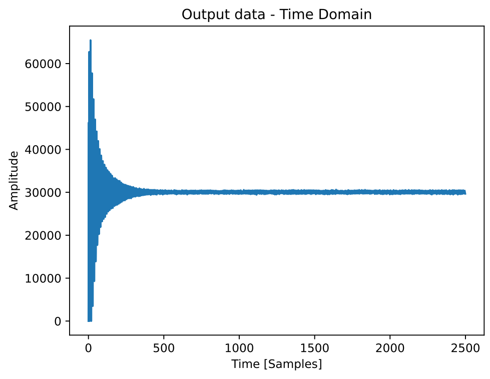
(b) ECG signal in frequency domain

Figure 6.2: Time domain and frequency domain comparison of notch & band stop filters

Function generator 50 Hz
To test the filter when realized on the STM32 MCU, a function generator was configured to output a pure 50 Hz sine wave with an amplitude of 3.000 V_{pp} and an offset of 1.500 V. This signal is then sampled for 5 seconds at 500 Hz by an implemented ADC. The resulting digital signal is then fed into the filter system. Figure 6.3 shows the filter input and filter output in the time-domain.



(a) Filter input time-domain



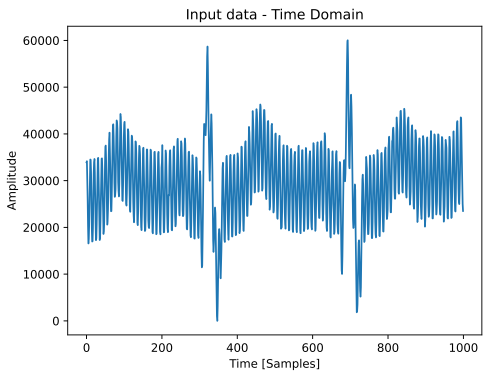
(b) Filter output time-domain

Figure 6.3: Filter input and output in the time-domain

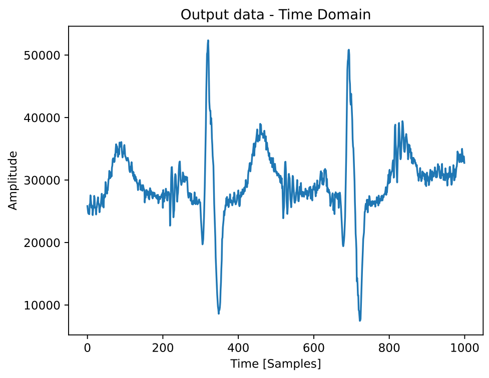
As can be seen in Figure 6.3b, the filter first needs to converge and then filters the data effectively. This is to be expected and is also explained in Chapter 6.1.1. Comparing to the input signal, shown in Figure 6.3a, the 50 Hz signal is as good as completely filtered in 1 second. A frequency spectrogram of the signal can furthermore be found in Appendix A in Figure A.5.

Frequency generator ECG Signal
With an eye on the end goal of this project, the filter system was tested on a repeating 2 second ECG signal, obtained from the first 2 seconds of patient 001 from the PTB Database [15] [16]. Additionally, a 50Hz signal was added to the original signal to simulate 50Hz power line interference, with an

amplitude of 0.2 * the peak-to-peak amplitude of the original ECG signal. Both these signals can be seen in Appendix A.1.1 as Figure A.6a and Figure A.6b respectively. At last, this signal was loaded onto a function generator (Tektronix AFG 3021B) and output to the Nucleo, with a frequency of 0.5 Hz and its voltage ranging from 0 to 3V. The method for loading an arbitrary signal is explained in Appendix C. The signal sampled by an ADC and the result after the filter system can be seen in Figure 6.4. In order to filter the 50 Hz component better, two 50Hz filters have been cascaded, resulting in bigger magnitude decrease of the 50Hz signal components.



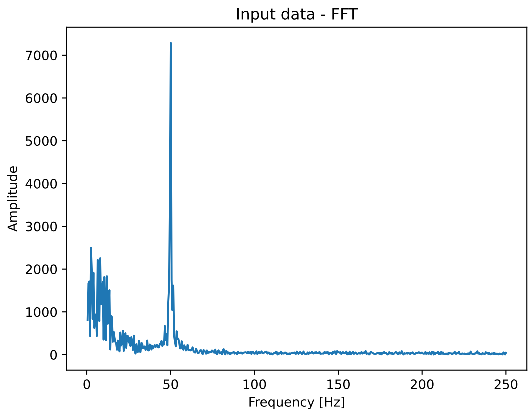
(a) Filter input time-domain ECG Signal



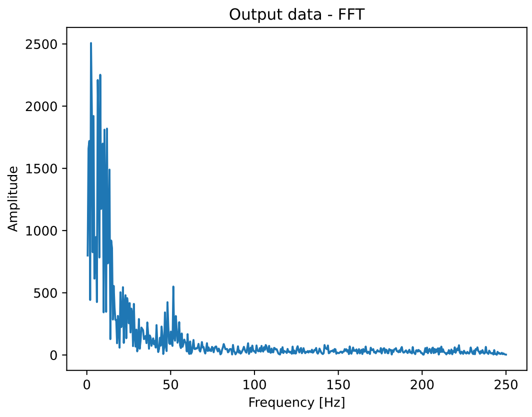
(b) Filter output time-domain ECG Signal

Figure 6.4: Filter input and output in the time-domain of ECG Signal

By looking at the result of the filter, Figure 6.4b, it can be seen that the signal visually already looks more clear. However, still some harmonics can be seen in the final output. By taking a look at the FFT of the input and output of the filter with the DC component removed, Figure 6.5a and Figure 6.5b respectively, it can be seen that the 50 Hz component has vastly decreased, but not completely removed. More filters could be cascaded to give an even better magnitude decrease, but this would introduce a bigger group delay, thus distorting the signal more; a trade-off has to be made between the two when deciding on the magnitude decrease.



(a) Filter input FFT ECG Signal - No DC

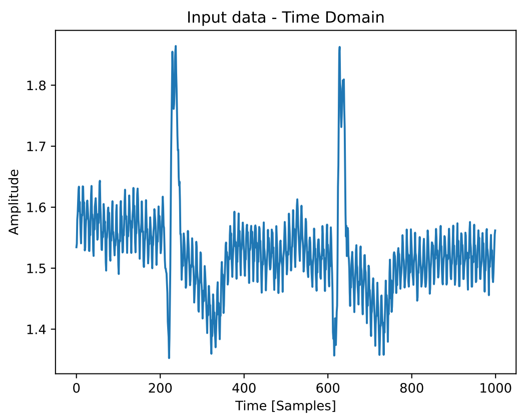


(b) Filter output FFT ECG Signal

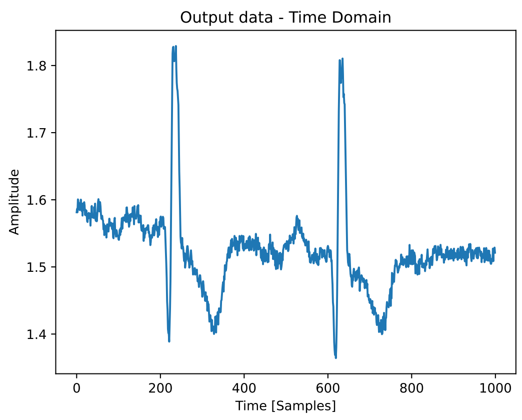
Figure 6.5: Filter input and output FFT of ECG Signal

6.2.2. Practical

Furthermore, to test the real-life application of the DSP subsystem, an experiment has been performed on a male human (S. Speekenbrink) to test the incoming signal and the filter system. The time-domain and FFT results can be seen in Figure 6.6 and Figure 6.7 respectively. Note that the FFT Figures are without a DC component.

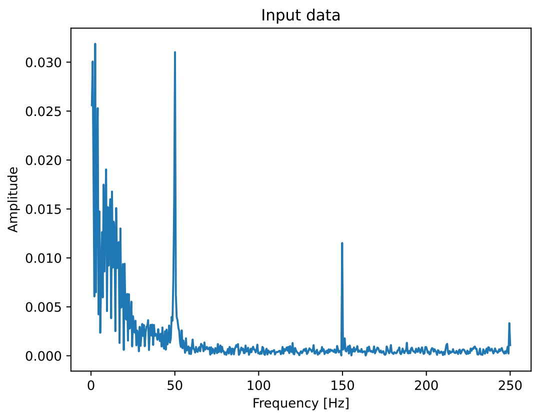


(a) Time-domain real ECG signal input

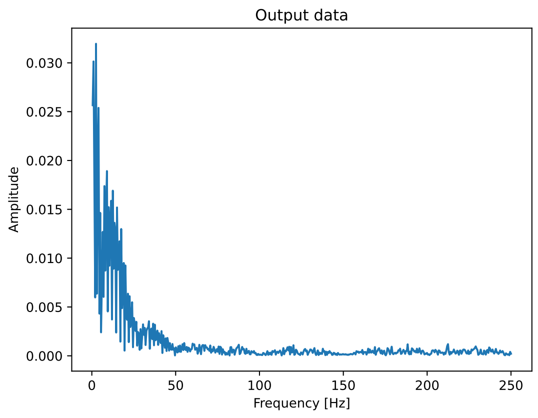


(b) Time-domain real ECG signal output

Figure 6.6: Time-domain real ECG signal input/output



(a) FFT real ECG signal Input - No DC



(b) FFT real ECG signal Output - No DC

Figure 6.7: FFT Real ECG Signal - No DC

First of all, by looking at Figure 6.7a, it can be seen that higher harmonics of PLI are present, namely 150Hz and even 250Hz. This is most likely accounted for by the fact that these tests were performed with a power supply connected to the main grid, instead of a power supply that is not connected to the main grid, i.e. batteries. It might be possible that higher harmonics will also be present in the case when a power supply is used that is not connected to the main grid. In that case, filters for these components would have to be implemented in the final system. Unfortunately, as of the moment of writing, this has not been possible yet, because the system could not be connected to a power supply not connected to the main grid at the moment of testing the real-life ECG signal. Therefore, 100Hz, 150Hz, 200Hz and 250Hz filters have been implemented together with a 50Hz filter to filter out the 50Hz PLI and higher order harmonics.
The FFT of the result can be seen in Figure 6.7b. By looking at Figure 6.7b, still a minor amount of 50Hz is present in the resulting signal. However, by looking at the resulting time-domain signal in Figure 6.6b, this 50Hz component is not dominating the signal anymore and is therefore sufficiently filtered, compared to the input signal in Figure 6.6a. The result is an ECG like signal.

7 Discussion

Here the results of this thesis are discussed and an outline is given on how the prototype could be further improved, in terms of efficiency (**G7**), size (**G8** & **G9**) and cost (**G11**). Additionally, recommendations are given for further research avenues.

7.1. Micro Controller power usage

In order to prevent bottlenecks in the development process, one of the most powerful MCUs was chosen that not only is capable of carrying out the functions of the DSP submodule, it does it extremely fast, as seen in table 6.1, at the cost of extra power usage. Measurements done by HW subgroup showed that the Nucleo board uses 200-300mA which accounts for ~ 50% of total power usage of the system. This power usage occurs when the MCU is running at its maximum clock frequency. Many low power MCUs do not have the hardware for double precision float computations. Therefore, if double precision is to be kept, the Nucleo has to be run at a lower and more reasonable clock frequency to ensure low power usage and double precision arithmetics.

However, if the single precision results tightly resemble ones of double precision filters, then a lower power MCU equipped with a Cortex M4 MCU (with Floating Point Unit) may be used to reduce power consumption drastically, but this would require addition of external peripherals, i.e. ADCs. Due to many low power boards being equipped with DMA, covered in start of chapter 5, the MCU will likely have enough computation power to realize the DSP functions.

Alternatively, all the functions implemented in the DSP and PROT subgroups may be implemented on a single MCU, namly an ESP32 MCU, as it has both WiFi and a dual core CPU design with a generous clock rate of 240MHz. However, the feasibility of this depends on the performance of the DSP library that is available for an ESP32 MCU.

With these options, total number of components used in the system decreases, improving the system efficiency while reducing its costs. Additionally, this facilitates the transition into a more miniaturized system.

7.2. Filtering efficiency

Based on both simulations performed in Matlab as well as measurements on the STM32 MCU, the designed filters consisting of cascaded notch filters are capable of removing PLI and its higher harmonics without distorting the ECG signal in the time domain. However, tests were not carried out on heart data of patients with arrhythmia to investigate the amount of distortion that is added. Given signals collected from patients with arrhythmia contain different frequency components, the amount of distortion cannot be predicted without being specifically tested.

Furthermore, other filter systems could be implemented to improve the Signal-to-Noise Ratio of the output signal. For example, a form of averaging could be applied to reduce the effect of white/coloured noise. Unfortunately, white noise filtering was not possible in the current system design as the RC time constant of the system at the terminal input was too high to gather enough samples quickly enough to use averaging efficiently, as a sample-and-hold circuit is used. Signal averaging could however be implemented with Article [45] explaining the use on ECG signals.

7.3. DAC Integral non-linearity error

The Texas Instruments DAC8574 that was selected for this prototype has a relatively high integral non-linearity error and was chosen only due to the long lead time of the DAC AD5675 by Analog Cicuits. DAC8574IPW, while not making the ECG signal distorted beyond recognition, should preferably not be used in further revisions of the prototype. The error introduced by DAC8574IPW before attenuation in the receiver module amounts to ±5mV instead of ±0.6mV, would the DAC of choice be available in the early weeks of development.

7.4. Testing and experiments

In order to verify the practical efficiency of the proposed/realized DSP system, real-life experiments and testing have to be performed. While theoretical testing was done, not enough experiments were performed with a variety of healthy heart data and heart data with arrhythmia to conclude the realized system works as intended. Due to time constraints, these experiments could not be realized, most important experiments performed by professional medical staff. While a lot of research has been done into ECG signals, not enough knowledge had been acquired to successfully interpret an ECG signal. Therefore, experiments should be performed with professional medical staff who are able to interpret ECG data and analyse the signal to see if the proposed solution is acceptable.

8 Conclusion & Recommendation

In this thesis a digital signal processing module, which included quantization and reconstruction as well, was designed and implemented. Here the aspects of the requirements that were successfully implemented are summarized along with recommendations on future research to further improve the reliability of digital signal processing part of the prototype.

8.1. Conclusion

The quantization of the 9 input signals was realized with 3 embedded ADCs in the STM32 MCU and could successfully sample a signal at 500Hz with a 16 bit resolution, thus realizing requirements **G6D.a**, **G6D.b** and **G2D**.

The digital IIR filters that were implemented could successfully remove powerline interference on the transmitter module without contamination/distortion of the input signal. Additionally, the designed filters have exceptional computational efficiency which makes them well suited for real time signal processing. Therefore, requirements **G5D** and **G7D**, which pertain to real-time processing of data, are fulfilled.

The reconstruction of digitized signals were implemented on a Nucleo development board and was tested with the prototype receiver board. The performance of the DACs in terms of reconstruction speed and frequency fell withing the specifications outlined by requirements **G6D.c** and **G6D.d**.

Unfortunately, requirement **G3D** could not be confirmed with the utmost level of certainty. While the general shape of an ECG signal from a healthy person was not distorted, experiments on heart data with arrhythmia was not done. Further testing was required to ensure requirement **G3D** was fully met, but was not possible due to time restrictions on the development window (**G10D**).

8.2. Recommendations

The prototype implemented, while adhering to the set requirements, could have been further improved. Furthermore, during the project new approaches were discovered that could not be implemented in time. These are provided here as future research prospects.

- There are filtering techniques available that make use of adaptive filters and/or neural networks that may be better at removing (PLI) noise from a contaminated ECG signal. However, these methods require a deeper knowledge of the ECG signals, more specifically different types of heart arrhythmia and their time domain characteristics. A possible research product may include the creation of a proper neural network and its implementation on an MCU such as STM32H743ZI using X-CUBE-AI.
- With the existence of optimized Fourier transforms in DSP libraries, implementing real-time signal processing on a MCU might now be possible given advancements in the processing power of micro controller processors. Further research can be done on the feasibility of real-time FIR

filtering, as this would ensure that distortion is never added in the ECG signal which adds to the reliability of the system.

- The system can be made more tightly integrated, especially with the PROT sub group by unifying the MCUs used. The feasibility of data acquisition, signal processing and transmission/re transmission on a single MCU could be a topic for future research.
- Switching noise of the DACs may also be reduced by upscaling the reconstruction rate by linear interpolation, should it not be possible to increase the sampling rate of the complete system and keep the real-time filtering aspect of the system. While increasing the DACs reconstruction rate will not completely remove the switching noise, it minimizes its amplitude. This will reduce the order of the filter that is required to remove it in the analog domain, reducing power usage, size and cost of the system.

References

[1] S. G.A.J. Custers, "Communication protocols for a wireless ecg solution," Delft University of Technology, Tech. Rep, 2022.

[2] R. v. K. P.J. Wiersma, "Circuit design for a wireless ecg device," Delft University of Technology, Tech. Rep, 2022.

[3] Ambulancezorg nederland. "Ambulancezorg nederland sectorkompas 2020, statistieken vanuit het rivm." (2022), [Online]. Available: https://www.ambulancezorg.nl/sectorkompas.

[4] M. Sejersten, O. Pahlm, J. Pettersson, *et al.*, "Comparison of easi-derived 12-lead electrocardiograms versus paramedic-acquired 12-lead electrocardiograms using mason-likar limb lead configuration in patients with chest pain," *Journal of Electrocardiology*, vol. 39, no. 1, pp. 13–21, 2006, ISSN: 0022-0736. DOI: https://doi.org/10.1016/j.jelectrocard.2005.05.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022073605002219.

[5] M. W. Dr. B. McGraw Dr. J. Lord. "Analysis and interpretation of the electrocardiogram, e-learning module, queens university of health sci." (2022), [Online]. Available: https://elentra.healthsci.queensu.ca/assets/modules/ECG/normal_ecg.html.

[6] AliveCor. "Kardiamobile 6l." (2022), [Online]. Available: https://www.kardia.com/kardiamobile6l/.

[7] Physio-control. "Lifepak 12 datasheet." (2022), [Online]. Available: https://www.physio-control.com/uploadedFiles/Physio85/Contents/Emergency_Medical_Care/Products/Operating_Instructions/LIFEPAK15_OperatingInstructions_3306222-002.pdf.

[8] Koninklijke Philips N.V. "Philips tempus als monitor." (2022), [Online]. Available: https://www.philips.nl/healthcare/product/HC989706000171/tempus-als-monitordefibrillator.

[9] ZOLL, an Ashahi Kasei Company. "Zoll heart failure management system (hfms)." (2022), [Online]. Available: https://cardiacdiagnostics.zoll.com/products/heart-failure-arrhythmia-management-system.

[10] corpuls. "Corpuls3 monitor." (2022), [Online]. Available: https://corpuls.world/en/products/corpuls3/#Monitoring-unit.

[11] L. G. Tereshchenko and M. E. Josephson, "Frequency content and characteristics of ventricular conduction," en, *J. Electrocardiol.*, vol. 48, no. 6, pp. 933–937, Nov. 2015.

[12] O. Kwon, J. Jeong, H. B. Kim, *et al.*, "Electrocardiogram sampling frequency range acceptable for heart rate variability analysis," en, *Healthc. Inform. Res.*, vol. 24, no. 3, pp. 198–206, Jul. 2018.

[13] V. Murthy, T. Grove, G. Harvey, and L. Haywood, "Clinical usefulness of ecg frequency spectrum analysis," in *The Second Annual Symposium on Computer Application in Medical Care, 1978. Proceedings.*, 1978, pp. 610–612. DOI: 10.1109/SCAMC.1978.679974.

[14] A. Habib, C. Karmakar, and J. Yearwood, *Choosing a sampling frequency for ecg qrs detection using convolutional networks*, 2020. DOI: 10.48550/ARXIV.2007.02052. [Online]. Available: https://arxiv.org/abs/2007.02052.

[15] R. Bousseljot, D. Kreiseler, and A. Schnabel, "Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das internet," *Biomed. Tech. (Berl.)*, pp. 317–318, Jul. 2009.

[16] A. L. Goldberger, L. A. Amaral, L. Glass, *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," en, *Circulation*, vol. 101, no. 23, E215–20, Jun. 2000.

[17] F. Censi, G. Calcagnini, I. Corazza, *et al.*, "On the resolution of ECG acquisition systems for the reliable analysis of the p-wave," *Physiological Measurement*, vol. 33, no. 2, N11–N17, Jan. 2012. DOI: 10.1088/0967-3334/33/2/n11. [Online]. Available: https://doi.org/10.1088/0967-3334/33/2/n11.

[18] G. Smith. "Types of a/d converters - the ultimate guide." (2020), [Online]. Available: https://dewesoft.com/daq/types-of-adc-converters.

[19] *16-bit, quad-channel, ultra-low glitch, voltage output digital-to-analog converter*, DAC8554, Texas Instruments, Oct. 2006.

[20] *Octal, 16-bit nanodac+ with i2c interface*, AD5675, Rev. A, Analog Circuits, Oct. 2010.

[21] *Quad, 16-bit, low-power, voltage output, i2c interface digital-to-analog converter*, DAC8574, Texas Instruments, Dec. 2004.

[22] *Stm32cubeide*. [Online]. Available: https://www.st.com/en/development-tools/stm32cubeide.html#:~:text=STM32CubeIDE%20is%20an%20all%2Din,for%20STM32%20microcontrollers%20and%20microprocessors..

[23] *Arduino uno rev3*. [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3.

[24] *Arduino mega 2560 rev3*. [Online]. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3.

[25] *Arduino due*. [Online]. Available: https://store.arduino.cc/collections/boards/products/arduino-due.

[26] *Portenta h7 lite*. [Online]. Available: https://store.arduino.cc/products/portenta-h7-lite.

[27] *4622 adafruit: Mouser*. [Online]. Available: https://nl.mouser.com/ProductDetail/Adafruit/4622?qs=7MVldsJ5UaxIlY4FBCFg7w%3D%3D.

[28] *Nucleo-f411re stmicroelectronics: Mouser*. [Online]. Available: https://nl.mouser.com/ProductDetail/STMicroelectronics/NUCLEO-F411RE?qs=Zt3UNFD9mQjdEJg18RwZ2g%3D%3D.

[29] *Nucleo-h743zi2 stmicroelectronics: Mouser*. [Online]. Available: https://nl.mouser.com/ProductDetail/STMicroelectronics/NUCLEO-H743ZI2?qs=lYGu3FyN48cfUB5JhJTnlw%3D%3D.

[30] P. Bing, W. Liu, Z. Wang, and Z. Zhang, "Noise reduction in ecg signal using an effective hybrid scheme," *IEEE Access*, vol. 8, pp. 160 790–160 801, 2020. DOI: 10.1109/ACCESS.2020.3021068.

[31] Y. Weiting and Z. Runjing, "An improved self-adaptive filter based on lms algorithm for filtering 50hz interference in ecg signals," in *2007 8th International Conference on Electronic Measurement and Instruments*, 2007, pp. 3-874-3–878. DOI: 10.1109/ICEMI.2007.4351057.

[32] H. Gholam-Hosseini, H. Nazeran, and K. Reynolds, "Ecg noise cancellation using digital filters," in *Proceedings of the 2nd International Conference on Bioelectromagnetism (Cat. No.98TH8269)*, 1998, pp. 151–152. DOI: 10.1109/ICBEM.1998.666440.

[33] H. K. Jayant, K. Rana, V. Kumar, S. S. Nair, and P. Mishra, "Efficient iir notch filter design using minimax optimization for 50hz noise suppression in ecg," in *2015 International Conference on Signal Processing, Computing and Control (ISPCC)*, 2015, pp. 290–295. DOI: 10.1109/ISPCC.2015.7375043.

[34] S. L. Joshi, R. A. Vatti, and R. V. Tornekar, "A survey on ecg signal denoising techniques," in *2013 International Conference on Communication Systems and Network Technologies*, Apr. 2013, pp. 60–64. DOI: 10.1109/CSNT.2013.22.

[35] E. Ajdaraga and M. Gusev, "Analysis of sampling frequency and resolution in ecg signals," in *2017 25th Telecommunication Forum (TELFOR)*, 2017, pp. 1–4. DOI: 10.1109/TELFOR.2017.8249438.

[36] mains frequency. "Measurement of the mains frequency." (2022), [Online]. Available: https://www.mainsfrequency.com/index.htm.

[37] swissgrid. "Frequency." (2022), [Online]. Available: https://www.swissgrid.ch/en/home/operation/regulation/frequency.html.

[38] entsoe. "[press release] continuing frequency deviation in the continental european power system originating in serbia/kosovo: Political solution urgently needed in addition to technical." (2018), [Online]. Available: https://www.entsoe.eu/news/2018/03/06/press-release-continuing-frequency-deviation-in-the-continental-european-power-system-originating-in-serbia-kosovo-political-solution-urgently-needed-in-addition-to-technical/.

[39] R. Klabunde. "Cardiovascular physiology concepts." (2019), [Online]. Available: https://cvphysiology.com/Arrhythmias/A009.

[40] "CMSIS DSP Software Library." (), [Online]. Available: `https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html`.

[41] *User manual: Description of stm32h7 hal and low-layer drivers*, UM2217 Rev 5, STMicroelectronics, Jul. 2020.

[42] *Reference manual: Stm32h742, stm32h743/753 and stm32h750 value line advanced arm®-based 32-bit mcus*, RM0433 Rev 7, STMicroelectronics, Feb. 2020.

[43] "Measuring code execution time on ARM Cortex-M MCUs." (2018), [Online]. Available: `https://embeddedcomputing.com/technology/processing/measuring-code-execution-time-on-arm-cortex-m-mcus`.

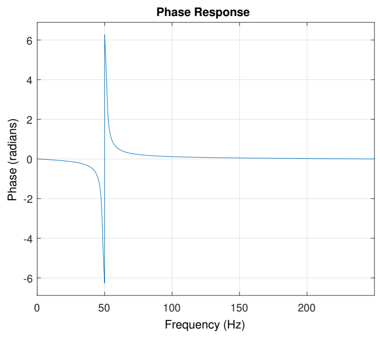
[44] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001. DOI: `10.1109/51.932724`.

[45] S. Narayanaswamy, "High resolution electrocardiography," en, *Indian Pacing Electrophysiol. J.*, vol. 2, no. 2, pp. 50–56, Apr. 2002.

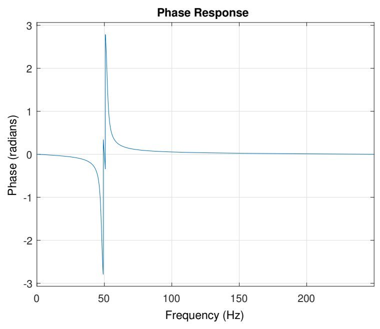
[46] Tektronix. "Waveform generator software." (), [Online]. Available: `https://www.tek.com/en/products/software/arbexpress-signal-generator`.

A Extra figures

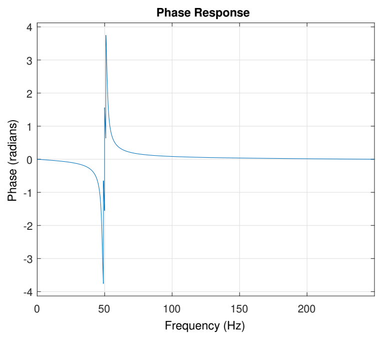
A.1. IIR designs



(a) 8th order Butterworth

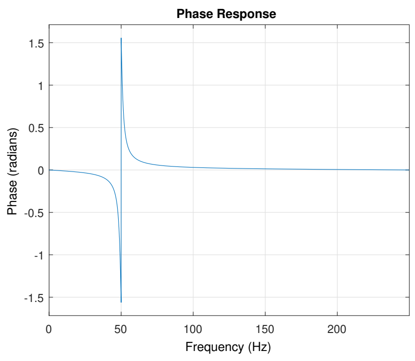


(b) 4th order ellicptic

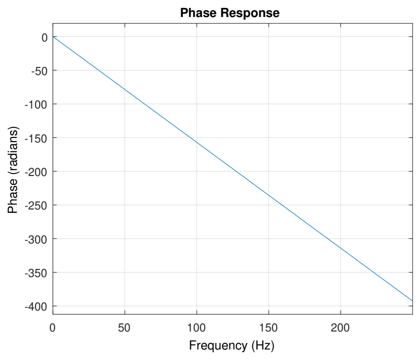


(c) 6th order Chebyshev type II

Figure A.1: Phase response of different IIR band-stop filters

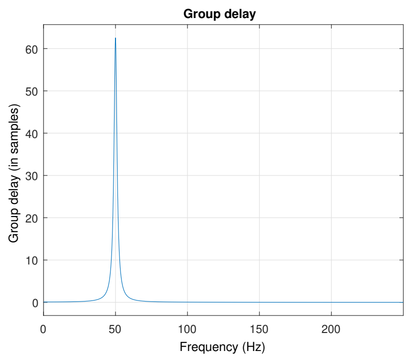


(a) Notch filter

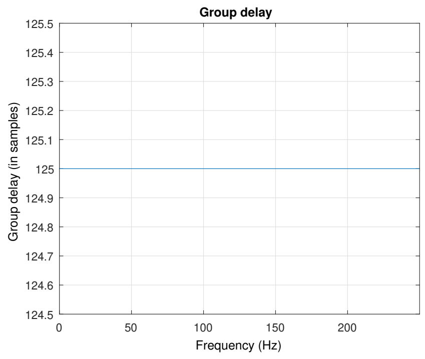


(b) 250^{th} order hamming band-stop filter

Figure A.2: Phase response comparison of IIR notch filter and FIR band-stop filter



(a) Notch filter



(b) 250^{th} order hamming band-stop filter

Figure A.3: Group delay comparison of IIR notch fiter and FIR band-stop filter

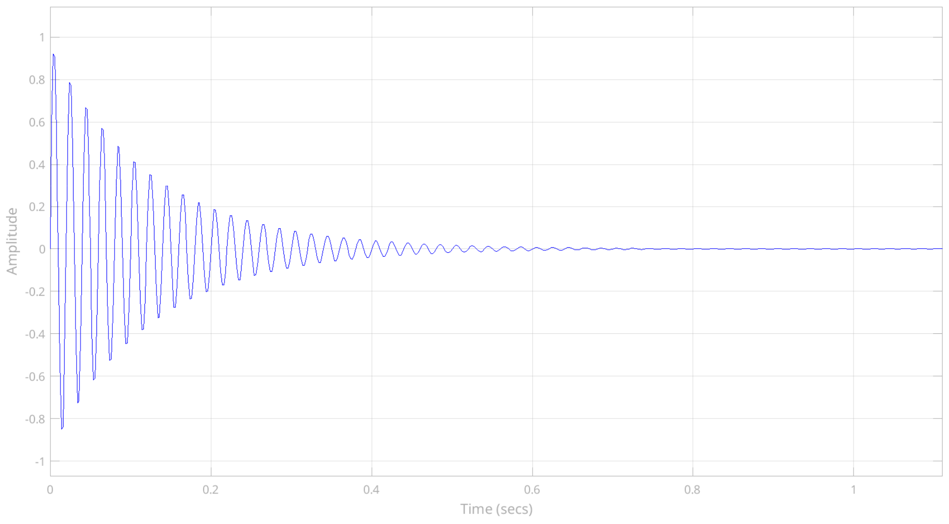
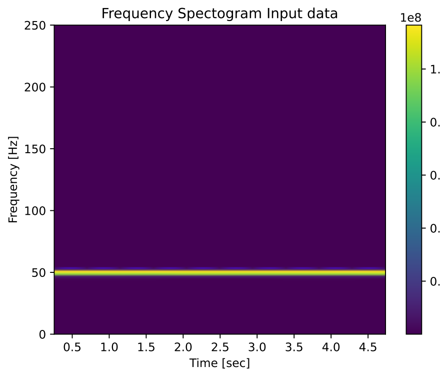
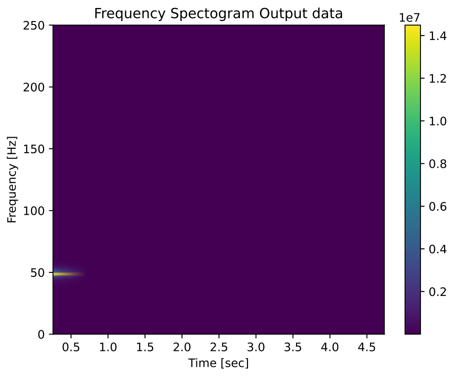


Figure A.4: Convergence of IIR filter attenuation over time

A.1.1. ECG Signal Filter Testing

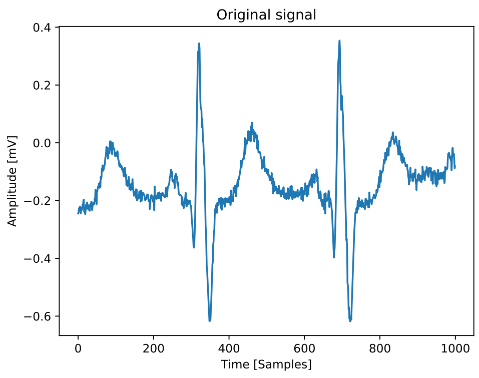


(a) Filter input Frequency-domain

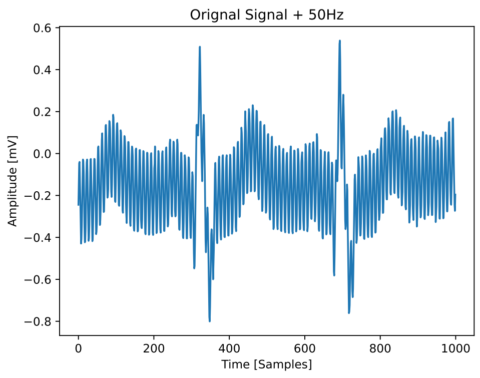


(b) Filter output Frequency-domain

Figure A.5: Filter input and output in the frequency-domain



(a) Original ECG Signal



(b) ECG Signal with 50 Hz added

Figure A.6: ECG Signal - Patient 001

B
Complementary Information

B.1. Notch filter transfer functions

In the main body of the thesis, only the transfer function for a 50Hz notch filter was included with precision up to 5 decimal points. In the implementation the precision is higher and thus, the transfer function with unrounded coefficients are included in this appendix.

- 50Hz Notch filter:

$$H(z) = \frac{0.984260526926093116 - 1.59256698635129967z^{-1} + 0.984260526926093116z^{-2}}{1 - 1.59256698635129967z^{-1} + 0.968521053852186231z^{-2}} \tag{B.1}$$

- 100Hz Notch filter:

$$H(z) = \frac{0.984260526926093116 - 0.608306459425206669z^{-1} + 0.984260526926093116z^{-2}}{1 - 0.608306459425206669z^{-1} + 0.968521053852186231z^{-2}} \tag{B.2}$$

- 150Hz Notch filter:

$$H(z) = \frac{0.984260526926093116 + 0.608306459425206447z^{-1} + 0.984260526926093116z^{-2}}{1 + 0.608306459425206447z^{-1} + 0.968521053852186231z^{-2}} \tag{B.3}$$

- 200Hz Notch filter:

$$H(z) = \frac{0.984260526926093116 + 1.59256698635129945z^{-1} + 0.984260526926093116z^{-2}}{1 + 1.59256698635129945z^{-1} + 0.968521053852186231z^{-2}} \tag{B.4}$$

B.2. DAC Operation

TRx: Temporary Register x of channel x

DACx: Output channel x of a single DAC8574

Table B.1: 4-byte operation of TI DAC8574

Slave Address+R/$\bar{W}$	Control Byte	2-Byte Data	Function
1 0 0 1 1 A1 A0 0	A3 A2 0 0 X Sel1 Sel0 0	MSB - LSB	Write data to TRx selected by Sel1-Sel0, on the DAC with matching A3-A0 pins
1 0 0 1 1 A1 A0 0	A3 A2 0 1 X Sel1 Sel0 0	MSB - LSB	Write data to TRx selected by Sel1-Sel0 and immediately update output of DACx, on the DAC with matching A3-A0 pins
0 0 1 1 A1 A0 0	A3 A2 1 0 X Sel1 Sel0 0	MSB - LSB	Write data to TRx selected by Sel1-Sel0 and update all 4 Channels, on the DAC with matching A3-A0 pins
1 0 0 0 0 0 0 0	X X 1 1 X 0 X X	X - X	Broadcast command for all DACs on the I^2C line to update all 4 channels with data in TRs

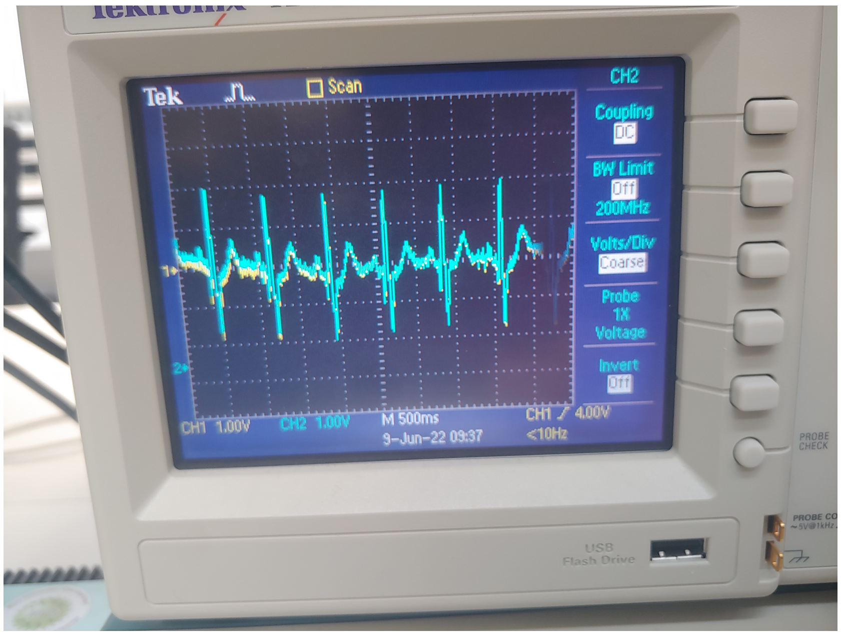


Figure B.1: DAC generated output on Channel 2 of an osciloscope

C
Signal creation

In order to independently test different parts of the system proposed, a function generator (Tektronix AFG 3012B) was used to output an ECG signal. This appendix will outline the method followed to load an ECG signal to an arbitrary waveform of the Tektronix AFG 3012B function generator.

1. Obtain up to 1000 samples to form into an arbitrary waveform sampled at a constant time interval
2. Install ArbExpress software from Tektronix [46]
3. Create a .csv file with a format as in Table C.1

Table C.1: Waveform format

(a) General format

#CLOCK={Sample rate oscilloscope}
#SIZE={Amount of samples in waveform (N)}
{Sample 1 value}
{Sample 2 value}
...
{Sample N value}

(b) Example format

#CLOCK=1.0000000000e+08
#SIZE=1000
0.209376141
0.219020873
...
0.736418009

4. Load the .csv file into the ArbExpress software by clicking: File -> Open (Cntrl + O key shortcut) and selecting the .csv file. When loading in the example format, ArbExpress wil look like Figure C.1
5. Save the waveform on an USB-device as a .tfw file by clicking: File -> Save as -> Save as type: AFG TFW(*.tfw)
6. Plug in the USB device in the function generator
7. Load the arbitrary waveform from the USB-device: Function: Arb -> Arb Waveform Menu -> Memory: USB -> Select waveform.
   For the example waveform, the result can be seen in Figure C.2
8. Set frequency, phase, amplitude and offset on the function generator as needed

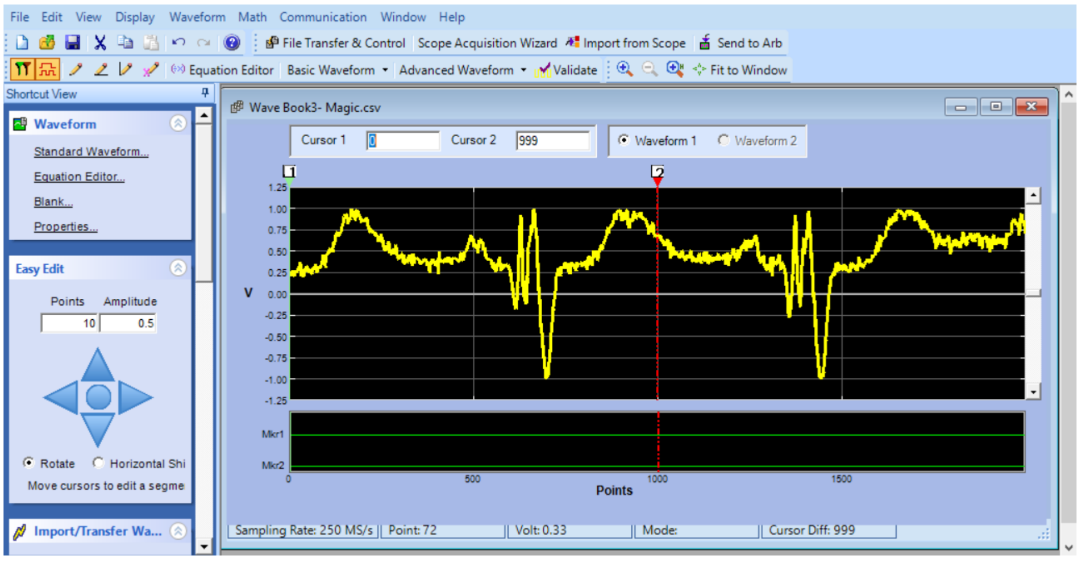


Figure C.1: ArbExpress waveform loaded

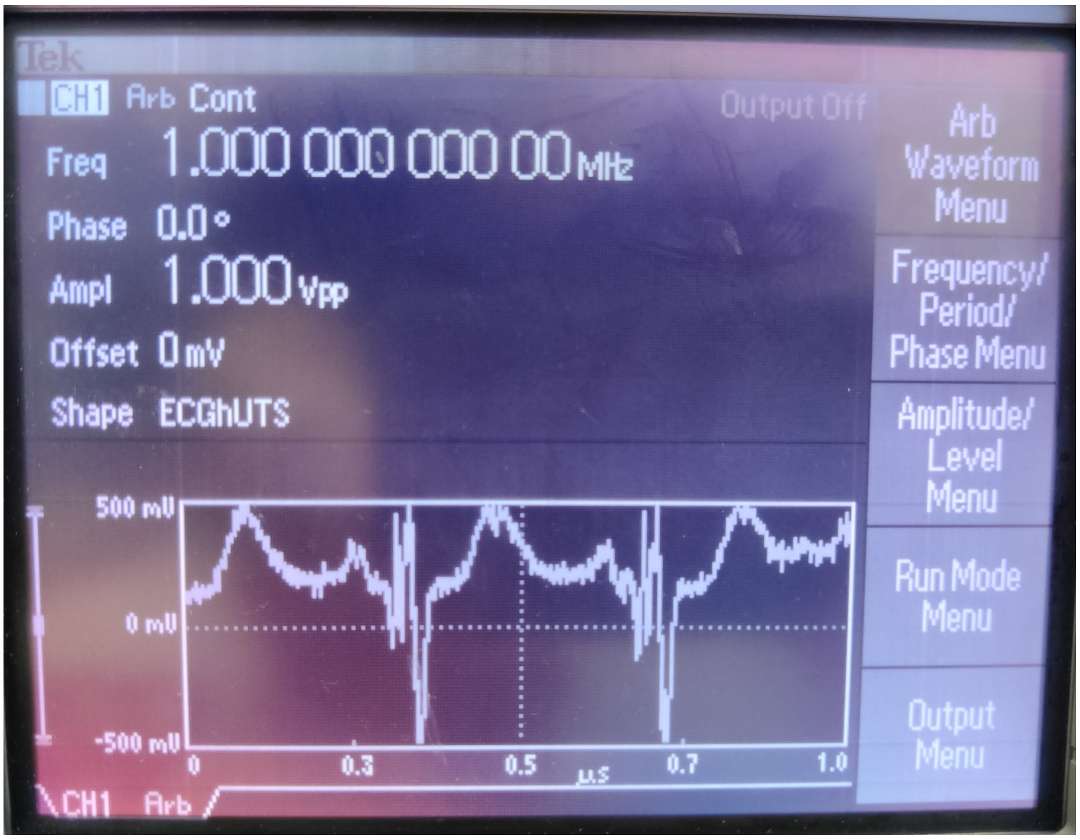


Figure C.2: Waveform loaded on Tektronix AFG 3021B