Developing a Multi-Lead Wireless ECG

A Major Qualifying Project submitted to the Faculty of WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the Degree of Bachelor of Science

Submitted by:
Emma Fountain
Ben Guerriero
Josh Reeder
Logan Young

Other Contributors:
Nisha Goel

Advised By:
Professor Edward A. Clancy
Professor Songbai Ji

March 2022

Acknowledgements

We would like to thank the following individuals for their contributions towards this project's success: Professor Edward Clancy, Professor Songbai Ji, He Wang, and Jianan Li. Without their continued guidance, support, and feedback in all aspects of the project including components, technical explanations, understanding and improving of MATLAB algorithms we would not have achieved what we can present today.

Abstract

This report focuses on the design of a multi-lead wireless electrocardiogram (ECG) monitor, the design and implementation of an R-wave detector in MATLAB, and the design of an atrial fibrillation detector in MATLAB. The circuit has electrodes that collect ECG signals from a subject and sends the signals to a Nordic Microcontroller (nRF52840) that uses Bluetooth to wirelessly transfer the data to a computer. Signal processing is done on the collected signal using an R-wave detector and atrial fibrillation detector. An ECG circuit using an instrumentation amplifier-based front end, linear filtering, and ADC conversion was designed on a protoboard, solderboard, and on a PCB. Performance analysis yielded CMRR readings of approximately 81 dB and Input Referred Noise of approximately 0.8 μV RMS RTI. An R-wave detector was implemented in MATLAB by cascading a bandpass filter (fourth-order Butterworth, 5-15 Hz passband), squaring operation, lowpass filter (second-order, 5 Hz cutoff), and a peak detector. We also implemented an atrial fibrillation detector in MATLAB that classifies beat segments based on calculated values for Shannon entropy and the time varying coherence function, TVCF. Both algorithms produce high sensitivity and predictivity ratings of 70% or greater on various test subject data.

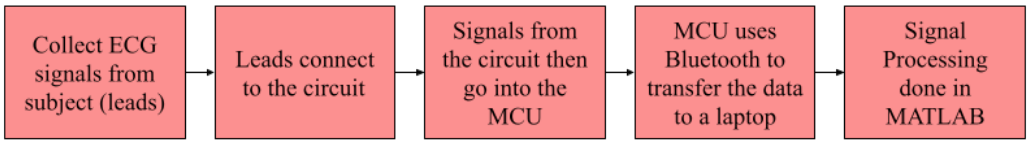
Table of Contributions

Section	Primary Author(s)	Primary Editor(s)
Abstract	Nisha Goel	Joshua Reeder
1: Introduction	Nisha Goel, Ben Guerriero	All
2.1 The Heart and ECG Signals	Emma Fountain	Nisha Goel
2.2.1: 12 Lead ECG	Logan Young	Joshua Reeder, Emma Fountain
2.2.2: 3 Lead ECG	Ben Guerriero	Logan Young
2.2.3: 2 Lead ECG	Ben Guerriero	Logan Young
2.2.4: Modern ECG Applications	Joshua Reeder	Ben Guerriero
2.3: Analog Front End and all subsections	Joshua Reeder	Ben Guerriero
2.4.1: Heart Rate	Emma Fountain	Nisha Goel
2.4.2: R-wave Detection Algorithm and all subsections	Nisha Goel	Emma Fountain, Joshua Reeder
2.4.3: Heart Rate Variability	Nisha Goel	Emma Fountain
2.4.4: Atrial Fibrillation and all subsections	Emma Fountain	Nisha Goel
2.5: Standards & Specifications Used	Logan Young	All
3.1: Original Client Statement	Provided by Project Advisor	N/A
3.2: Objectives and Constraints	Nisha Goel	Joshua Reeder
3.3: Project Approach	Nisha Goel	Joshua Reeder
3.3.1: Tools	Nisha Goel, Joshua Reeder	Joshua Reeder
3.4: Revised Client Statement	Emma Fountain	All
4.1: Instrumentation Amplifier	Logan Young	Ben Guerriero
4.2: Operational Amplifier	Logan Young	Ben Guerriero
4.3: Analog to Digital Converter	Joshua Reeder	Ben Guerriero
4.4: Voltage Regulator	Ben Guerriero	Joshua Reeder
4.5: Microcontroller	Joshua Reeder	Ben Guerriero
4.6: Multi-Leads Selection	Nisha Goel	All
4.7.1: R-Wave Detection	Nisha Goel	Emma Fountain
4.7.2: Atrial Fibrillation Detection	Emma Fountain	Nisha Goel
5.1: Design Process	Logan Young	Nisha Goel, Joshua Reeder
5.1.1: PCB Layout	Nisha Goel	Emma Fountain
5.2: Hardware Design Verification in Multisim	Ben Guerriero	Nisha Goel, Joshua Reeder
5.3: Hardware Design Verification on a Protoboard	Joshua Reeder	Nisha Goel, Joshua Reeder
5.4: Hardware Design Verification on a Solderboard	Logan Young	Nisha Goel, Joshua Reeder
5.5: Discussion	Ben Guerriero	Nisha Goel, Joshua Reeder
6: Embedded and all subsections	Josh Reeder	
7: R-Wave Detection Algorithm and all subsections	Nisha Goel	All
8: Atrial Fibrillation Detection Algorithm and all subsections	Emma Fountain	Nisha Goel
9: Final Design	Logan Young, Ben Guerriero, Emma Fountain, Josh Reeder	All
10: Discussion	Nisha Goel, Ben Guerriero, Emma Fountain	All
11: Conclusions and Recommendations	Logan Young, Emma Fountain	All

Executive Summary

The study of electrocardiogram (ECG) signals produced by the human heart has allowed technological and medicinal advancements over the last 100^{+} years. In more recent years, information that ECGs provide have been used to diagnose heart ailments such as atrial fibrillation (an irregular rapid heartbeat), heart attacks, and heart disease. Looking at an ECG trace provides information on the locations of where blood flow may be reduced or where there are sudden rapid heartbeats, and this allows for quick medical intervention when necessary.

Our Major Qualifying Project was to build a battery-powered, three-lead wireless ECG monitor that will be comfortably mounted on the human body for an extended period of time, collect the subject's ECG signals, and wirelessly transmit the signals for analysis using an R-wave detector and an atrial fibrillation detector. The project was split into three major sections: hardware, R-wave detection algorithm, and atrial fibrillation detection algorithm. The diagram below shows the overall flow of how our system is intended to function:

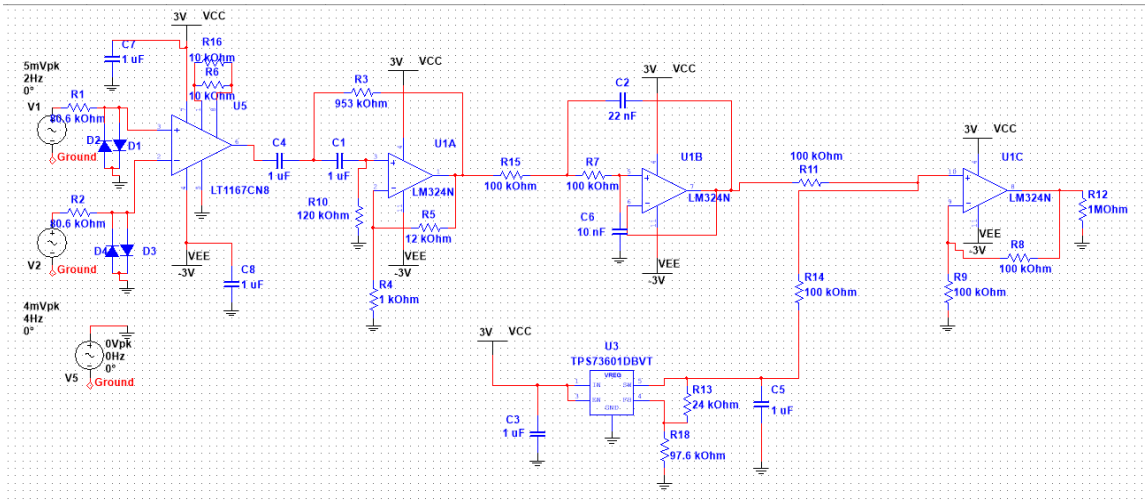


High Level Blackbox Diagram of System

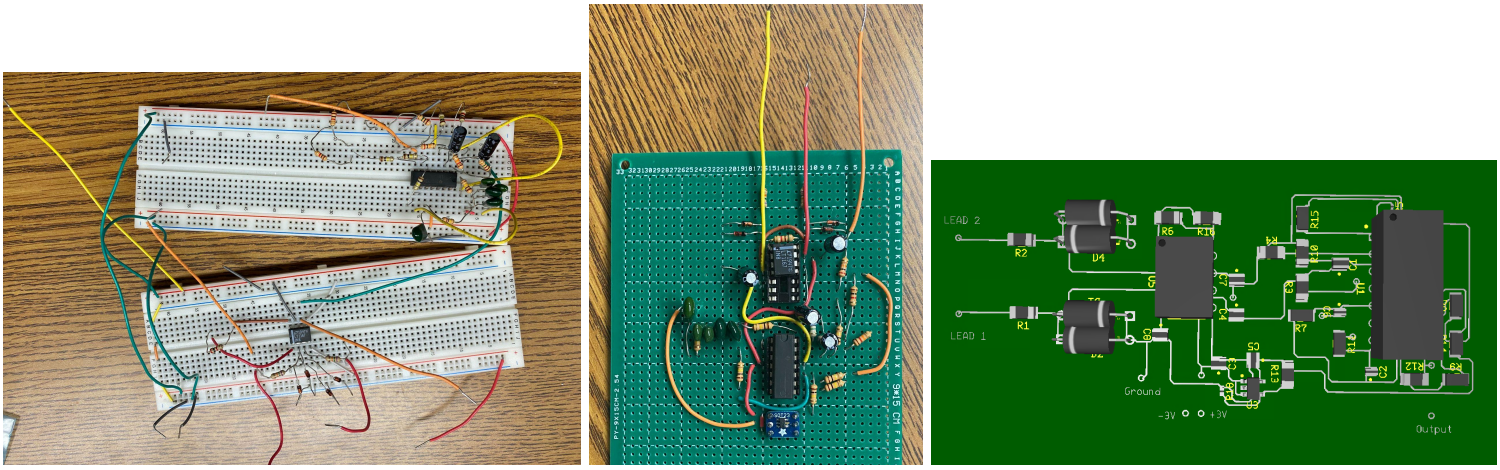
The hardware portion consisted of building an ECG circuit, programming the Nordic (nRF52840) Microcontroller units (MCU), and testing all components individually and when combined to produce satisfactory ECG signals. The signal processing portion of the project was to design and implement an R-wave detection algorithm and an atrial fibrillation detection algorithm. Once written, the detection algorithms were first tested on the MIT/BIH arrhythmia and atrial fibrillation databases (atrial fibrillation detection algorithm only used the atrial fibrillation database) and then optimized for accuracy.

The ECG circuit was built using an LT1167 instrumentation amplifier, an LM324N quad operational amplifier, and a TPS73615MDBVREP voltage regulator. These components and a series of basic components (resistors and capacitors) comprised the entirety of the circuit design. The ECG signal is processed through an instrumentation amplifier stage to take the difference and amplify the signal between two electrodes. Then a cascade of highpass and lowpass stages are implemented to shape the waveform to the desired frequency range and remove external

artifacts. Finally, the signal is level shifted upwards to properly adjust to a unipolar power supply for an analog to digital converter (ADC). We simulated the circuit in NI Multisim and created a protoboard, solderboard, and PCB version of the circuit, as shown in the figures below. The three circuit inputs represent the electrodes of the three ECG wires, represented as the three voltage sources, and the singular output, represented as the final resistor, is connected directly to the Microcontroller in our final applications.



ECG Circuit Schematic



ECG circuit on a protoboard (left), solderboard (middle), and PCB mock-up (right)

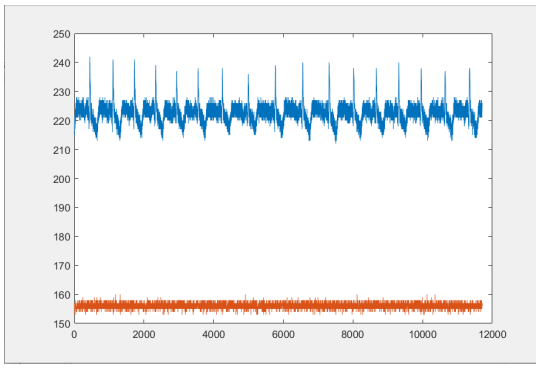
The circuits were tested against a series of performance requirements, the results of which are shown in the table below.

System	Wireless ECG Requirement	Measured
Input Amplitude	± 2 - 4 mV	Satisfies requirement
Input Frequency	0.05 - 150 Hz	lowpass cutoff of 150Hz and highpass cutoff of 0.05 Hz
Selectable Gain	600 for Full Scale Range of ADC	300 (hall of FSR)
CMRR	≥ 80 dB @ 60 Hz	≈ 81 dB @ 60 Hz
Input-Referred Noise	≤ 1 μV RMS RTI full band	≈ 0.8 μV RMS RTI
Size	Approximately the size of two fists	30 in² (5 in x 6 in)
Voltage	3V maximum	No component requires more than 3V
ADC	12-bit resolution	12-bit resolution onboard Microcontroller
Isolation	Medically isolated	Medically isolated
Current Consumption	≤ 20 mA @ 3 V	2.1 mA @ 3V

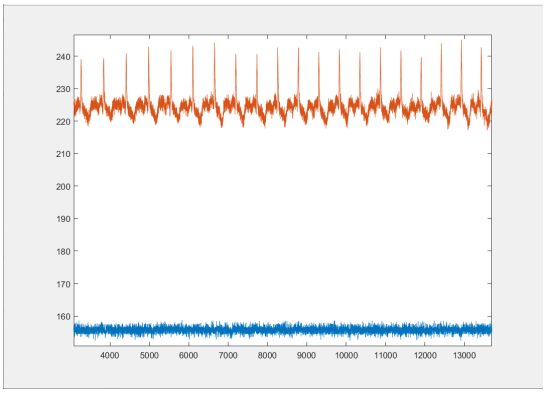
Table of ECG System Design Requirements and Measured Performance

Upon completion of the hardware processing, the signals generated from the heart are processed through a series of embedded programs and transmitted through the Nordic Microcontroller. The transmission and reception is completed via the utilization of three separate Microcontrollers. The peripheral node Microcontroller is connected to the output of the solder board or PCB board. A secondary transmission Microcontroller acts as a dead signal and transmits no relevant data (Included so that the project could use an available system configuration). Finally, the third and final central node Microcontroller, responsible for data reception, is powered and connected to a laptop or desktop. A series of MATLAB scripts and applications visualize the data received and produce a recorded heart rate across the entire interval of time. The signals do undergo real-time software filtering as well. A 60 Hz power line notch filter and an additional lowpass filter are applied to the signal before MATLAB

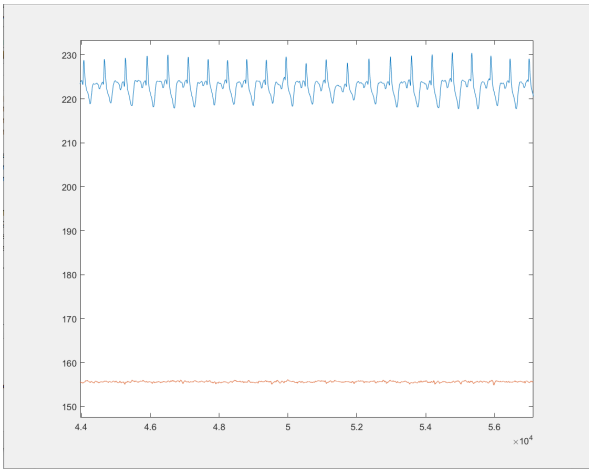
visualization to smooth out the signal. Images of the ECG signal without any software filtering, with just the notch filter, and with the final lowpass filter can be seen in the following figures. All three figures had data collected from the same subject within a ten minute period of collection.



Amplitude vs. Samples of an ECG Signal without Filtering

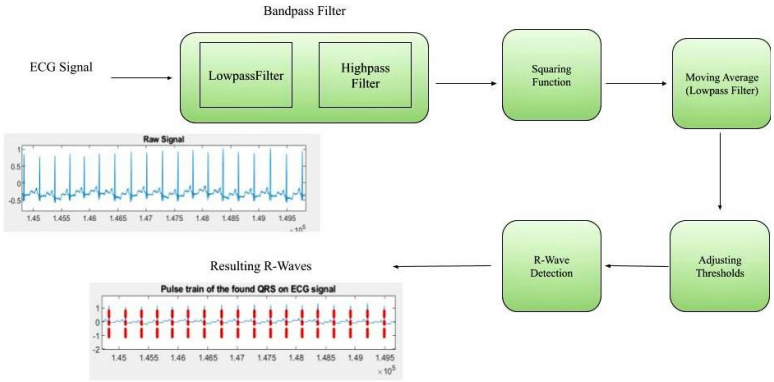


Amplitude vs. Samples of an ECG Signal with a 60 Hz Notch Filter



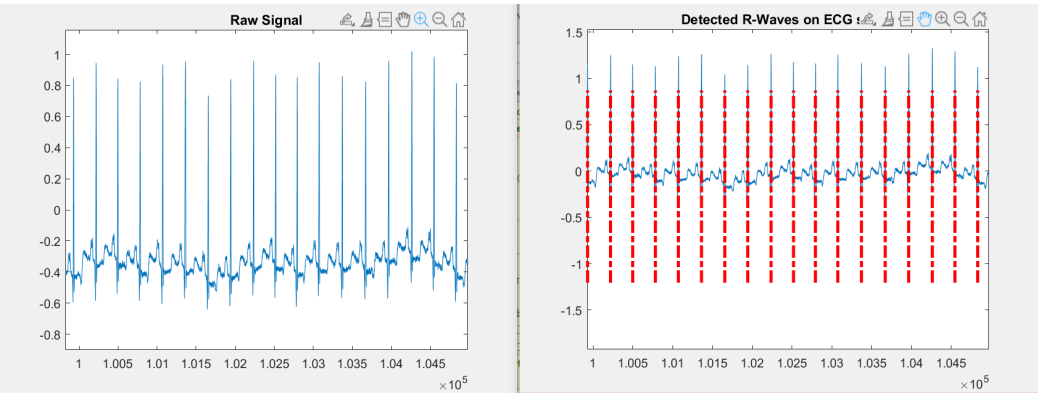
Amplitude vs. Samples of an ECG Signal including 150 Hz Butterworth lowpass Filter

The second part of our project was to write detection algorithms in MATLAB. The R-wave detection algorithm was implemented using Pan-Tompkins' methodology. We designed a fourth-order Butterworth bandpass filter with a passband of 5-15 Hz, a squaring function, and a second-order Butterworth lowpass filter with a cutoff frequency of 5 Hz using MATLAB filter design functions. After that, we implemented a peak detector using thresholds that automatically update with each new peak detected. The blackbox diagram below illustrates the flow of the signal being filtered and the R-waves being detected.



Blackbox diagram of the R-wave detection signal flow

Once the detection algorithm was complete, we tested it using the MIT/BIH arrhythmia and atrial fibrillation databases, which combined, had a total of 71 subjects, over 255 hours of recordings, and the recordings included normal sinus rhythm, arrhythmias (such as PVCs), and atrial fibrillation. We were able to produce graphs of whole recordings and overlay vertical red dashed lines where detections are found, as shown in the figure below:



Left graph shows normal sinus rhythm for Lead I of subject 100, showing 4500 samples of data. With a sampling rate of 200Hz. Red dashed lines in the right graph mark each detected R-wave, using the same data.

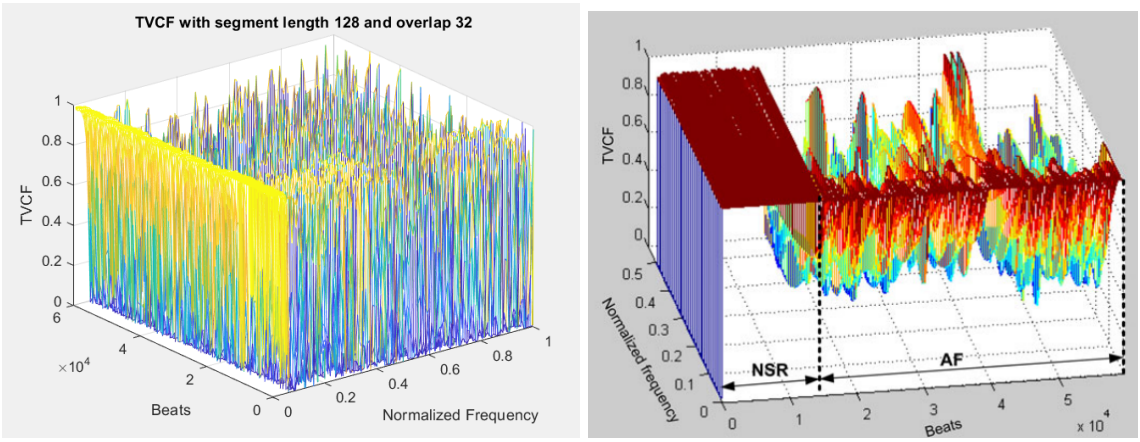
Our initial algorithm produced a QRS sensitivity of 85% or higher for 64/71 of the records we tested from both the Physionet arrhythmia and atrial fibrillation databases, and a QRS positive predictivity of 91% or higher for 69/71 of the records from both databases when tested on the first channel of the ECG array. There were seven "outlier" datasets, which we later realized was because six out of the seven recordings had very low amplitudes. To fix this amplitude issue, we attempted to use the second channel in the ECG array for all the records. Six out of the seven outliers from our first test produced a QRS sensitivity of 98% or higher and positive predictivity of 97% or higher; however, some of the records that performed well using Channel 1 did not perform as well using Channel 2, producing QRS sensitivities of 50% or less. This makes sense, because generally one channel has lower amplitudes and poor SNR, and for those records, Channel 1 produced the optimal signal. There were also records that performed well using both channels, with minor changes in the QRS sensitivity and positive predictivity.

We also modified the bandpass and lowpass filter orders and cutoff frequency and tested all the records with the modifications to determine the best parameters with the highest QRS sensitivity and positive predictivity. Specifically we changed our second-order lowpass filter to a first-order lowpass filter and modified the bandpass filter cutoff frequency from 5-15 Hz to 5-12 Hz. With this modification, we produced a QRS sensitivity of 85% or higher for 61/71 of the traces we tested from both the Physionet arrhythmia and atrial fibrillation databases, and a QRS positive predictivity of 91% or higher for 70/71 traces from both databases. While we were able to remove one of the seven outliers that the first set of testing produced, our overall improvement was not significant.

Our initial hypothesis was that modifying the cutoff frequency and changing the order of our filters would fix all of our outlier datasets, however, simply changing the channel significantly improved the accuracy for the outliers. Additionally, there is one record in the atrial fibrillation database (subject 07162) that has low values for every modification that we tried. To make this algorithm more accurate, more complex logic would need to be implemented to ensure better QRS sensitivity and QRS positive predictivity for noisier signals with very high amplitudes.

The atrial fibrillation detection algorithm created was based on the methods used by Lee et al.[1] RR intervals were calculated using R-wave locations found from an R-wave detector. The time varying coherence function (TVCF) was applied to segments of 128 RR intervals and

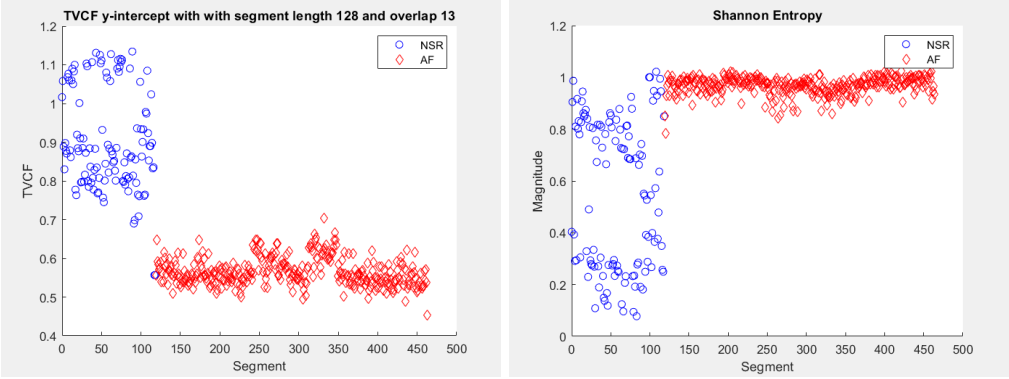
the results analyzed. The window size used by Lee et al. in their TVCF calculations was changed for our algorithm because we used a nonparametric versus their parametric approach which resulted in noisier data as seen in the figure below. To smooth the data we decreased the window size and averaged multiple spectral estimates within the 128 sample interval.



Plot of TVCF data for MIT/BIH Atrial Fibrillation subject 08455 with RR interval segment length of 128. Our data left and Lee et al. data right

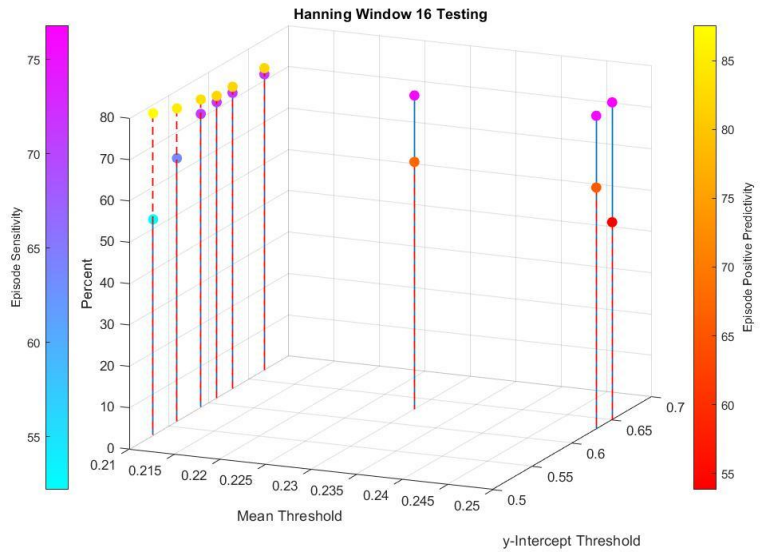
Instead of calculating the frequency variation of the TVCF as done by Lee et al., a line was fit vs. frequency for each TVCF segment. The y-intercept of the fitted line and the mean across the frequency spectrum were found to be good indicators of atrial fibrillation and were therefore used in our classification algorithm.

Another indicator that was used to classify atrial fibrillation was Shannon entropy. It measures the chance that patterns exhibiting regularity over some duration of data will also exhibit similar patterns over the next duration. Both the y-intercept and Shannon entropy values for subject 08455 are shown below.



Y-intercept (left) and Shannon entropy (right) with classification from MIT/BIH Atrial Fibrillation annotations for subject 08455

Segments are classified as atrial fibrillation if they have a Shannon entropy above 0.79, a y-intercept below 0.59, and a mean below 0.21. These threshold values were found when the hanning window size was 16 and the overlap 15. The results of testing used to determine these thresholds is shown below.



Average Sensitivity and Positive Predictivity plot for Hanning Window of 16

Ultimately, the goals of this report were accomplished. We completed our ECG circuit schematic, implemented it on a protoboard and solderboard, and designed a PCB. We were able to design and implement an R-wave detector in MATLAB, test on the MIT/BIH arrhythmia and atrial fibrillation databases, vary parameters such as filter orders and cutoff frequencies, and compare the differences between them, which resulted in us sharing our recommendations for how to further improve the algorithm. The atrial fibrillation algorithm was able to calculate the TVCF, the y-intercept of the frequency spectra, and the Shannon entropy for RR interval segments. The algorithm uses this information to classify segments as atrial fibrillation or normal sinus rhythm. Most importantly, data was generated through the circuitry, transmitted wirelessly through the Microcontrollers, visualized in real time, and applied to the algorithms we have developed. These deliverables were made possible through extensive research and design done early in the project.

Table of Contents

Table of Figures

Table of Tables

1: Introduction

The human body creates electrical signals mediated by neural impulses, often initially triggered by stimuli. Stimuli can include a sound or pressure, but chemical release by other neurons can also trigger nerve impulses. Some electrical signals that can be measured from the body emanate from the heart, eye muscles, stomach muscles, and the brain, just to name a few. The primary focus of our research was on the heart and how to measure and analyze the electrical signals, electrocardiograms (ECG), it produces. Additionally, our research was focused on how to detect issues in the heart using ECG signals in order to diagnose illnesses. Finally, we focused on the hardware side, primarily how to build an ECG circuit and all the component specifications that we looked into to determine the best option for our circuit, since our objective was to build a multi-lead wireless ECG monitor.

ECG has historically been measured using tethered wired measurement systems, but the newest generation of devices, such as exercise ECG monitors and the Apple Watch, are both wireless and untethered. These systems allow people to see their heart signals in a non-clinical setting, get their heart rate (as opposed to their pulse rate that most devices such as Fitbit provide), and also record the traces so that they can be reviewed at a later time. Our ECG monitor also follows the new generation of devices, being wireless and untethered, and is used to record ECG signals for a long period of time for clinical use. It is intended to be small so that it can be mounted on the body for an extended period of time and collect ECG signals, which can be wirelessly sent to a computer for signal processing. For the signal processing, an R-wave detector and atrial fibrillation detector are used so that if the subject has atrial fibrillation, an episode can be detected. The ideal wireless system will be low power and use small components for efficiency and convenience and the ideal signal processing algorithms will be highly accurate and efficient.

2: Background

This chapter focuses on ECGs and ECG signals, the different types of algorithms that exist to analyze raw ECG signals, and the technical research that we did to understand the components needed to build our ECG circuit. We used literature written by experts in the fields to understand and later determine how we want to create our multi-lead wireless ECG monitor, R-wave detection algorithm, and atrial fibrillation detection algorithm.

2.1 The Heart and ECG Signals

The heart is the organ responsible for pumping blood throughout the body. Through the contraction of the cardiac muscles of the heart, blood moves between four chambers and circulates throughout the body and lungs. The four chambers of the heart consist of the right and left atria, and right and left ventricles. These chambers are separated by valves that only permit the flow of blood in one direction. The location of the chambers and valves can be seen in Figure 2.1. Deoxygenated blood moves into the right atrium (RA) then into the right ventricle (RV) where it is then pumped to the lungs and becomes oxygenated. The oxygenated blood moves to the left atria (LA) and finally into the left ventricle (LV). The LV is responsible for pumping oxygenated blood to the rest of the body. This whole process is part of the cardiac cycle. [2]

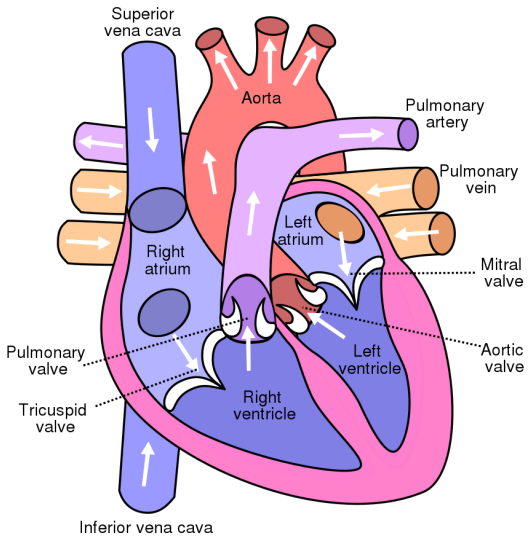


Figure 2.1: Basic heart anatomy [3]

The cardiac cycle is triggered by the sinoatrial node. This group of cells, located in the right atria, acts as the "heart's natural pacemaker" and sends a signal to the RA to contract. The signal is then passed to the atrioventricular node where there is a delay of about 0.1 seconds providing adequate time for the ventricles to fill with blood. The signal is then passed to the Bundle of His and then Purkinje fibers which are located in the inferior aspect of the heart and in the lateral aspects of the right and left ventricles**.** This signal causes the two ventricles to contract at the same time. The conduction system is depicted in Figure 2.2. The impulses in the heart create an electric potential that can be measured and used to calculate heart rate and diagnose many heart conditions. [4]

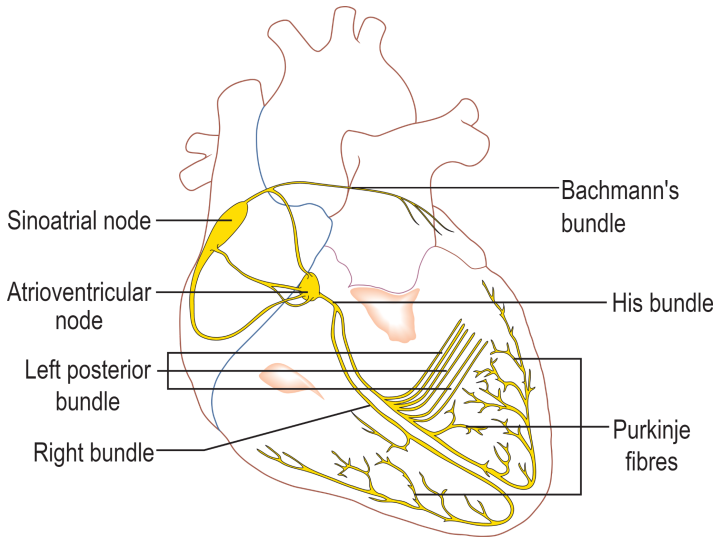


Figure 2.2: Electrical nodes of the heart [5]

An electrocardiogram (ECG, used interchangeably with EKG) is used to record electrical signals from the heart. There are both non-invasive and invasive tests available which use electrodes and leads attached onto specific sites of the body. Figure 2.3 shows the lead connections for non-invasive tests. The electrodes are then connected to electronics that transduce the signals from the body. LA corresponds to the left arm, RA corresponds to the right arm, LL corresponds to the left leg, and RL corresponds to the right leg.

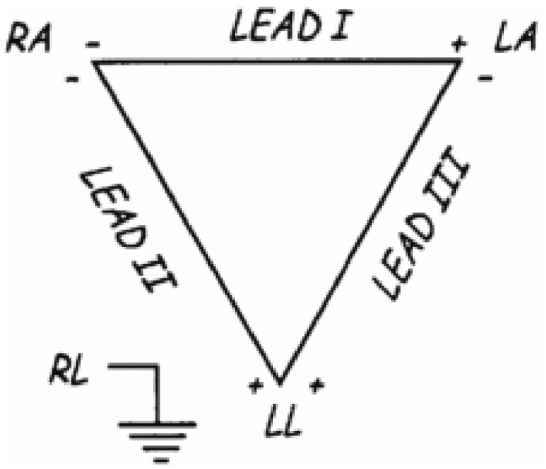


Figure 2.3: ECG Electrode Placement [6]

A normal ECG during one heartbeat resembles the signal in Figure 2.4. Figure 2.4 also identifies the five waves in an ECG: the P-wave, QRS complex, and T-wave. The P-wave corresponds to atrial depolarization, which is the contraction of the atria. The QRS wave corresponds to the ventricular depolarization and atrial repolarization, which means that the ventricles contract and the atria relaxes. Finally, the T-wave corresponds to the ventricular repolarization, where the ventricles relax. Thereafter, the next P-wave begins [7]**.**

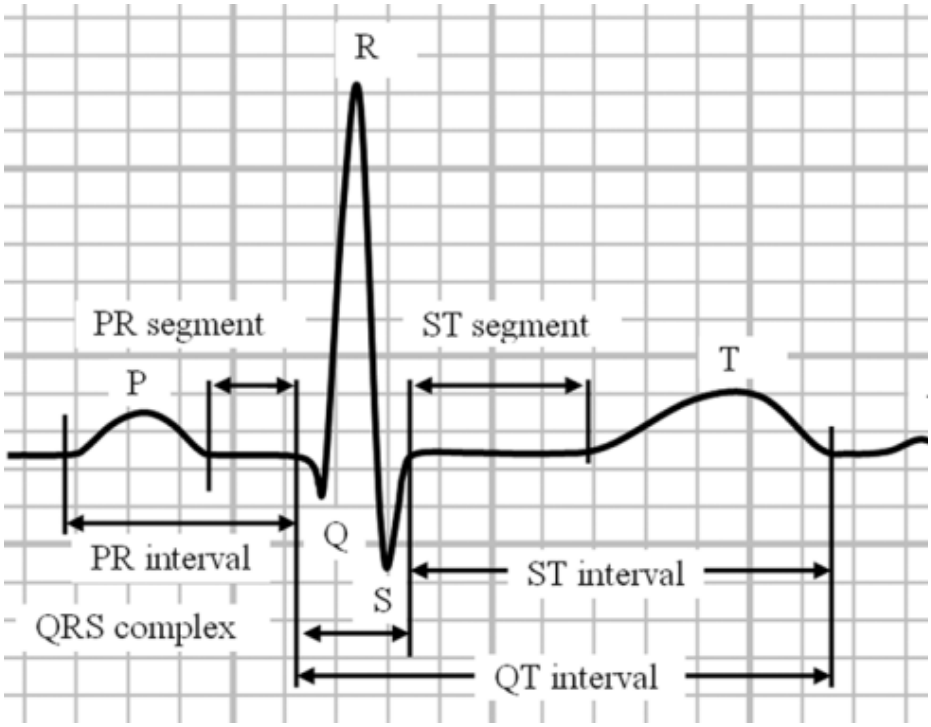


Figure 2.4: An ECG Signal with Labels [8]

It's important to note the QRS complex labeled in Figure 2.4 because later we will discuss a QRS detection algorithm, which can be used for extracting information on how the heart signal looks and for determining if the patient has any medical conditions. All information from the ECG signal is important, including the duration between each wave, in determining abnormalities.

Types of ECG abnormalities include beat and rhythm abnormalities. Beat abnormalities that originate in a location other than the SA node are called ectopic beats, the most common are premature ventricular contractions and premature atrial contractions [9]. These are characterized by their short followed by long R-wave-like impulse between normal beats [1]. Rhythm abnormalities are indicated by the heart beating too fast (tachycardia), too slow (bradycardia), or irregularly. The most common irregular heart rhythm is atrial fibrillation which will be discussed in a later section [10].

2.2: ECGs Today

ECG recordings are used in clinical environments to make sure that the heart is functioning as intended. These recording devices are built to be monitored by professionals, specifically 12-lead ECGs. In addition to the 12-lead ECGs used in professional environments such as hospitals, there are modern innovations in technology that have made electrocardiogram recordings more accessible and practical for average consumer use. Devices such as the Apple Watch (series 4 and onwards) and the Fitbit (Sense or Charge 5) have revolutionized the ECG field by creating a means of analyzing your heart from your wrist.

2.2.1: Twelve Lead ECG

ECG recordings are extremely common procedures in doctors' offices, clinics, and hospitals. They are noninvasive procedures that pose essentially no risk of electrical shock and allow the monitoring of electrical signals going through the heart [11].

Higher end systems found in hospitals tend to record twelve leads in order to best observe the pattern of the heart. The system is referred to as a twelve lead system, but it is actually commonly composed of ten electrodes on the body. There are six precordial electrodes placed on the chest in a semi-circular pattern above the heart. These unipolar electrodes around the heart

are identified as V1, V2, V3, V4, V5, and V6. These electrodes are all measuring data with respect to the common voltage reference which, in this 12 lead case, is the electrode on the right leg. V1 and V2 measure the right ventricle while V3 and V4 measure activity in the left ventricle and the front of the heart. Lastly, V5 and V6 cover the side of the heart. This formation is so that the leads can generate more amplitude from the signals in the heart with alternative “views" of the heart’s electrical activity as well as less noise when compared to the remaining electrodes on the extremities [12].

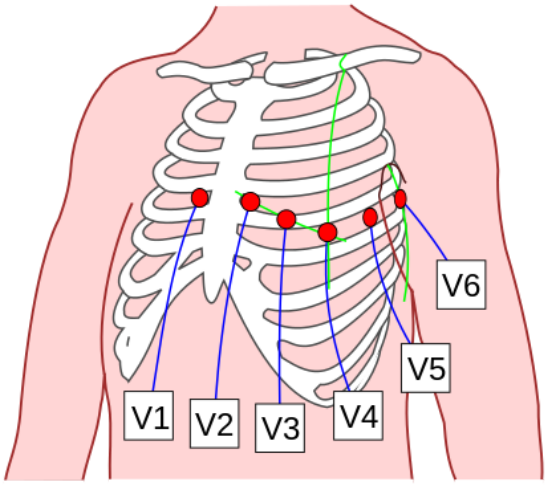


Figure 2.5: Placement of six unipolar electrodes [13]

With each heart contraction, blood is pumped through the body propagating the heart beat’s signal. As the heart contracts and pumps blood it moves around slightly in the chest. This movement causes the cleanest signal produced by the heart to move as well. The heart is a 3 dimensional object and as a result to get a full view of the heart the 6 electrodes are very important to get a full reading. The heart’s movement can make the signal hard to read and with the electrodes placed around the heart the ECG is able to use the different electrodes to create a more accurate signal as well [11].

The remaining four limb electrodes are put in the four corners of the body with one on the left arm (LA), one on the right arm (RA), one on the left leg (LL), and one on the right leg (RL). In a twelve lead ECG, the right leg electrode acts as the ground for the system, with the benefit of minimizing ECG artifact [14].

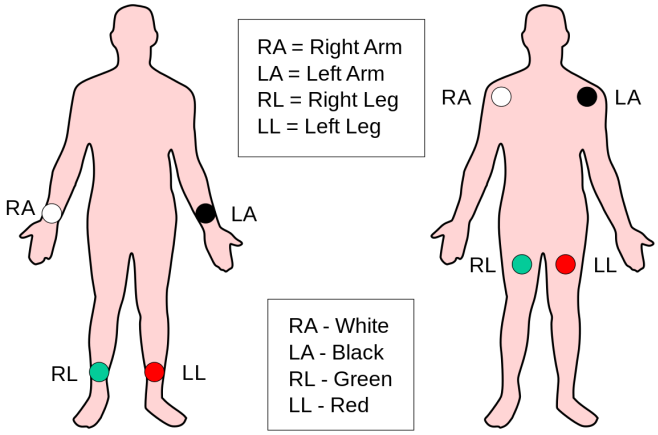


Figure 2.6: Placement of four dipole electrodes [15]

With four limb electrodes it is possible to receive 6 leads worth of data in a system known as Einthoven's Triangle. The principle behind Einthoven's Triangle is that the electrodes record not only relative to themselves which produce leads aVR, aVL and aVF but also relative to each other resulting in bipolar leads I, II, and III. This forms an equilateral triangle that we refer to as Einthoven's Triangle (Figure 2.7) [11].

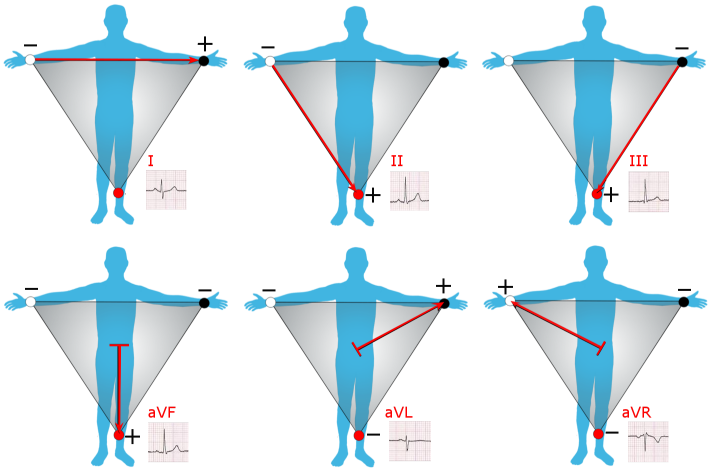


Figure 2.7: Einthoven's Triangle [16]

2.2.2: Three Electrode ECG

A 12 lead ECG system is standard for use in hospitals, as it provides the most information and reliability. A 12 lead ECG requires 10 electrodes, which are not practical for a system that would be worn on the body outside of the hospital. For a system designed to be worn

for long periods of time without restricting a person's normal daily activity, an ECG device with fewer electrodes is desired. As mentioned above, ECG measurements can also be taken using only 3 electrodes. Using Einthoven's Triangle, voltage differences between the electrodes are measured with electrodes placed at the shoulders and left leg [37]. Lead I is defined between the right and left shoulders, lead II is between the right shoulder and left leg, and III is between the left shoulder and left leg. This configuration allows an equilateral triangle to be formed with each lead equidistant from the heart. Since the triangle forms a closed loop, the sum of the voltages must be zero according to Kirchoff's laws. In Einthoven's triangle, the polarity of lead II is reversed. Therefore, the voltage at lead II is subtracted from the sum of I and III to equal a total voltage of zero.

In addition to placing electrodes around the body, a circuit must be constructed to view the ECG measurements taken from the electrodes. The voltage difference at lead I will be amplified using an instrumentation amplifier. The electrode placed on the right leg acts as a reference node to ground to help reduce noise in the system. The resulting wave after the difference is amplified must be filtered to block out interference at unwanted frequencies outside of the ECG signal range. Additional amplification may also be required. The ECG circuit operation and relevant components are discussed in later sections.

2.2.3: Two Electrode ECG

The purpose of using a 3 electrode ECG device is to make a system that is more comfortable for the user. Reducing the number of necessary electrodes for ECG measurement from 3 to 2 would create an even more convenient system. In Einthoven's Triangle, the ECG measurements are taken using the difference at lead I and an additional electrode is placed at the right leg as a ground reference. Removing this ground node and creating a virtual ground between the other two electrodes still allows for ECG measurements to be taken [38]. However, the resulting signal would likely have additional noise that would not be seen with 3 electrodes. It is possible to see clear measurements with only 2 electrodes, but additional filtering is usually required.

2.2.4: Modern ECG Applications

Two modern practices include the placement of electrodes on your chest or the use of a single chest strap. The first method, requiring the placement of electrodes across your body, is the Holter Monitor. Holter Monitors are often requested by healthcare providers when symptoms such as dizziness, fatigue, low blood pressure, fainting, and palpitations are prevalent and the resting ECG measurements are inconclusive. Holter Monitors, shown in the image below, include the placement of five electrodes across the chest. The electrodes are wired to the ECG reading device (the Holter Monitor) and readings are recorded across a 24 hour period.

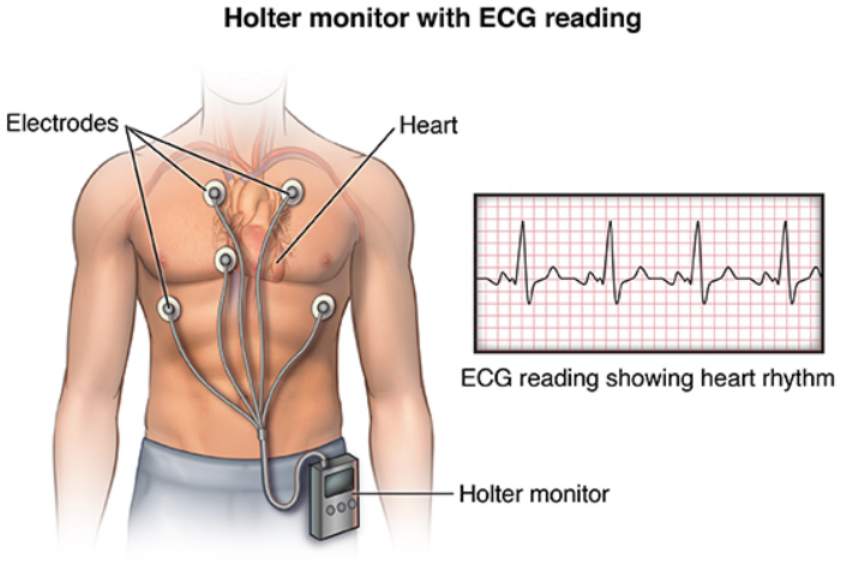


Figure 2.8: Holter Monitor Demonstration [17]

The second method of chest worn ECG recording devices is the use of a single compact chest strap. The Polar H7 Heart Rate Sensor and Fitness Tracker is a compact, Bluetooth-capable fitness device that provides features such as measuring heart rate, recognizing heart rate variability, and counting calories. The device is also waterproof and has 200 hours of battery life, making the device practical for endurance athletes. A validation study for the device, consisting of a variety of male subjects (age, body composition, and fitness level), was conducted and the results demonstrated that the H7 produces interchangeable results with lab based ECG readings at rest, excellent agreement during exercise, and reasonable agreement during intense exercise.

Apple Watches perform pulse rate and electrocardiogram measurements through the use of two features within the watch. By utilizing the pulse rate monitor on the underside of the Apple Watch, along with placing an individual's finger on the digital crown to the side of the watch, the watch can collectively record and present valuable and relatively reliable data on your heart. The pulse rate monitor on the underside of the Apple Watch uses a process known as photoplethysmography (PPG). The PPG sensor emits green LED light paired with photodiodes to detect how frequently an individual's blood is pushed through the arteries in their wrist. By flashing the LED light hundreds of times per second the watch can recognize the variation in light absorbed, thus, creating a proper pulse rate measurement (this sensor is capable of recording between 30 and 210 beats per minute). The digital crown has built-in electrodes that, when a fingertip from your contralateral hand is placed on top, create a closed circuit between your heart and both arms. Apple advertises the watch as capable of recording recognition of sinus rhythm (depolarization of the heart beginning in the sinus node characterized by correctly oriented P-waves), atrial fibrillation (heart beating irregularly, the most common form of serious arrhythmia), a low or high heart rate, and an inconclusive result. As of the most current application software, the Apple Watch is capable of measuring heart rate readings within the range of 50 to 150 beats per minute.

The Fitbit Sense and Charge 5 operate similarly to the Apple Watch. Both the Sense and Charge 5 utilize sensors within the frame of the watch along with their biosensor core to record ECG measurements when an individual places their fingers to the sides or corners of the watch frame. Possible results from Fitbit measurements as advertised include sinus rhythm, atrial fibrillation, and inconclusive.

Without the addition of the digital crown in the Apple Watch or the sensors within the Fitbit frames to close the circuit across an individual's arms and chest, the watches would only be capable of measuring pulse rate (a significant distinction). Heart rate, measured with ECG devices, reads the quantity of times that the heart contracts per minute. Pulse rate, on the other hand, reads the pulse of blood flow through the arteries (caused by the contraction of the ventricles). Apple Watch and Fitbit both utilize green LED lights to calculate the blood flow through the capillaries, thus, without the additional sensors, would only be measuring pulse.

An emerging substitute for the Apple Watch or Fitbit is a wearable PPG recording device that can be worn on the finger. The Oura Ring is a multisensory wellness device that weighs

roughly 4 grams and can survive for one week on a single battery charge. Through studies within a relatively small sample group (60 participants) it was determined that the PPG sensors within the Oura Ring are capable of producing nearly identical measurements to the ECG recording devices provided to the group. These measurements, however, are limited to the ring's capability of reading only pulse rate or pulse rate variability. While the Oura Ring is less capable than an Apple Watch or Fitbit, the small size and comfortability provides the unique opportunity to collect nightly readings across an extended period of time when compared to its competitors.

Finally, further revolutions within the field are bringing ECG measurements to headphones. In 2019, Heart.Zone released headphones with continuous ECG reading capability. The technology uses electrical contacts within the ears and on the skin, and its performance is advertised as unaffected by sweat, hair, or wobbling due to motion. The Application Programing Interface (API) within the headphones enables communication with cellular applications and provides real time visualization of the ECG measurements as well as the capability to store the data measurements on a phone or in the cloud. In late 2020, Apple announced their commitment to create a miniature version of their PPG sensor for use within Beats headphones.

2.3: Analog Front End

The upper limit of ECG voltage amplitude on the skin while the heart is beating in regular sinus rhythm is 2.5 - 3.0 mV and the typical frequency range for ECG readings is 0.5 to 150 Hz [18]. ECG readings can be measured utilizing two different methods, bipolar or unipolar. Bipolar leads register the voltage difference between two electrodes on the skin while unipolar leads compare the difference between the heart's activity and what is known as an indifferent or Wilson electrode. The indifferent electrode acts as a virtual reference (negligible voltage activity). Given the voltage and frequency ranges, and in understanding the functionality of unipolar and bipolar ECG readings, to properly analyze the electrical signals generated from an individual's heart the circuitry must be capable of taking the difference, amplifying, and filtering the signal to a reasonable viewing standard. The following section will discuss the methodology in doing so along with relevant explanations of the chosen circuit components.

2.3.1: Differential and Instrumentation Amplifiers

As mentioned previously, bipolar ECG readings measure the electrical potential difference between corresponding electrodes attached to the skin and unipolar readings compare the heart's activity with a virtual reference. Therefore, circuitry that can take the difference between the two electrical potentials and output that difference is required. Furthermore, typical voltage readings within sinus rhythm are in the range of millivolts. The circuit must also be capable of amplifying the resulting difference for proper viewing. Figure 2.9 shows an example of circuitry capable of this functionality.

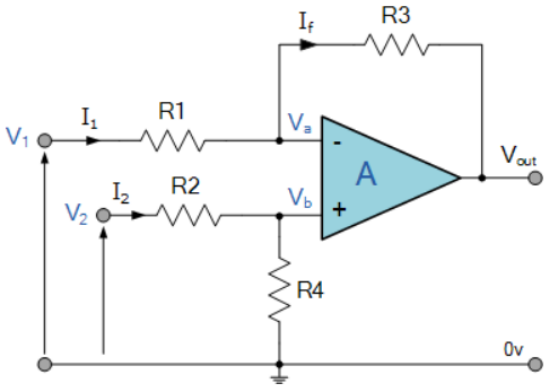


Figure 2.9: Differential Amplifier Circuit

Differential amplifier circuits utilize both inverting and non-inverting amplifier configurations. If the input voltage V_2 is 0, the circuit assumes the role of an inverting amplifier with input-output expression:

$$V_{OUT} = (-R3/R1)V_1 \qquad (2.1)$$

If the input voltage V_1 is 0, the circuit assumes the role of a non-inverting amplifier with input-output expression:

$$V_{OUT} = (R3/R2)V_2 \qquad (2.2)$$

Through applying the superposition principle to the two previous equations, and equating R2 to R1 as R and R3 to R4 as RR, an input-output expression for the entire differential amplifier is:

$$V_{OUT} = (V_2 - V_1)(RR/R) \qquad (2.3)$$

The gain of a difference amplifier is reliant on the resistance values within the expression above. Namely, for there to be significant gain, the value of R must be smaller than RR. The differential amplifier functions properly while inputs to the circuit have a low output impedance.

However, when input impedances are similar to resistance R, -as is the case for ECG- the measurement becomes faulty. To resolve this issue, the following circuit maintains the amplification functionality of the difference amplifier with the necessary additional high input impedance. This elaborated circuit, shown in Figure 2.10, is referred to as an instrumentation amplifier.

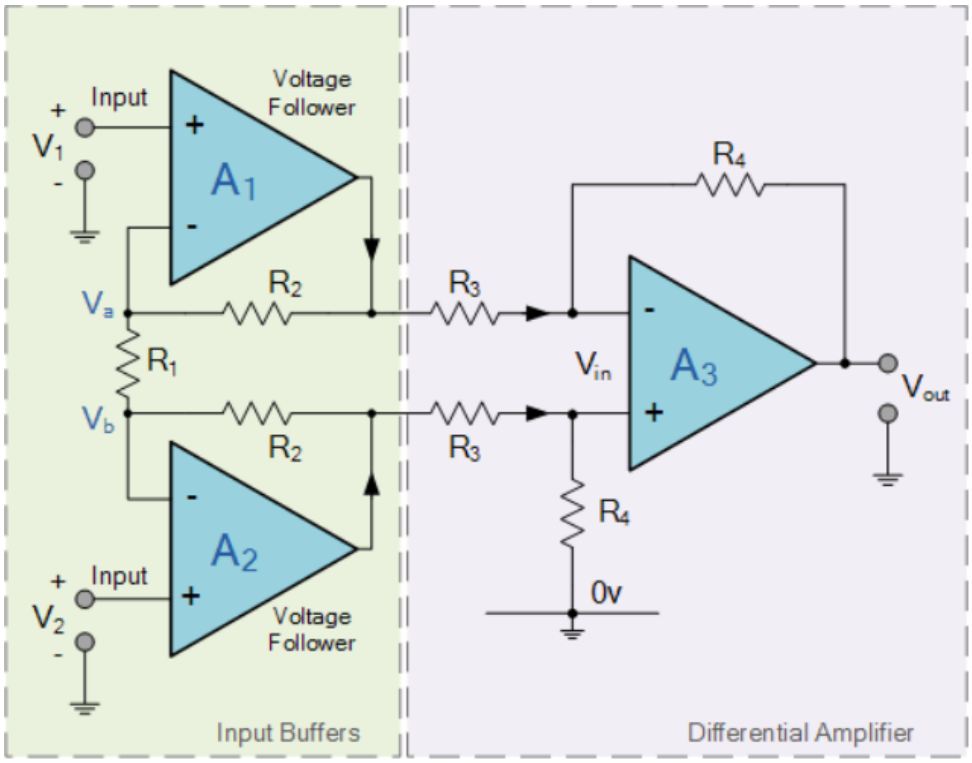


Figure 2.10: Instrumentation Amplifier Conceptual Design

The instrumentation amplifier introduces two non-inverting amplifiers to the input of the difference amplifier. The output of each non-inverting "buffer" amplifiers serve as the input to the differential amplifier and can be modeled with input-output expression:

$$V_{OUT} = (1+ 2R2/R1)V_{IN} \quad (2.4)$$

By combining the expression for the non-inverting amplifiers with the previous input-output expression for the differential amplifier, the entire input-output expression for the instrumentation amplifier is obtained as (Konar et al. 2019):

$$V_{OUT} = (V_2 - V_1)(1+ 2R2/R1)(R4/R3) \quad (2.5)$$

The instrumentation amplifier circuit forces the output impedance of the non-inverting amplifiers to serve as the input impedance of the difference amplifier. Therefore, the impedance

can be selected to prevent interference with the differencing. Ultimately, the final circuit produces high input impedance to the instrumentation amplifier and low output impedance from the instrumentation amplifier. Analog Devices's AD620 instrumentation amplifier is one of the two models used in preliminary ECG circuit testing for the group. The AD620 is capable of producing gains of up to 10,000 while drawing a maximum current supply of 1.3 mA. The low power capabilities of the AD620 make it ideal for battery powered operation, thus making the amplifier advantageous for future design implementations. Finally, the AD620's gain is set by an external programmable R1 resistor and the unit has eight pins. Analog Devices' AD624 on the other hand, has gain set by programmable pins, with the chip providing gains of 1, 100, 200, 500, and 1000 by pin selection. There is also the capability to set gains between 1 and 10,000 with a programmable resistor. The AD624, unlike the AD620, has sixteen external pins including input and output null pins.

2.3.2: Filtering

Typical ECG measurement produces frequencies between 0.05 and 150 Hz (for non-infants) in the passband. The QRS complex, the most significant portion of the ECG wave for the group's measurements, falls within 8-50 Hz. Furthermore, a collection of other unwanted artifacts, including baseline wander, powerline interference, EMG noise, and electrode motion artifacts amongst others also prevent a clean ECG signal reading. Therefore, it is necessary to use a variety of filtering techniques to eliminate the unwanted artifacts.

As aforementioned, the pin layout on the AD624 differs significantly from the AD620. Input pins 4 and 5 on the AD624 are the designated input null pins. These pins have the capability of nullifying the potential 300 mV offset voltage within the differentiator. DC offset measures the voltage across a gel-to-gel electrode pair due to the difference in their electrode half-cell potentials. In an ideal situation the potentials would be identical, however, impurities within the metal electrode or variations within the gels produce a maximum DC offset of 300 mV [19].

Utilizing the AD620 instrumentation amplifier removes the built-in input nulling functionality that the AD624 provides, however, the filtering techniques used after each amplifying stage will be identical. Active highpass and lowpass filters would be cascaded

withthe output of both amplifiers to form a bandpass filter. Sallen-Key filters are commonly used highpass and lowpass circuit topologies given the easily cascadable lowpass and highpass stages. Figures 2.11 and 2.12 are the topologies of the second order highpass and lowpass Sallen-Key filters respectively:

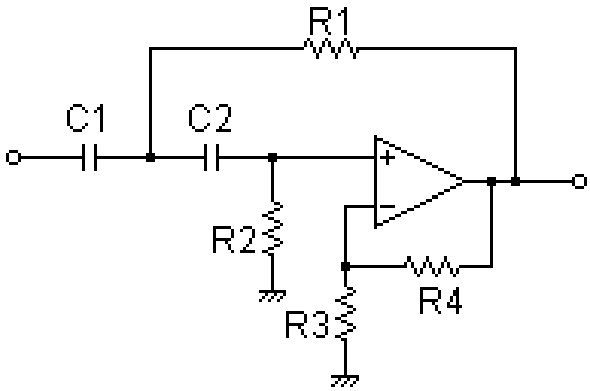


*Figure 2.11: Second order Sallen-Key highpass filte*r

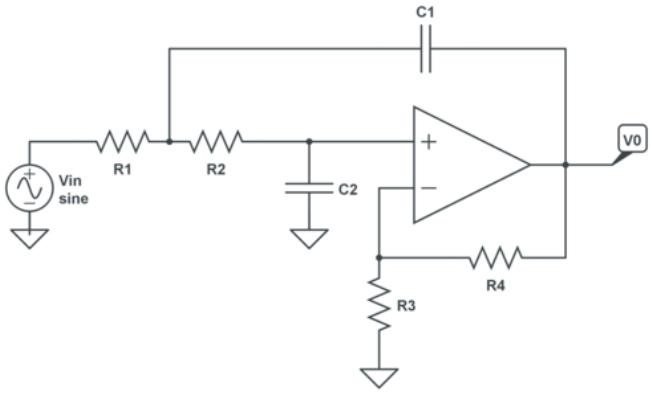


*Figure 2.12 : Second order Sallen-Key l*owpass filter

The transfer functions of the Sallen-Key second order highpass filter and second order lowpass filter respectively are [20]:

$$H(s) = \frac{\frac{R_3-R_4}{R_3}s^2R_1R_2C_1C_2}{s^2(R_1R_2C_1C_2) + s(R_2(C_1+C_2) + R_1C_2(1-\frac{R_3+R_4}{R_3})) + 1} \quad (2.6)$$

$$L(s) = \frac{\frac{R_3+R_4}{R_3}}{s^2(R_1R_2C_1C_2) + s(C_1(R_1+R_2) + R_1C_2(1-\frac{R_3+R_4}{R_3})) + 1} \quad (2.7)$$

With the ideal bandpass filter, a flat magnitude response is desired in the pass band. Butterworth filters are often referred to as “maximally flat filters” due to their flattest pass band magnitude response (when compared with other filter designs such as Chebyshev or Elliptic). Seen below is the ideal magnitude response for a Butterworth lowpass filter:

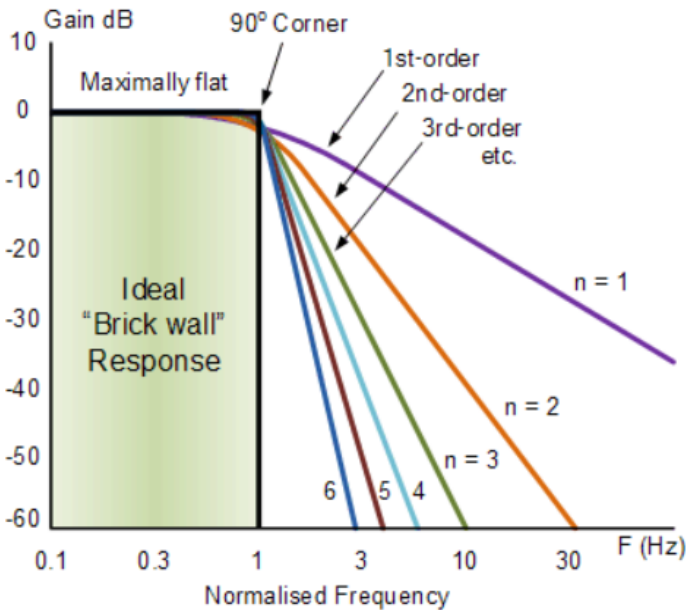


Figure 2.13: Ideal Frequency Response for a Butterworth Lowpass Filter

Butterworth filters can be constructed by cascading Sallen-Key active filters. As can be seen from Figure 2.13, the higher order the filter the steeper the roll off. Therefore, higher order filters are desirable to attenuate the various artifacts. However, this comes at the cost of a greater number of circuit components, greater power consumption, and differences with impulse and step response. To construct the necessary bandpass filter, the output of the instrumentation amplifier is passed through an n^{th} order highpass filter then through an n^{th} order lowpass filter. The active highpass filter occurs before the lowpass filter to kill the potential offset present after the instrumentation amplifier. If there was no potential for a DC offset, there would be no difference between the order of the highpass and lowpass stages.

2.3.3: Summing Amplifier Level Shift

Following the filtering stage of the ECG circuit is a necessary level shifting phase. This stage is necessary due to the input voltage readings to the circuit swinging both positive and negative, with the negative parts being amplified throughout the circuit. The entirety of the circuit prior to the level shifting stage operates off a bipolar power supply. The analog to digital converter (ADC) on the Microcontroller, however, operates off unipolar power. Given the switch from bipolar to unipolar power supply, an upward shift of the voltage is necessary. This level

shift is accomplished with a summing amplifier stage. An example of a summing amplifier is shown below in Figure 2.14.

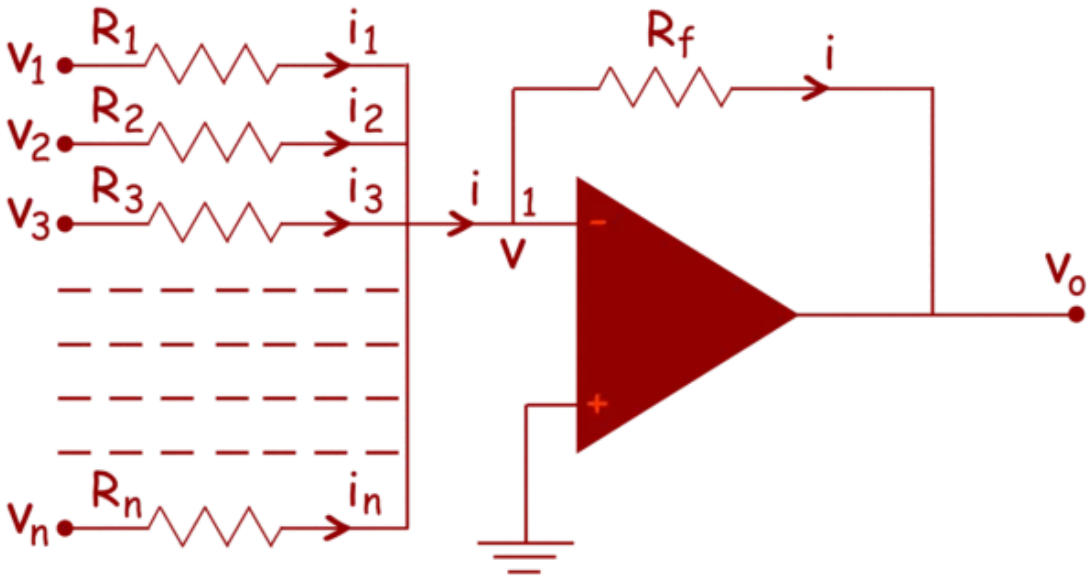


Figure 2.14: Summing Amplifier Level Shift Example [21]

Given the fundamental functionality of no voltage differential between input terminals in op amps (exhibiting stable negative feedback), the voltage into the amplifier can be determined utilizing Kirchoff's Current Law as follows. By treating the negative input node to the amplifier as V:

$$(V1 - V)/R1 + (V2 - V)/R2 + \ldots + (VN - V)/RN = 0 \qquad (2.8)$$

If each of the resistance values are identical they cancel out within the equation, leaving only:

$$V1 + V2 + \ldots + VN = NV \qquad (2.9)$$

The final voltage of the circuit can then be expressed as:

$$V = (V1 + V2 + \ldots + VN) / N \qquad (2.10)$$

This results in a simple voltage added, while simultaneously reducing the amplitude of each signal by the quantity of signals being summated. Implementing a non-inverting feedback stage can reverse the decrease in amplitude of each signal and effectively create the desired summated signal.

2.3.4: Analog to Digital Converter

Regardless of the instrumentation amplifier chosen for the design, both designs would require the connection of an analog to digital converter across the output of the filtering stage. An analog to digital converter (ADC) converts the analog output of the instrumentation amplifier into digital data, which is necessary for processing and storing the information with a data logging format. Traditional 12 lead ECG systems were hardware intensive. Minimizing both the cost and power consumption of the ADC component is crucial for this project's final application.

There are a variety of stand-alone ADCs available including but not limited to flash, folding, half-flash, pipelined, SAR, and integrating (unknown) ADCs. Furthermore, Microncontrollers exist in the current market with ADCs implemented within the system. When discussing ADCs there are three universal performance parameters to compare; sampling rate, resolution, and power dissipation. For the application within this MQP, the performance characteristics are weighed against the cost of the component. Figures 2.15 and 2.16 demonstrate the stated (as advertised) and effective (as measured) number of bits per sampling rate for each of the ADC types.

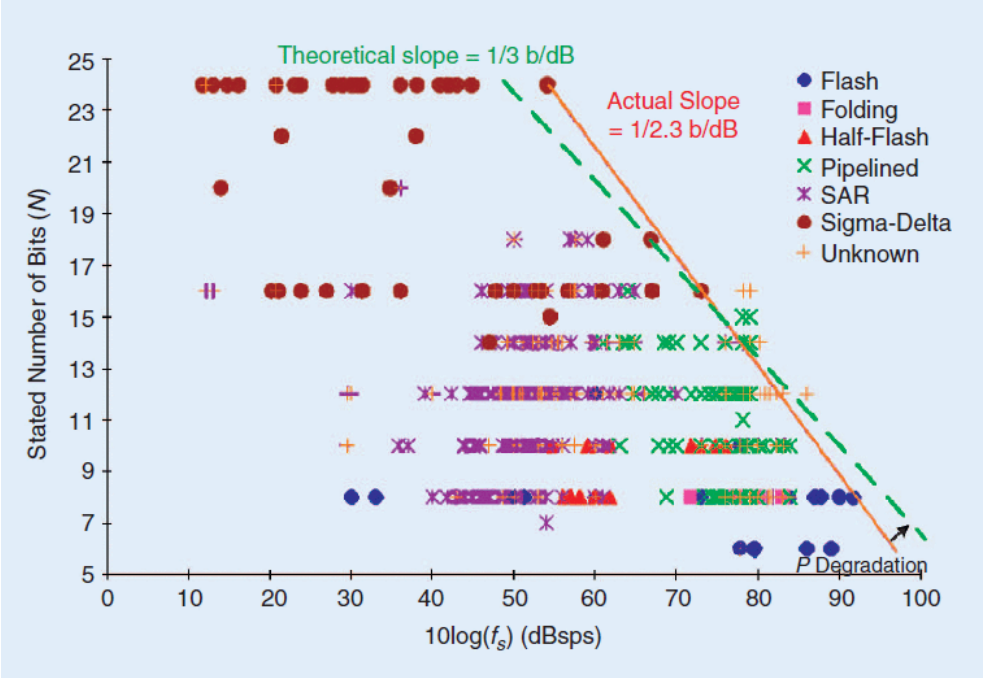


Figure 2.15: ADC Stated Number of Bits vs. Sampling Rate (sps = samples per second) [22]

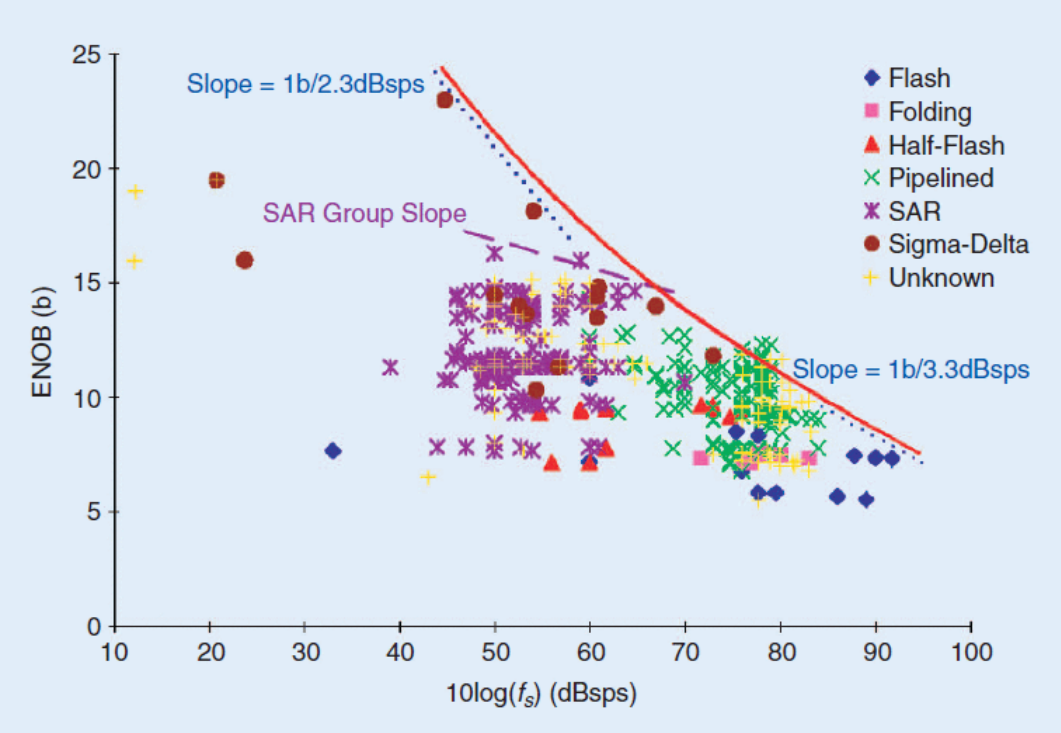


Figure 2.16: ADC Effective Number of Bits vs. Sampling Rate [22]

The pipelined and unknown structures have the best overall performance and are frequently used in military applications. SAR ADCs provide a wide range of speeds and resolutions at a relatively low cost and power dissipation rate. Sigma-Delta ADCs have the highest resolution with low sampling rates while Flash ADCs have the highest sampling rates. Figure 2.17 demonstrates the power dissipation per sampling rate for each of the ADCs.

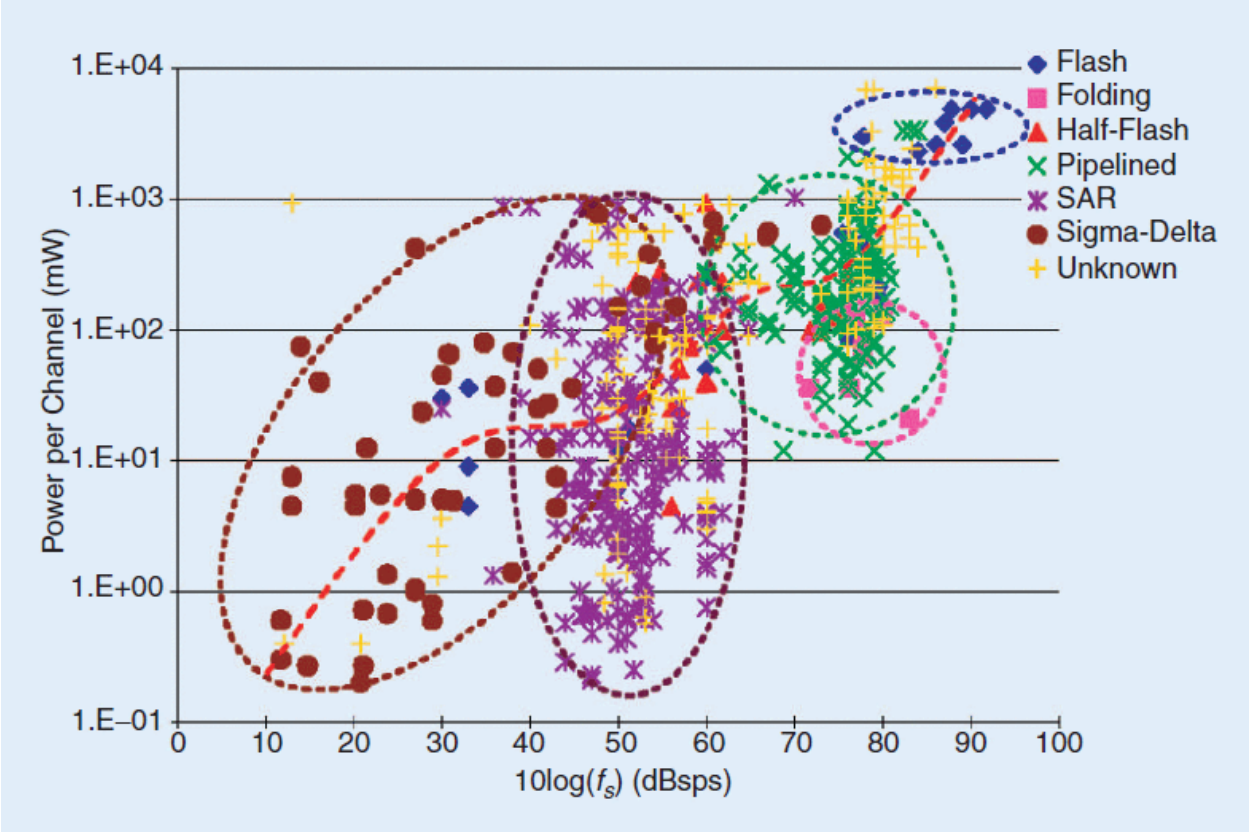


Figure 2.17: ADC Power Dissipation vs Sampling Rate [22]

After evaluating the power dissipation and bit rates of the various ADCs, two figures-of-merit can be determined. These figures-of-merit, P and F, are widely used and are modeled in their respective equations below. In these equations, B stands for the effective number of bits (ENOB), f_S is the sampling rate, and P_{diss} is the power dissipation. Figure-of-merit P evaluates the combined resolution and speed performance of the ADC while F factors in power efficiency against performance [22].

$$P = 2^B * fs \tag{2.11}$$

$$F = (2^B * fs) / P_{diss} \tag{2.12}$$

2.3.5: Additional Components

The final component necessary within the nodes of the ECG system are the wireless Microcontroller units (MCUs). Both Texas Instruments and Nordic Semiconductor offer wireless

Microcontrollers, among others. TI and Nordic MCUs offer similar performance characteristics/prices and include ADCs within the board. The wireless MCUs are necessary to transfer the signal data via Bluetooth to our data logging device (a phone or laptop). Notable MCU choices are the TI CC2642R and the Nordic nRF52840. Table 2-1 offers relevant features for both Microcontroller options.

Table 2-1: MCU Features

Feature	TI CC2642R	Nordic nRF52840
Price	$1-2 MCU alone Observed development kits ranging from $40-50	$3-6 MCU alone Observed development kits ranging from $40-70
Processor	Arm Cortex-M4F	Arm Cortex-M4F
Clock Speed	48 MHz	64 MHz
Memory	352 KB flash + 256 KB ROM + 8 KB SRAM + 80 KB ultra-low leakage SRAM	1 MB flash + 256 KB RAM
ADC	12 bit, 200 KS/s, 8 channels	12 bit, 200 KS/s, 8 channels
Receiver Sensitivity	-105 dBm	-103 dBm

The ADS1291 ECG system on a chip (SoC) presents an entirely different alternative to the previous analog designs discussed. The ADS1291, manufactured by Texas Instruments, features a 24 bit delta-sigma ADC along with built-in ECG analog front end with programmable gain control, internal reference, and an on-board oscillator. Furthermore, the SoC requires low power (335 μW/ channel) and can function properly with a unipolar or bipolar power supply.

Regardless of whether the AD620, AD624, or ADS1291 is utilized for the peripheral nodes, the intent is to power the unit with a bipolar source. In other words, two batteries will be the power source for each of the nodes attached to the individual's body. This, while occupying more physical space, will assist in making our low power circuit design.

2.4: ECG Information Analysis

After a raw signal is collected, it is important to analyze the signal to find any anomalies. Often, the raw signal is noisy and needs to be filtered to make the signal cleaner and then use the information provided by that signal to determine measurements such as heart rate. This can be accomplished by running the signal through algorithms that will, for example, detect the R-waves and determine data such as the time interval between each heartbeat and subsequently the heart rate. An algorithm for atrial fibrillation will detect any irregular beats in the signal through calculations and using the R-waves that are detected.

2.4.1: Heart Rate

One simple use for ECG is to calculate heart rate. Heart rate is usually expressed in units of beats per minute. If the time between successive R-waves is known, the heart rate can be found. The heart rate for a healthy adult should be around 60 to 100 beats per minute [23]. One method of finding heart rate is to find the average of the instantaneous heart rate within a certain period of time. The instantaneous heart rate is calculated based on the time between successive beats. Another method is to manually count the number of beats in a minute, or count the number of beats in 15 seconds and multiply that by 4 to convert to the beats per minute. [24]

2.4.2: Pan and Tompkins R-wave Detection Algorithm

There have been various R-wave detection algorithms designed in the past years. The Pan-Tompkins R-wave detection algorithm, also known as QRS detection algorithm, was originally designed in assembly language to operate on a Z80 or NSC800 microprocessor. It utilizes a dual threshold technique, which means that thresholds are determined from the filtered signal at two different filtering stages. The only disadvantage with this technique is that it works well mostly for regular heart rates and arrhythmias, but not so well with abnormalities such as bigeminy or trigeminy, since the time interval would change. Pan-Tompkins tested their algorithm using data from the MIT/BIH Arrhythmia database. The algorithm is designed so that the ECG signal goes through an analog filter that bandlimits the signal to 50 Hz. This is the same

as the dataset used in the Pan-Tompkins tests, and it utilizes an analog-digital converter (ADC) to sample the signal at 200 samples/s. [25]

2.4.2.1: Algorithm Design Overview

The algorithm designed by Pan and Tompkins passes a raw ECG signal through three digital filters: the first is a linear integer coefficient bandpass filter which is composed of cascaded lowpass and highpass filters to reject noise. The second filter approximates a derivative, and the third filter squares the amplitude and then sends the signal through a moving-window averager. After filtering, the decision portion of the algorithm is initiated, and is divided into three distinct phases: two learning phases and detection.

The first learning phase requires two seconds of data to initialize and set threshold values for peaks from the ECG signals and noise. The second learning phase utilizes two full heartbeats to initialize the RR-interval average and RR-interval limit values. The detection phase creates two sets of thresholds, one set of signal and noise thresholds after the bandpass filter, and the other set of signal and noise thresholds after the moving average. They are used to make the algorithm as accurate as possible and to detect any missed beats, as the SNR improves after bandpass filtering. For every QRS complex that is detected, there is a 200 ms refractory period, which matches the physiological function of the heart [25].

2.4.2.1.1: Filter 1- Bandpass Filter

The first filter, a bandpass filter, is used to reduce any muscle noise and outside frequency interference. However, to ensure the desired passband frequency of 5-15 Hz that is used to maximize the QRS energy, Pan-Tompkins cascaded lowpass and highpass filters. In doing so, they achieved a 3 dB passband frequency of 5-11 Hz, which is close to their design requirements [25].

A lowpass filter is used with the following system function H(z) and amplitude response |H(ωT)||, where T = sampling period:

$$H(z) = \frac{(1-z^{-32})}{(1-z^{-1})} \qquad (2.13)$$

$$|H(\omega T)| = \frac{sin^2(3\omega T)}{sin^2(\frac{\omega T}{2})} \quad (2.14)$$

This results in the following difference equation that has a lowpass cutoff frequency of 11 Hz (based on a 200 Hz sampling frequency), a gain of 36, and has a filter processing delay of six samples:

$$y[n] - 2y[n-1] + y[n-2] = x[n] - 2x[n-6] + x[n-12] \quad (2.15)$$

A highpass filter is used after the lowpass filter. It has the following system function H(z) and amplitude response |H(ωT)||, T = sampling period:

$$H(z) = \frac{(\frac{-1}{32} + z^{-16} - z^{-17} + \frac{1}{32}z^{-32})}{(1-z^{-1})} \quad (2.16)$$

$$|H(\omega T)| = \frac{[256+sin^2(16\omega T)]^{\frac{1}{2}}}{cos^2(\frac{\omega T}{2})} \quad (2.17)$$

This results in the following difference equation with a low cutoff frequency at 5 Hz, a gain of 32, and has a delay of 16 samples:

$$y[n] - y[n-1] = \frac{-x[n]}{32} + x[n-16] - x[n-17] + \frac{x[n-32]}{32} \quad (2.18)$$

With the removal of all the noise and extra points from the raw signal, this filtered signal is used for learning phase one to determine the thresholds [25].

2.4.2.1.2: Filter 2 - Derivative

To obtain the QRS-complex slope, the signal is differentiated using a five-point derivative with the following system function H(z) and amplitude response |H(ωt)||, T = sampling period:

$$H(z) = 0.1T(-2z^{-2} - z^{-1} + z + 2z^2) \quad (2.19)$$

$$|H(\omega T)| = \frac{1}{4}T[sin(2\omega T) + 2sin(\omega T)] \quad (2.20)$$

This results in the following difference equation with essentially a linear frequency response between DC and 30 Hz:

$$y[n] = \frac{1}{8}[2x[n] + x[n-1] - x[n-3] - 2x[n-4]] \quad (2.21)$$

The derivative filter mostly adjusts the slope to make the signal more visible but it does not have much of an impact on the overall algorithm performance [25].

2.4.2.1.3: Filter 3 - Squaring Function and Moving-Window Averager

Once the derivative signal is created, every single point is squared to make the data points non-negative and to provide non-linear amplification. The equation for this is:

$$y[n] = x[n]^2 \qquad (2.22)$$

It then goes into moving-window averaging to merge the Q, R, and S waves into one large wave that can be detected as the largest peak by the algorithm. The number of samples, denoted by N in the equation, is important in the displaying of the waveform. We want the width of the window to hold the widest QRS complex, because if it is too narrow then there will be extra peaks in the averaged waveform and make it a challenge for QRS detection, and if it's too wide then the averaged waveform would merge the QRS and T waves. The equation to calculate the moving-window integration is:

$$y[n] = (\frac{1}{N})[x[n - (N - 1)] + x[n - (N - 2)] + \ldots + x[n] \qquad (2.23)$$

With this signal now filtered and modified to be more visible, the R-wave can soon be detected once parameters are set [25].

2.4.2.1.4: Setting Thresholds & R-Wave Detection

To determine the thresholds, a fiducial mark, which is the location of a potential R-wave, is first set to determine when to expect an R-wave. Since the time duration of the rising edge corresponds to how wide a single QRS complex is expected to be, the fiducial mark can be determined either from where the maximum slope of one QRS complex is located, or from the peak of the R-wave.

At the start of signal analysis, the first and second learning phases are initialized. The first learning phase requires two seconds of data to set threshold values for peaks from the ECG after the moving average and the bandpass filter for the signals ($SigPk_{est}$, $SigPkFilt_{est}$) and noise ($NoisePk_{est}$, $NoisePkFilt_{est}$). These values are the running estimates of the noise and signal thresholds that are automatically updated after each peak is detected and classified as a signal peak or a noise peak. The second learning phase utilizes two full heartbeats to initialize

the RR-interval average ($RRAvg$) and RR-interval limit values ($RR\ Low\ Limit,$ $RR\ High\ Limit,\ RR\ Missed\ Limit$).

Next, for the detection phase, there are four thresholds created in total: a signal threshold after the ECG signal is filtered through the digital bandpass filter ($SigPkFilt$), a noise threshold after the ECG signal is filtered through the digital bandpass filter ($NoisePkFilt$), a signal threshold after the ECG signal goes through the moving average ($SigPk_{final}$), and a noise threshold after the ECG signal goes through the moving average ($NoisePk_{final}$). The higher threshold for both sets, generally the signal threshold, is used for the initial analysis of the signal. The lower threshold for both sets, usually the noise threshold, is used only in searchback and only if there is no peak caught by the higher threshold in the expected time interval (which was determined by the fiducial mark). Searchback is a technique that looks back once the whole signal is analyzed to detect any peaks that may have been missed the first time around. Equations 2.24-2.27 calculate what the estimated signal peak and noise peak thresholds would be for the final signal after all filtering:

$$SigPk_{est} = 0.125SigPk_{final} + 0.875SigPk_{est} \quad (2.24)$$

$$NoisePk_{est} = 0.125NoisePk_{final} + 0.875NoisePk_{est} \quad (2.25)$$

$$Thresh1 = NoisePk_{est} + 0.25(Sigpk_{est} - NoisePk_{est}) \quad (2.26)$$

$$Thresh2 = 0.5 * Thresh1 \quad (2.27)$$

Thresholds are applied to the bandpass filtered signal in the same way as above in Equations 2.28-2.31:

$$SigPkFilt_{est} = 0.125SigPkFilt + 0.875SigPkFilt_{est} \quad (2.28)$$

$$NoisePkFilt_{est} = 0.125NoisePkFilt + 0.875NoisePkFilt_{est} \quad (2.29)$$

$$ThreshFilt1 = NoisePkFilt_{est} + 0.25(SigpkFilt_{est} - NoisePkFilt_{est}) \quad (2.30)$$

$$ThreshFilt2 = 0.5 * ThreshFilt1 \quad (2.31)$$

During searchback, the signal peak is recalculated since the threshold is at a lower amplitude. Equations 2.32 and 2.33 present the new calculation:

$$SigPk_{est} = 0.25SigPk_{final} + 0.75SigPk_{est} \quad (2.32)$$

$$SigPkFilt_{est} = 0.25SigPkFilt + 0.75SigPkFilt_{est} \quad (2.33)$$

Additionally, if irregular heart rate is present, then both thresholds are reduced by 50% to accommodate lower amplitudes and make the algorithm more sensitive.

After this, the average RR-interval and rate limits are determined. There are normally two RR-intervals that are maintained: the first being the average of the eight most-recent beats ($RRAvg2$), and the other based on the average eight most-recent beats that fall within certain limits ($RRAvg2$). These limits are located at the threshold or very close to it:

$$RR\ Low\ Limit\ =\ 92\%\ *\ RRAvg2 \tag{2.34}$$

$$RR\ High\ Limit\ =\ 116\%\ *\ RRAvg2 \tag{2.35}$$

$$RR\ Missed\ Limit\ =\ 166\%\ *\ RRAvg2 \tag{2.36}$$

Finally the T-waves are identified when an RR interval is less than 360 ms. This identification is based on the maximal slope of the waveform, and if a maximum slope of a peak is determined to be less than half of the maximum slope of the preceding peak, then it is a T-wave. Otherwise, it is a QRS-complex. This can be tested using a database that has ECG signals; specifically, Pan and Thompkins used the MIT/BIH arrhythmia database to test their algorithm. [25]

2.4.3: Heart Rate Variability

Heart rate variability (HRV) is the measure of the time variation between instantaneous heartbeats and RR intervals. A healthy heart should vary in heart rate, and these variations are caused by changes, such as exercise, emotions, temperature, etc. Since there are variations in the heartbeats, the R-wave time locations that are detected are generally non-periodic; to make the sampling of the heart rate periodic, the time series is resampled by interpolation so that the new periodic series can then be analyzed using power spectrum analysis. These time differences are normally on the order of hundreds of milliseconds [26].

HRV is important because it can be used to find anomalies in patients. An example of this is atrial fibrillation, where HRV is used to calculate the root mean square of successive differences (RMSSD), which is the difference between the size of adjacent RR-intervals. In the RMSSD formula below, “a” represents RR intervals in a sequence of length *l* and a(j) represents the RR-interval in “a”. [27]

$$RMSSD = \sqrt{\frac{1}{(l-1)} \sum_{j=1}^{l-1} (a(j+1) - a(j))^2} \quad (2.37)$$

There are algorithms that have been designed to calculate HRV, using time domain, statistical, and geometric methods. Time domain uses the detection of the QRS complex to determine the instantaneous heart rate or cycle length. The intervals between QRS-intervals are called normal-to-normal (N-N) intervals. Statistical methods are great for long ECG traces, such as ones that are 24+ hours long. HRV can then be determined in one of two ways: either from the direct measurements of the instantaneous heart rate or derived from the differences between normal RR intervals. Using the standard deviation of the occurrences of normal RR intervals, variability can be determined (among other measures). RMSSD is a measure that is derived from these interval differences. Geometric methods create a sample density distribution using normal RR intervals. This method needs longer traces such as a minimum of a 20 minute trace, but 24 hours is preferred [26].

2.4.4: Atrial Fibrillation

Atrial fibrillation (AF) is a heart condition where the atria of the heart contracts irregularly [28]. Kannel et. al. [29] found that the incidence of AF increases every decade after 50 years of age and by the age of 80 about 10% of people are likely to develop this condition [29]. Men are 50% more likely to develop AF than women. The nonuniform electrical impulses occurring in the atria cause irregularly timed heart beats because of the unpredictability of impulses across the atrioventricular node [28]. The atrioventricular node sends signals to the ventricles telling them to contract, and when the node sends abnormal signals from the atria, the ventricles will contract irregularly [28].

Some possible causes of atrial fibrillation include surgical procedures, lung disease, hypertension, and coronary artery disease [28]. Those diagnosed with atrial fibrillation often report experiencing fatigue, dyspnea, dizziness, chest pain, and palpitations [28]. Atrial fibrillation increases the risk of thromboembolism and stroke which is why it is important to diagnose it as early as possible [28].

Electrocardiograms are used to help diagnose atrial fibrillation [28]. One aspect of the ECG wave that is associated with atrial fibrillation is the absence of a P wave [28]. This property

of atrial fibrillation is hard to detect with computer algorithms, so most algorithms use the irregularity of R-wave timings as the primary indicator. These two characteristics can be seen in Figure 2.18.

Figure 2.18: Bottom arrow points to a normal ECG P wave. Top arrow points to an ECG with atrial fibrillation

2.4.4.1: Atrial Fibrillation Detection Algorithms

We describe three methods in the literature that outline how to detect atrial fibrillation, with only one indicating that it could run in real-time.

2.4.4.1.1: Algorithm 1: Tateno and Glass, 2001

Tateno and Glass [30] used the coefficient of variation of RR intervals and probability density histograms. The coefficient of variation (CV) represents the relative variability in a data set [23, Ch. 4, pp. 169]. It is calculated by dividing the standard deviation by the mean and multiplying by 100% [23, Ch. 4, pp. 169].

$$CV = \frac{\sigma}{\mu} * 100\% \tag{2.38}$$

$$\sigma = \sqrt{\frac{\Sigma(x_i - \mu)^2}{N}} \tag{2.39}$$

In the above equations μ is the mean, N is the number of RR intervals, and x_i is each RR interval value in the data set. The higher the CV, the greater variability is present [23, Ch. 4, pp.169]. For atrial fibrillation, because of its inherent variability, the coefficient of variation is a good indicator of its presence in an ECG signal.

From the MIT/BIH ECG atrial fibrillation database, standard density histograms were created by categorizing beats into bins based on their RR and ΔRR intervals, where the ΔRR interval is the difference between successive RR intervals. After the histograms are constructed, the standard CV of the RR intervals and the standard CV of ΔRR intervals are calculated. When calculating the CV of the ΔRR interval, ΔRR is divided by the RR interval mean. This is done because the ΔRR mean from the histogram is approximately zero. Next, both the CV of RR and ΔRR intervals were calculated from a test data set. If the test coefficients of variation are within ±35% of the standard coefficients of variation, calculated from the histogram data, it is labeled as atrial fibrillation [30].

The data from the test recording is placed into a density histogram, and with the Kolmogorov-Smirnov test, compared to the standard density histogram created from the atrial fibrillation database. The Kolmgorov-Smirnov test is used to measure the difference between two distributions [30]. The equation for this test is

$$p = Q(\lambda) = 2\sum_{j=1}^{\infty}(-1)^{j-1} * e^{-2j^2\lambda^2} \qquad (2.40)$$

where

$$\lambda = \frac{N_1 N_2}{(N_1+N_2)}, \qquad (2.41)$$

where N_1 is the number of points in the standard distribution and N_2 is the number of points in the test distribution. If the value of *p* is greater than parameter P_c=0.01 it is an indication of atrial fibrillation. The sensitivity and specificity of the Tateno-Glass model were 94.4% and 97.2%, respectively [30].

2.4.4.1.2: Algorithm 2: Dash et al., 2009

The model of atrial fibrillation detection developed by Dash, Chon, Lu, and Reader [27] finds the turning points ratio, the root mean square of successive RR differences, and Shannon entropy of an ECG in order classify a heart rhythm as being in atrial fibrillation. This model was published in 2009 and can run in real time [27].

The turning point ratio (TPR) is a nonparametric statistical test used to determine if the data set is random. For atrial fibrillation detection it is used to determine if the time between RR

intervals is random by comparing them to their neighbors. The expected turning point ratio of an independent random set with Gaussian distribution is

$$\frac{2l-4}{3} \pm \sqrt{\frac{16l-29}{90}}, \tag{2.42}$$

where *l* is the length of the series. The TPR is calculated using the following equations,

$$\overline{p} = \frac{2l-4}{3}, \tag{2.43}$$

$$var(\overline{p}) = \frac{16l-29}{90}, \text{ and} \tag{2.44}$$

$$z = \frac{|p-\overline{p}|}{\sqrt{var(\overline{p})}} \tag{2.45}$$

where $\overline{p}$ is the number of turning points in a random set, $var(\overline{p})$ is the variance of the random set, p is the number of turning points in the set of RR intervals, and z is the TPR. If the series has a turning point ratio outside of the 95% confidence interval, it is considered to be in regular sinus rhythm [27].

The root mean square of successive RR difference (RMSSD) is an atrial fibrillation feature because of its high variability. It helps compensate for outliers that can cause false detection of atrial fibrillation. The RMSSD is used to find the value for RmsTHresh, one of the classification parameters. RmsThresh is the ratio of the RMSSD and the mean RR interval [27].

Shannon entropy measures the chance that patterns exhibiting regularity over some duration of data will also exhibit similar patterns over the next duration. White Gaussian noise has a Shannon entropy, SE, of 1, which is the maximum value because it is entirely random, whereas a sinusoidal wave has a SE of 0. For this application, the first segment of RR intervals is used to create a histogram. The eight highest and lowest values are removed and considered outliers. The remaining values are sorted into equally spaced bins. Once the values are sorted, the probability distribution is calculated for the i^{th} bin by dividing the number of beats in the bin by the total beats in the segment, i.e.:

$$p(i) = \frac{N_{bin(i)}}{l-N_{outliers}} \tag{2.46}$$

where N_{bin} is the number of beats in the bin, l is the length of the segment, and $N_{outliers}$ is 16. Once all p(i) are found, SE can be calculated using the following equation,

$$SE = -\sum_{i=1}^{16} p(i)\frac{log(p(i))}{log(1/16)} \text{ [27]} \quad (2.47)$$

For the Dash et al. method, the presence of ectopic beats tends to result in the false classification of atrial fibrillation. In order to fix this issue, they propose filtering these beats out before applying the classification algorithms. The Dash et al. method has a sensitivity of 94.4% and a specificity of 95.1% [27].

2.4.4.1.3: Algorithm 3: Lee et al., 2013

The method of atrial fibrillation that had the highest sensitivity and specificity is that of Lee, McManus, and Chon [1]. It uses time varying coherence functions and Shannon entropy to classify ECG signals as having atrial fibrillation. The time-varying coherence function (TVCF) can be used to detect atrial fibrillation because the values for the function will significantly decrease if all or part of the segment contains atrial fibrillation [1].

Before the TVCF and Shannon entropy can be calculated ectopic beats are removed to reduce detection error. The RR interval ratio is calculated using the equation,

$$RR(i)/RR(i-1) \quad (2.48)$$

where RR is the RR interval vector and i is the index value. The ratio is calculated for every value in the vector. The 1st, 25th, and 99th percentile were calculated for the ratio data. The conditions for a beat to be considered ectopic are

$$RR(i)/RR(i-1) < percentile\ 1, \quad (2.49)$$

$$RR(i+1)/RR(i) > percentile\ 99, \quad (2.50)$$

$$\text{and } RR(i+1)/RR(i+2) > percentile\ 25 \quad (2.51)$$

This method assumes that the surrounding beats are normal and therefore will not remove all the ectopic beats present but will remove enough to reduce overall error.

The TVCF can be found by calculating the nonparametric time-frequency spectra of the RR interval time series. The equation for the time-frequency spectra is:

$$|\gamma(t,f)|^4 = \frac{|S_{xy}(t,f)|^2}{S_{xx}(t,f)S_{yy}(t,f)} \frac{|S_{yx}(t,f)|^2}{S_{yy}(t,f)S_{xx}(t,f)} \quad (2.52)$$

where $S_{xy}(t,f)$ and $S_{yx}(t,f)$ are the time-frequency cross spectrum of two signals, x and y. $S_{xx}(t,f)$ and $S_{yy}(t,f)$ represent the auto spectra of the two signals. The result of the above equation is a vector in the frequency domain. This equation is applied to two adjacent segments consisting of RR interval data from an ECG. The first segment is signal x and the next segment is signal y. AF is indicated when the magnitude of the TVCF in the frequency domain decreases below a threshold as the frequency increases [1].

Once the TVCF is found the Shannon entropy is calculated to aid in accurate detection of atrial fibrillation. If the Shannon entropy value is above a threshold value of 0.79 and the TVCF frequency variation threshold value of 0.019, then the beats indicate atrial fibrillation. This method has a sensitivity of 97.41% and a specificity of 97.54% [32]**.** In Table 2-2, direct comparison of the three detection algorithms can be seen.

Table 2-2: Comparison of different atrial fibrillation detection algorithms

Comparison of Atrial Fibrillation Detection Algorithms			
Methods	MIT/BIH atrial Fibrillation database		MIT/BIH Normal Sinus Rhythm database
	Sensitivity (%)	Specificity(%)	Specificity (%)
Lee et al [32]	97.41	97.54	100
Dash et al [27]	94.4	95.1	99.7
Tateno and Glass [30]	94.4	97.2	N/A

2.5: Standards & Specifications Used

- To prevent any possible risk of shock to the person, we will be using high value resistors to keep the current at the worst case scenario of the full 4 V to the person with a current of under 10μV. This is deemed a safe level by ANSI/AAMI ES1—1993
- IEC 60601-2-25:2011 applies to the basic safety and essential performance of electrocardiographs intended by themselves or as a part of a medical electrical system, for the production of electrocardiogram reports for diagnostic purposes. [33]
- IEEE 802.15.1-2002 This IEEE Standards product is part of the 802 family on LAN/MAN. Abstract: The lower transport layers [(Logical Link Control and Adaptation Protocol (L2CAP), Link Manager Protocol (LMP), baseband, and radio] of the Bluetooth™ wireless technology are defined. Bluetooth is an industry specification for short-range radio frequency (RF)-based connectivity for portable personal devices. [34]
- ANSI/AAMI EC38:1998, the American National Standard for Ambulatory ECGs, and ANSI/AAMI EC57:1998, the American National Standard for Testing and Reporting Performance Results of Cardiac Rhythm and ST Segment Measurement Algorithms: These are the standards that are used to create beat-by-beat (bxb) comparison reports.

3: Project Strategy

In this section, we will share our deliverables, how we completed a portion of this project, and future directions we plan to take with the remainder of the project.

3.1: Original Client Statement

In this project, you will design a wireless, wearable ECG system composed of multiple independent peripheral nodes and one central node. Each peripheral node will record one ECG lead with all electrodes in close proximity, such that no wires need to extend from the node (and, of course, there will also be no wires between the multiple nodes). All of the peripheral nodes will simultaneously transmit to a common central node. An ability to record multiple ECG leads simultaneously and unobtrusively will advance the information available from these wearable devices—just as most clinical ECG recordings are collected from multiple wired leads simultaneously.

3.2: Objectives and Constraints

The end goal of this project is the prototype production of a small, battery-powered, three electrode wireless ECG system that logs data in real time and processes the signal afterwards. Specifically, the signal processing portion will include detecting R-waves and atrial fibrillation. The system will consist of a central node to house most of the electronics in addition to peripheral ECG nodes placed around the body. These lead nodes will wirelessly communicate with the central node so that data can be collected and analyzed. The data will then be sent to a computer for the signal processing (the central node can be wired to a computer). Our system will preferably be small enough that it can be concealed on a person's body and capable of monitoring continuously.

Our project's ultimate goal was to test our circuit on human subjects with healthy sinus rhythms and atrial fibrillation. An IRB application was submitted and tests were performed consensually with a subject. Clinical bench testing was performed on the circuit to ensure that it was safe and make our Microcontrollers compatible with our circuit.

3.3: Project Approach

Figure 3.1 shows the overall flow of how our system is intended to function.

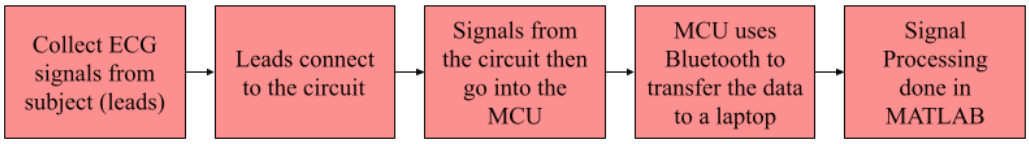


Figure 3.1: High Level Blackbox Diagram of System

We split the project into three major deliverables: hardware, R-wave detection algorithm, and atrial fibrillation detection algorithm. The hardware portion consisted of building an ECG circuit, programming the Nordic Microcontroller units (MCU), and testing all components individually as well as when combined to produce satisfactory ECG signals. Additionally, the battery and ECG electrodes need to be packaged well. The R-wave detection algorithm portion consisted of writing the detection algorithm in MATLAB, testing on the MIT/BIH Arrhythmia and Atrial Fibrillation databases, and optimizing the algorithm to produce satisfactory results. The atrial fibrillation detection algorithm portion consisted of writing the detection algorithm in MATLAB, testing on the MIT/BIH Atrial Fibrillation database, and optimizing the algorithm to produce satisfactory results. Our Gantt Charts are in Appendix A and they show our timeline for accomplishing all of our objectives.

3.3.1 Tools

To accomplish this project, we used various software and hardware tools to create, modify, and test all the different portions of our design. To build our ECG circuit, we used NI Multisim (National Instruments), an online circuit simulator and schematic tool, to design and test all of the different circuits we were considering. This also gave us an idea on how the circuit might operate prior to us purchasing any hardware components. Once we finalized our circuit design, we were able to transport the schematic into NI Ultiboard, which is a Printed Circuit Board (PCB) layout program. Ultiboard allowed us to place components where we wanted them, route our signals, and then verified our design.

To write code, we used MATLAB and Segger. We used MATLAB to design our R-wave detection and atrial fibrillation algorithms, and used the MIT/BIH Arrhythmia and Atrial Fibrillation databases on Physionet to test both algorithms. PhysioNet has the ECG recordings that have been reviewed by cardiologists who have indicated the time locations of R-waves and different types of arrhythmias. This information is used to determine the accuracy of our algorithms.

Segger is the IDE that is compatible with the Nordic Microcontroller units (MCU) that we used for this project, and it was programmed in C/C++. Once the device was constructed and programmed, the data it collected was analyzed and conclusions were formed. Data was sent wirelessly from the system to MATLAB for processing, and the results were displayed for the user.

Tools used throughout the hardware design included a digital multimeter, an oscilloscope, and a series of basic components. Those components included jumper wires, resistors, capacitors, and a protoboard. Throughout the design process, prototypes were constructed on the protoboard using the basic components along with a collection of instrumentation amplifiers and operational amplifiers utilized by previous MQP applications (namely the AD620 instrumentation amplifier and LM348 operational amplifier). As prototypes were constructed, the digital multimeter and oscilloscope verified proper voltage, current, resistance, and signal shape. Finally, as the protoboard design became mature, a solder board and soldering iron were utilized to create the final effective prototype of the circuit.

3.4: Revised Client Statement

In this project, you will design a wireless, wearable ECG system composed of one independent peripheral node. The node will record one ECG lead with all electrodes in close proximity, such that no wires need to extend from the node. The node will transmit to a common central node. An ability to record multiple ECG leads simultaneously and unobtrusively will advance the information available from these wearable devices—just as most clinical ECG recordings are collected from multiple wired leads simultaneously. In addition to the recording and transmission of the ECG signal, processing of the signal will occur to detect R-waves and atrial fibrillation.

4: Design Options

The construction of a wireless ECG signal reader necessitates analog front end signal conditioning along with Bluetooth communication via a Microcontroller to deliver data to the receiver (laptop) for heart rate analysis. Figure 4.1 describes the topology of that wireless system.

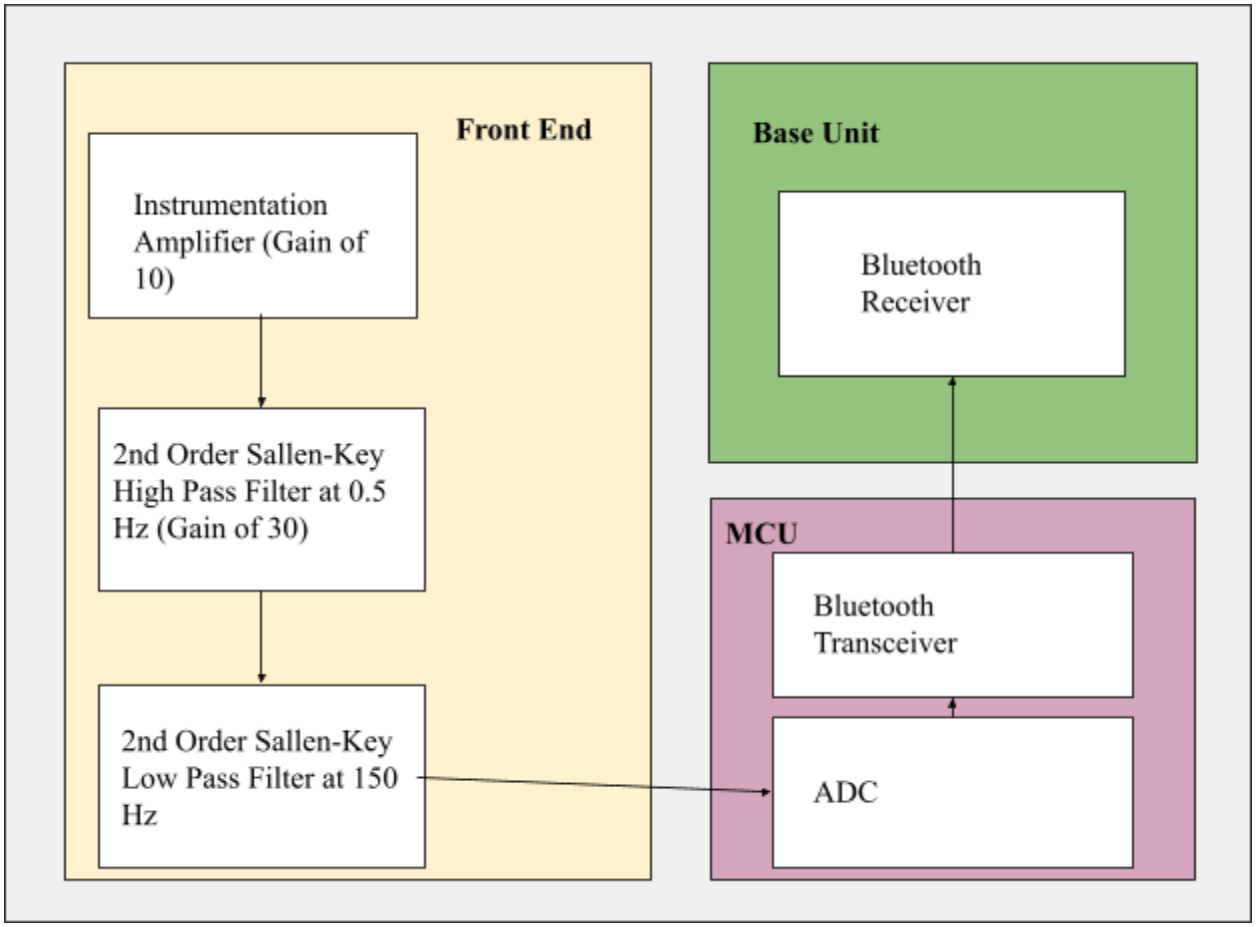


Figure 4.1: Block Diagram of Wireless ECG Topology

The steps outlined in the topology consist of an instrumentation amplifier stage, two second order Sallen-Key filters, selectable gain, an ADC, and Bluetooth transceivers/receivers. The following sections will discuss the design options for each component necessary for the wireless ECG circuit along with design options for heart rate analysis algorithms.

4.1: Instrumentation Amplifier

For the purposes of this project an instrumentation amplifier needed to be used to amplify the signal coming from the electrodes on the body. The three amplifiers that were looked at were the INA118, the LN1167, and the AD 624. All of these components work for our applications and have different benefits but in our final design only one was used.

The first one that was looked at was the AD624, a 16 pin DIP (Dual in Line Package) that met our requirements for our early prototyping design. The AD624 requires a minimum supply voltage of ± 6 V. One of the major benefits of this IC is that it has the gain already built into the package. If a gain of 100, 200, 500, or 1000 is needed, the corresponding pins need to be shorted. One of the drawbacks of this system, however, is that it only comes in a DIP form and it is fairly hard to get a hold of due to it being somewhat outdated. In the scenario where the circuit needs to be shrunk to a smaller overall package this IC would not allow the freedom of using a SOIC (small outline integrated circuit) package. The primary issue is that this IC needs more voltage than our current system is able to supply. [35, p. 624]

The next instrumentation amplifier we considered was the AD8227, an eight pin IC that is available in both SOIC and DIP. This IC also has a high enough maximum supply voltage of ± 18 V as well as a minimum input voltage of ± 1.5 V which works far better for a lower power system using ± 3 V. The issue with this IC is its availability, it is unrealistic that we would be able to have it delivered in time. The AD8227 also has a price tag of between $3 to $10 for the SOIC and DIP systems. [36]

The last of the instrumentation amplifiers looked at was the LT1167, another eight pin IC that is available in both SOIC and DIP. This amplifier has a slightly higher maximum supply voltage of ± 20 V and a minimum of ±2.3 V which also works for the intended supply voltage of ± 3 V. The LT1167 has plenty of availability as of this writing and should pose no issue in obtaining the components needed. A benefit that the LT1167 has over the other two amplifiers is that it is slightly cheaper with prices ranging from $8 to $15 on the SOIC and DIP systems. [37]

For the sake of this project the instrumentation amplifier that will be used is the LT1167 due to its availability, its price, and its specifications meeting all the needed requirements. It is likely that any of these different amplifiers would work for our application but for this project the LT1167 will be used. For more information see Table 4-1.

Table 4-1: Comparison of Instrumentation Amplifiers

Amp	Min Vin	Input-Referred Voltage Noise	Maximum Single Resistor Gain
AD624	±6 V	10 nV/√Hz	1000
AD8227	±1.5 V	24 nV/√Hz	1000
LT1167	±2.3 V	7.5 nV/√Hz	10000

4.2: Operational Amplifier

For this project there is a need to use multiple operational amplifiers in the highpass and lowpass filters. There are a lot of choices when it comes to op-amps and the one that the group ended up selecting was the LM324. This rail-to-rail amplifier is a 16 pin DIP that has four amplifiers in the package which is enoughfor us to only need one chip. The LM324 also has a minimum supply voltage that is low enough for our intended use of ± 3 V. One of the major benefits of this op-amp is that it can be bought in either a SOIC or DIP version and each of the styles is less than a dollar per package. Lastly the main benefit of this amplifier is the availability of the ICs as there were some available to us now as well as many available online [38].

4.3: Analog to Digital Converter

The design choices for the ADC has the most flexibility, and also the widest range in price variation among all the components being selected. The available options we considered were the ADS1291 (discussed earlier in Section 1.3.4), a TI sigma-delta ADC, a USB-1608G series ADC module from Measurement Computing, or utilization of the on-board ADC of the Microcontrollers available to us. The ADS1291, designed specifically for ECG circuitry, is a low power device with one input channel, low noise programmable gain amplifiers, two high resolution amplifiers, and a built in oscillator. A sigma delta ADC uses a sigma modulator and digital filtering. Filtering of an ADC involves oversampling of the signal, noise shaping, digital filtering, and decimation. The USB-1608G series can connect to a PC through the USB port and has 16 bit resolution. Furthermore, the USB-1608G also has a maximum sampling rate of 250

kS/s. Finally, the on-board ADC of the Microcontrollers has 12-bit resolution and a maximum sampling rate of 200 kS/s.

There is also a significant distinction in the price of each of the ADCs. The ADS1291 is roughly $7 per unit, making it extremely cost effective for our design. It will, however, require a degree of software installation and configuration to the Microcontroller and PC. The USB-1608G ranges between $400 and $800 USD. While the PC based models are significantly more expensive, they are being considered based on their capabilities for configuration with PCs and MATLAB.

4.4: Voltage Regulator

In order to use the summer amplifier to level shift the voltage within the range of the ADC, 1 V must be added to the signal. As the voltage from the battery decreases over time, a voltage regulator is required to maintain a constant voltage at the input of the summing amplifier. One of the components considered was the TPS73615MDBVREP, a voltage regulator made by Texas Instruments. It had an input voltage range of 1.7-5.5 V, which is ideal for an expected input of 3 V. It also has a 400 μA operating supply current and within 0.5% voltage output accuracy. The TPS73615MDBVREP is a surface mount component, which is not as convenient for protoboarding and soldering as a through-hole component [39].

The LP38841T-1.5/NOPB is a through-hole voltage regulator similar to the TPS73615MDBVREP but with slightly different performance characteristics. The LP38841T-1.5/NOPB also has a constant output voltage of 1 V and has an input voltage range of 1.575-5.5 V. Its operating supply current is 32 mA and its output voltage accuracy is within 1.5% [40].

With its better accuracy and lower supply current, the surface mount TPS73615MDBVREP was chosen over the through hole LP38841T-1.5/NOPB. In addition to the voltage regulators, surface mount to through-hole adapter sockets were used to allow for the convenience of a through-hole component.

4.5: Microcontroller

Design options for the MCU were narrowed down to the TI CC2642R and the Nordic nRF52840. As discussed in Section 1.3.4: Additional Components, the two Microcontrollers have similar performance characteristics. Both models have identical processors and 12-bit ADC sampling at 200 kS/s. Furthermore, the Nordic model has more memory and faster clock speeds than the CC2642R. Finally, the difference in receiver sensitivity between the two MCUs is negligible; the nRF52840 having a sensitivity of -103 dBm and the CC2642R having a sensitivity of -105 dBm. The Nordic MCU is more expensive, but only in the range of a few dollars, so that factor is not important for the decision making process. A primary advantage in favor of the Nordic MCU, outside of performance characteristics, is its recommendation of use by graduate students assisting our project group. These recommendations come from prior MQP experience. In previous projects, the TI units produced transmission collisions. To accommodate that problem, the project groups had to insert a third inactive device to prevent the missing packets and introduced delays. Nordic models allow the packets to be more spread out, and thus, don't require the third additional device. Outside of graduate student recommendations, and factoring in their relatively similar performance capabilities, it can be generally noted that there is no substantial difference between the two MCUs for our application.

4.6: Multi-Electrode Selection

For the multi-lead portion of the device, we can build either a two or three electrode ECG peripheral node. A two-electrode device is grounded using the signal path rather than an electrode, which makes it noisier compared to having an electrode that is meant to serve as the ground. However, an advantage of it is that the hardware is not very complex so it works excellent for a wireless ECG monitor [39]. Another advantage of a two-electrode ECG is that since there are fewer electrodes, it is a much more comfortable fit on the body compared to a three-electrode system. However, we are choosing to build a three-electrode device because the signals are more accurate and less noisy, but it is at the expense of being large for a system that is wireless and is mounted on the body for a long period of time.

4.7: Algorithm Selection

One of our major deliverables was to write algorithms that will successfully process the signal and produce an ECG waveform. Additionally, the algorithms needed to detect R-waves and atrial fibrillation. There are various algorithms and methods designed by experts that can accomplish both of these, but in this section we elaborate on the specific algorithms we implemented and our reasoning for choosing it.

4.7.1: R-Wave Detection

For R-wave detection, we implemented a detector using Pan and Tompkins' QRS detection algorithm methodology. We accomplished this by using a MATLAB implementation of this algorithm found online as a reference and eliminated unnecessary steps, such as the derivative filter, which did not have much of an effect on the final output of the signal. The success of our algorithm was tested by using the MIT/BIH ECG Arrhythmia Test Database and the MIT/BIH ECG Atrial Fibrillation Database and comparing the results to Pan and Thompkins' results.

4.7.2: Atrial Fibrillation Detection

For the detection of atrial fibrillation, algorithm three was implemented. The characteristics of AF were used to identify its presence in an ECG signal. By finding the TVCF and Shannon entropy, segments of the ECG signal were tested for the presence of AF. This algorithm consists of two primary steps and is nonparametric. Algorithms one and two involve more computation, a greater number of tests, and were found in the literature to be less accurate than algorithm three as seen in Table 2-2. This is why algorithm three was chosen for this application. The success of our algorithm was tested by comparing it to the results of Lee et al.'s paper and the MIT/BIH Atrial Fibrillation database.

5: Hardware

Our final circuit design is constituted by an LT1167 instrumentation amplifier, an LM324N quad operational amplifier, the TPS73615MDBVREP voltage regulator and a series of basic components (resistors and capacitors). Each of these components was selected to satisfy a low power circuit condition (± 3 V rail-to-rail design). The selection was limited to availability of components in the market due to the electronic components shortage during the COVID-19 pandemic, along with an understanding of the trade-off between power consumption and noise. Originally, the instrumentation amplifier of choice was Analog Device's AD8227. However, the AD8227 was unavailable when we were constructing the prototypes and had a lead time of approximately one year. The LM324N and the TPS73615MDBVREP were both selected as they satisfied the low power requirement of the circuit along with the design requirements, which we will elaborate on in Section 5.1. The hardware construction process, outlined in greater detail in the following sections, began with virtual simulation in Multisim, followed by prototype construction of the circuit on a protoboard, succeeded by constructing the circuit on a solderboard, and finalized with implementing the circuit on a PCB.

5.1: Design Process

The analog front end design corresponds to our initial set of design requirements for circuit performance listed in Table 5-1:

Table 5-1: Requirements for Analog Front End Design

System	Wireless ECG
Input Signal Amplitude	± 2-4mV
Input Signal Frequency	0.05 Hz - 150 Hz
Selectable Gain	600
CMRR	≥ 80 dB at 60 Hz
Input-Referred Voltage Noise	≤ 1 μV RMS RTI for full frequency band
Size	Relatively small, within the size of two fists
Voltage	3 V
ADC	12 bit resolution
Isolation	Medically isolated
Current Consumption	≤ 20 mA @ 3 V

Our analog front end design is fairly conventional and consists of six sections: the input, the instrumentation amplifier, the highpass filter, the lowpass filter, and the level shifter (Figure 5.1).

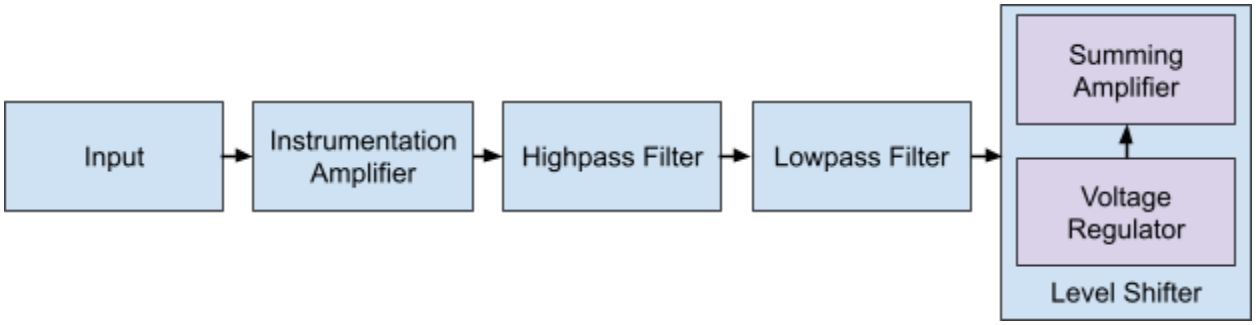


Figure 5.1: Abstract Circuit Diagram

Starting at the input in Figure 5.1 there are three electrodes that will be used, one will be tied to the circuit's ground to act as a reference for the other two. The other two will pass through an 80 kΩ resistor separately that limits the highest possible current going to the body at 50 μA. This value was found by taking the maximum possible voltage in the circuit which, if the batteries are fully charged, can be as high as ± 4 V and dividing it by the maximum permissible current of 50 μA.

After this resistor on each electrode, there are two diodes going to and from ground that protect the circuit in the unlikely scenario where the body outputs enough voltage to pass through the diodes in either direction. This works as the diodes require a minimum voltage in order to trigger which, under normal operation, the human body will not reach. If for some reason the body exceeds this voltage threshold of 400 mV, then it will be shorted to ground. In summary, the four diodes within the circuit operate as electrostatic shock protection for test subjects. After this, both electrodes go into the next stage which is the instrumentation amplifier.

For this project, we used the LT1167 instrumentation amplifier, with gain set to 10 using a 5 kΩ resistor across pins one and eight. The relatively low gain of this stage is intentionally set due to possible DC offsets introduced at the onset of the instrumentation amplifier. The instrumentation amplifier uses the full range of the two batteries with the amplifier tied to both the positive and negative rail.

The combined signal from the instrumentation amplifier is then sent through a second-order Sallen Key highpass filter with a gain of 30 and a cut-off frequency of 0.05 Hz to filter out as much of the low frequency noise as possible and increase the strength of the signal. Then the highpass filter goes into the second-order Sallen Key lowpass filter with no gain and a cutoff of 150 Hz, which also attempts to get rid of noise in the signal. Both of these filters are using the LM324 operational amplifier in the same four amplifier package. The gain of these stages was set to ensure that noise amplification is reduced. The gain of 300 brings the overall amplitude of the signal to around 1 V peak-to-peak depending on the input amplitude, which is perfect for the 3 V range of the ADC. This value does not approach the full scale range of the ADC, nor does it meet our original design requirements for the system, which will be discussed further within the discussions section of this chapter.

The last two stages go together as a level shifter in order to use the ADC on the nRF52840. The ADC is unipolar, however the signal coming out of the lowpass filter is bipolar. In order to ensure proper functionality, a 1 V DC offset needs to be added to the signal. This is done by using a summing amplifier that combines the signal from the lowpass filter with a 1 V DC coming from the voltage regulator. The voltage regulator is used so that as the voltage of the batteries varies slightly, the 1 V offset will remain constant. Once the summing amplifier combines these and adds a final stage of amplification to the original signal, it can be sent to the nRF52840 MCU and have the signal converted to digital. The overall circuit diagram can be seen

in Figure 5.2 with each major stage labeled. It is important to note that the voltage sources in Figure 5.2 represent the 2 - 4 mV signal generated from the body and are necessary for simulation in Multisim. Voltage source V5 represents the third electrode utilized in our system and is intentionally driven to an external reference point.

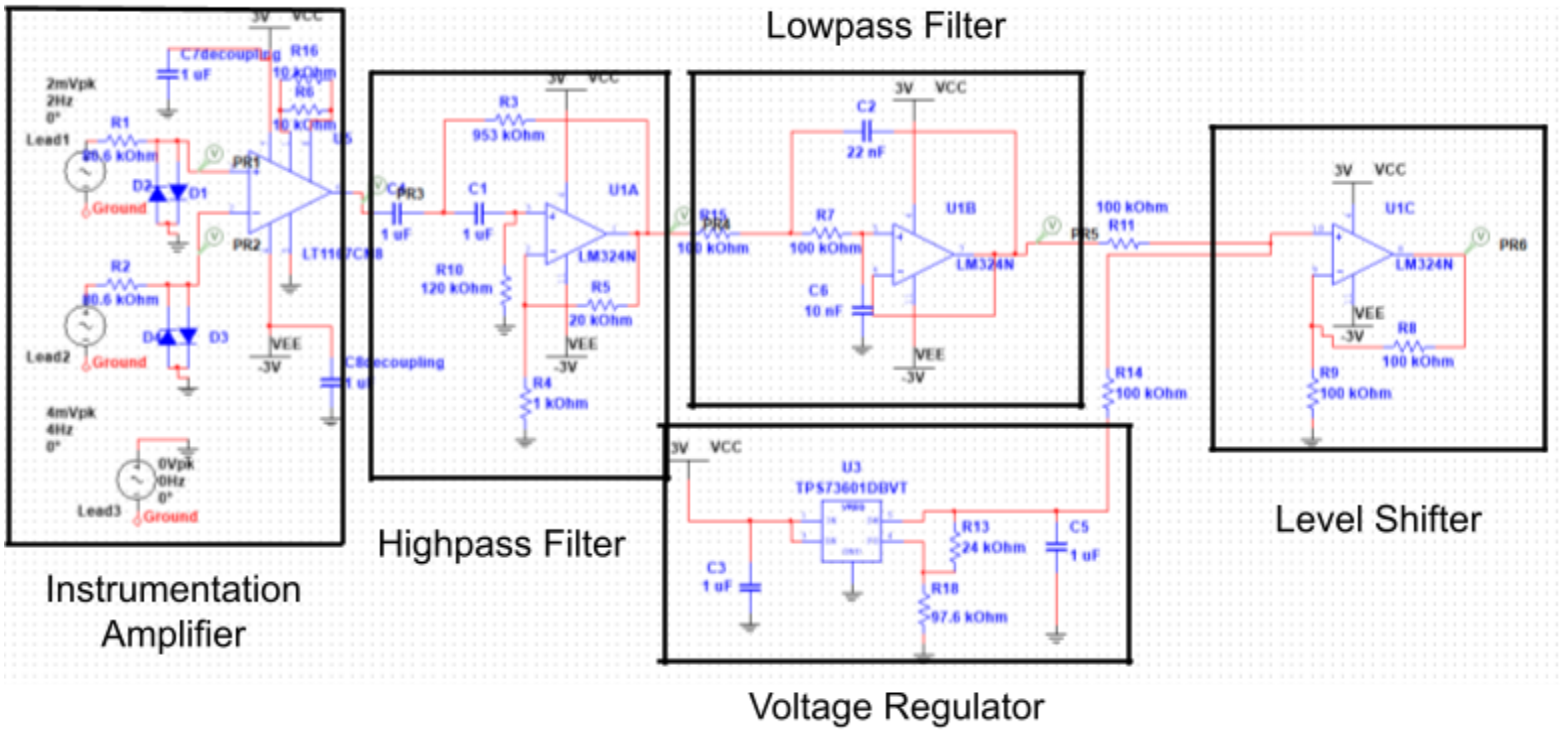


Figure 5.2: Overall Circuit Diagram

5.2: Hardware Design Verification in Multisim

With all of the components chosen and the final circuit designed, the hardware was verified through simulation in Multisim. Figure 5.3 is a plot of the output of the circuit with sine waves applied at the input. A 2 mVpk 2 Hz sine wave and a 4 mVpk 4 Hz sine wave with no phase difference were used as the two ECG inputs in the simulation shown below. If the circuit performs as expected, the difference between the input signals will be amplified, filtered, and level shifted.

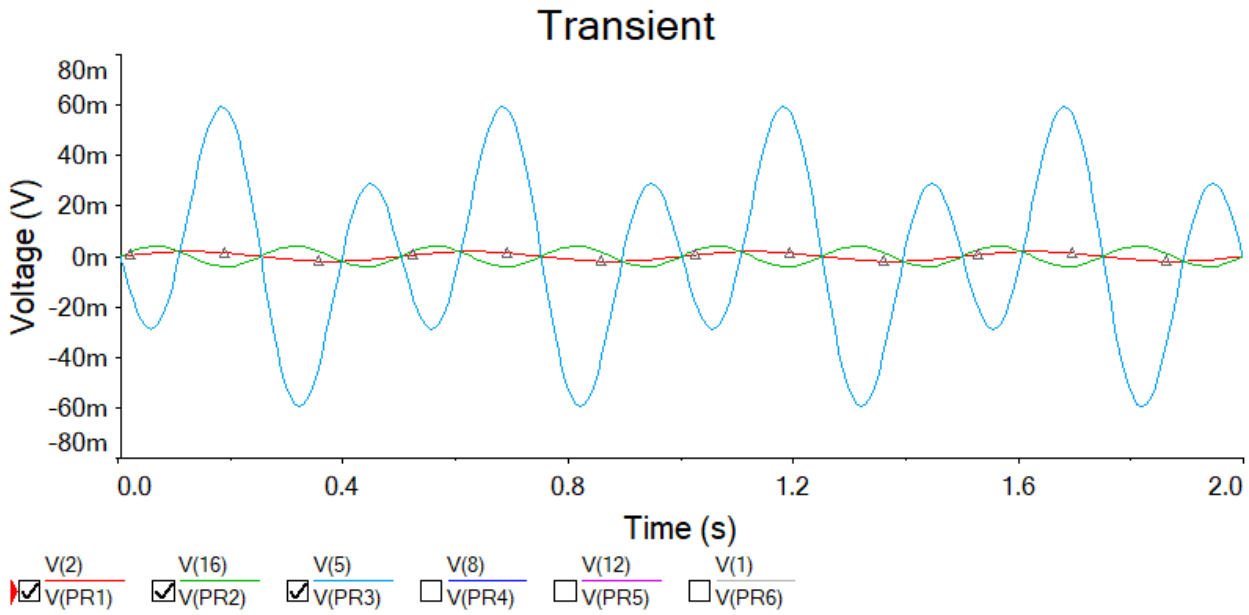


Figure 5.3: Multisim Output

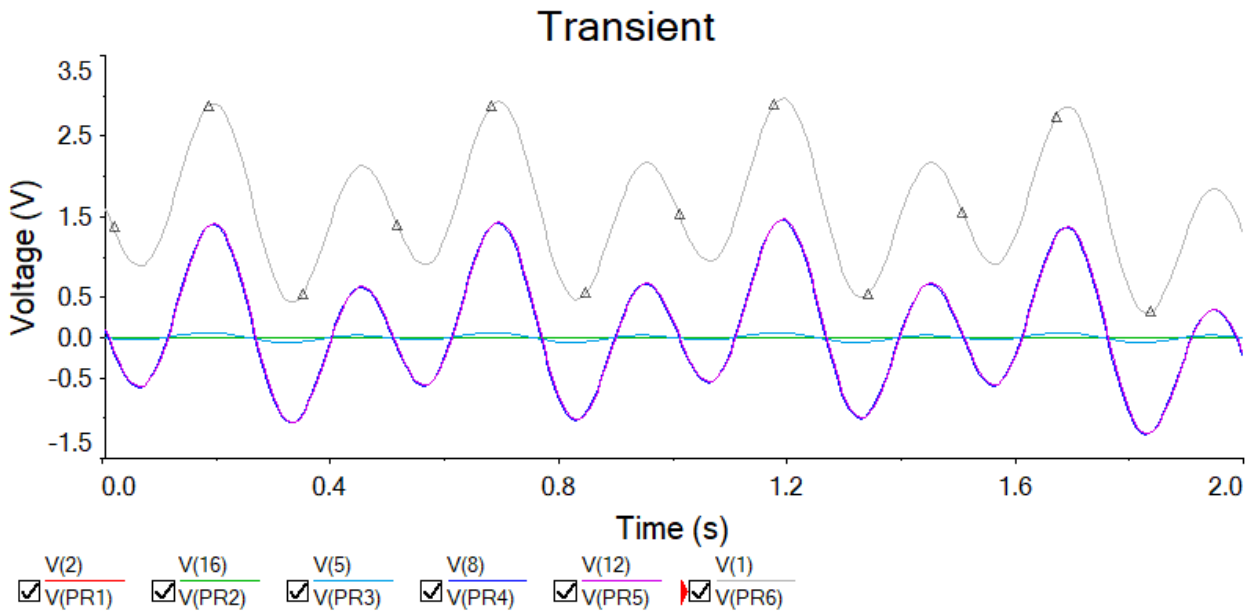


Figure 5.4: Multisim Output

The two small input waves are the red and green waves shown in Figure 5.3. The difference between them is slightly amplified at the output of the instrumentation amplifier, the blue wave on the plot. The resulting signal is then highpass and lowpass filtered with additional gain at the highpass stage. The purple wave in Figure 5.4 shows the result after highpass filtering with amplification. The lowpass filter is then applied, producing an identical wave due to unity gain and lack of noise in the simulation. Finally, the signal is level shifted to center the wave at 1.5 V and allow it to be passed into the ADC, shown as the gray wave on the plot. The output signal ranges from 0 and 3 V and is centered at 1.5 V as desired. At every stage of the circuit, the simulated design performed as expected. The resulting gray output waveform demonstrates an amplification of the difference between the red and green input waveforms along with a voltage shift upwards (as intended with our design). With the correct circuit operation verified in Multisim, the hardware section of the project could then be further developed.

5.3: Hardware Design Verification on a Protoboard

Utilizing available components, the team verified the proper functionality of the circuit on a protoboard. The chosen components for the instrumentation amplifier and operational

amplifiers are the LT1167 and the LM324N respectively. Due to significant shipping delays due to the ongoing COVID-19 pandemic, a prototype was created on a protoboard utilizing the AD620 instrumentation amplifier and the LM348 quad operational amplifier. The voltage regulator (TPS73615MDBVREP) for the summing amplifier stage was modeled with a separate 1 V DC power supply. Finally, the input voltage to the circuit utilized an ECG signal generating device, the TechPatient Cardio developed by HE Instruments. This device produces a realistic ECG signal (proper amplitude, waveform, and can model realistic artifacts within the body). With all the components listed, the circuit properly offset the input voltage by 1 V and introduced the desired gain. Switching the components used in the prototype to the components for the final design should constitute almost no changes in performance, since the LT1167 and LM324N are low power components. While there may be slightly more noise introduced due to that fact, the general performance of the circuit is expected to be identical. An image of the finalized circuit on the protoboard can be seen in Figure 5.5 with the resulting waveform in Figure 5.6.

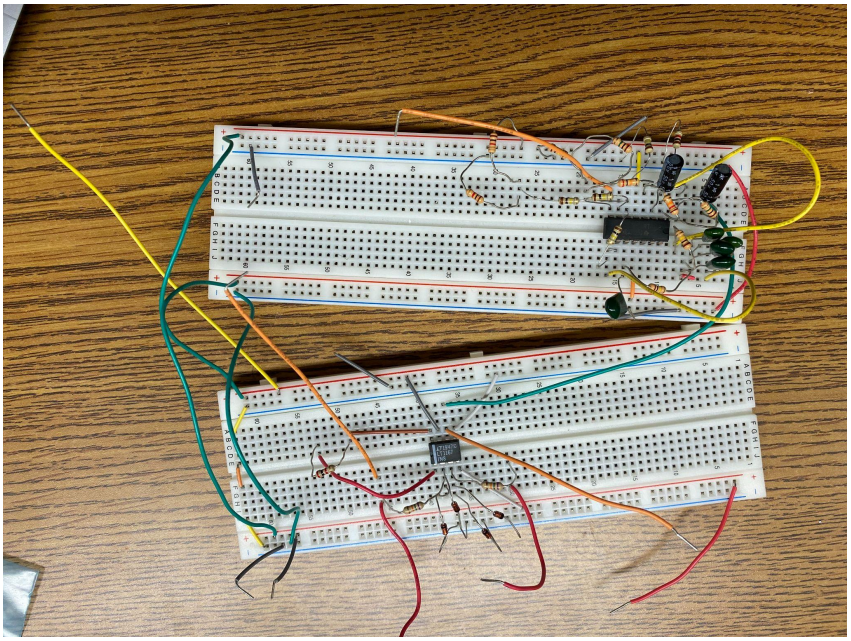


Figure 5.5: Finalized protoboard ECG circuit

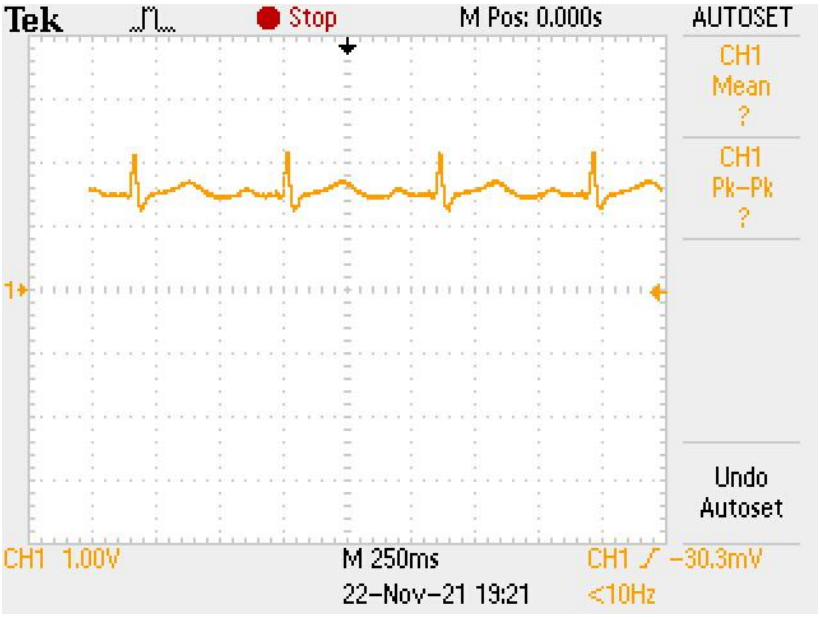


Figure 5.6: Output of ECG protoboard circuit utilizing ECG signal device

5.4: Hardware Design Verification on a Solderboard

Once the design was validated on the protoboard, that same design was replicated on a solderboard to test and attempt to gather data from it. The board was made using the same components as the previous designs. In the iteration shown in Figure 5.7, the only difference between the protoboard version and this version of the circuit is that it has been condensed and put in an easier to use format that will allow for testing to occur. You can see the final output wave of the circuit in Figure 5.8.

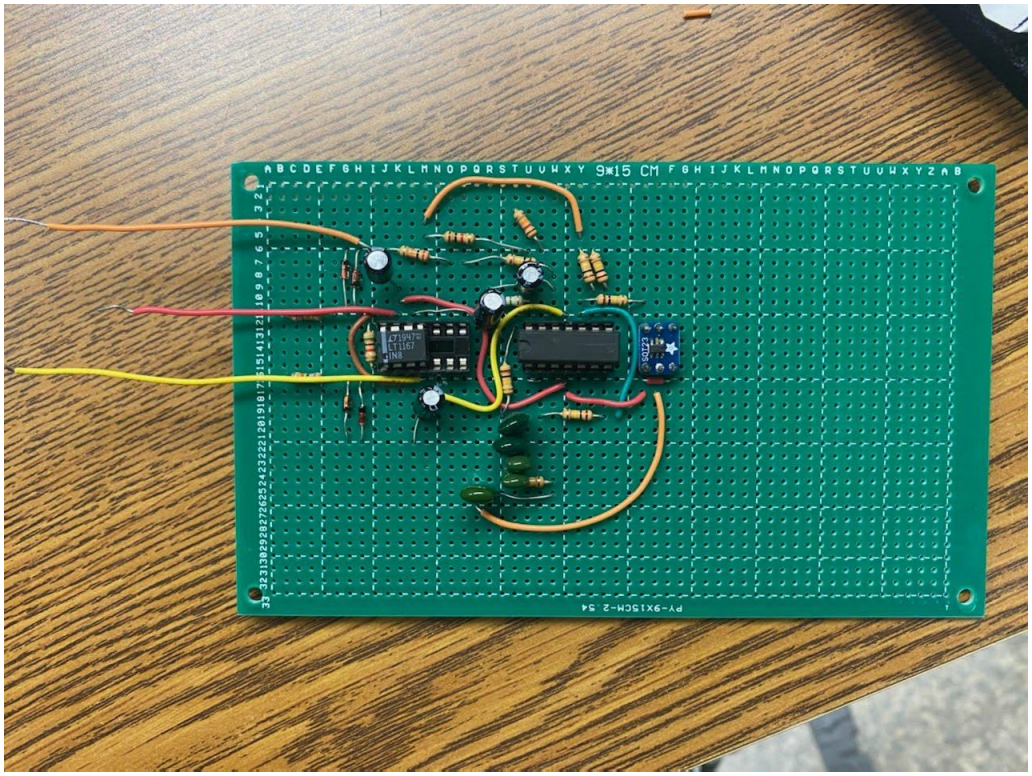


Figure 5.7: ECG Solderboard Circuit

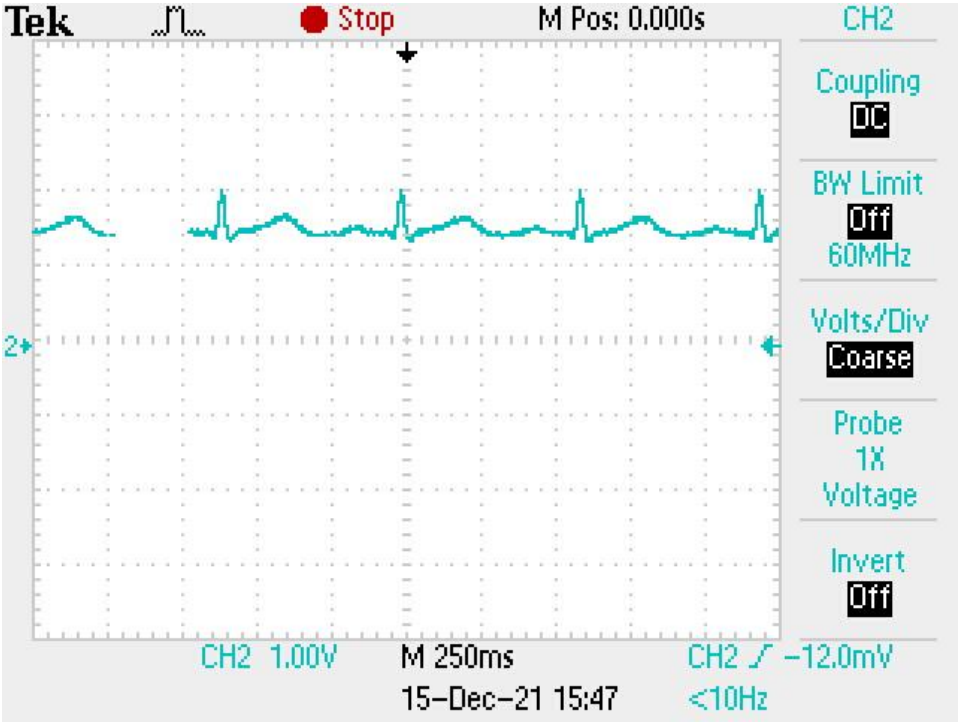


Figure 5.8: Output of ECG Solderboard Circuit Utilizing ECG Signal Device

5.5: PCB Design and Layout

Once the circuit was finalized in Multisim, we transported our design onto NI Ultiboard, where we prepared the layout for the PCB. This process was completed in parallel to the construction of the protoboard and solderboard prototypes. Our PCB is built using the components in the circuit diagram (Figure 5.2), specifically, a LT1167 instrumentation amplifier, an LM324N quad operational amplifier, a TPS73615MDBVREP voltage regulator, resistors of various values that help us achieve a gain of 300, and capacitors of various values that serve as decoupling capacitors to maintain voltage supply levels. Figure 5.9 shows our final PCB layout.

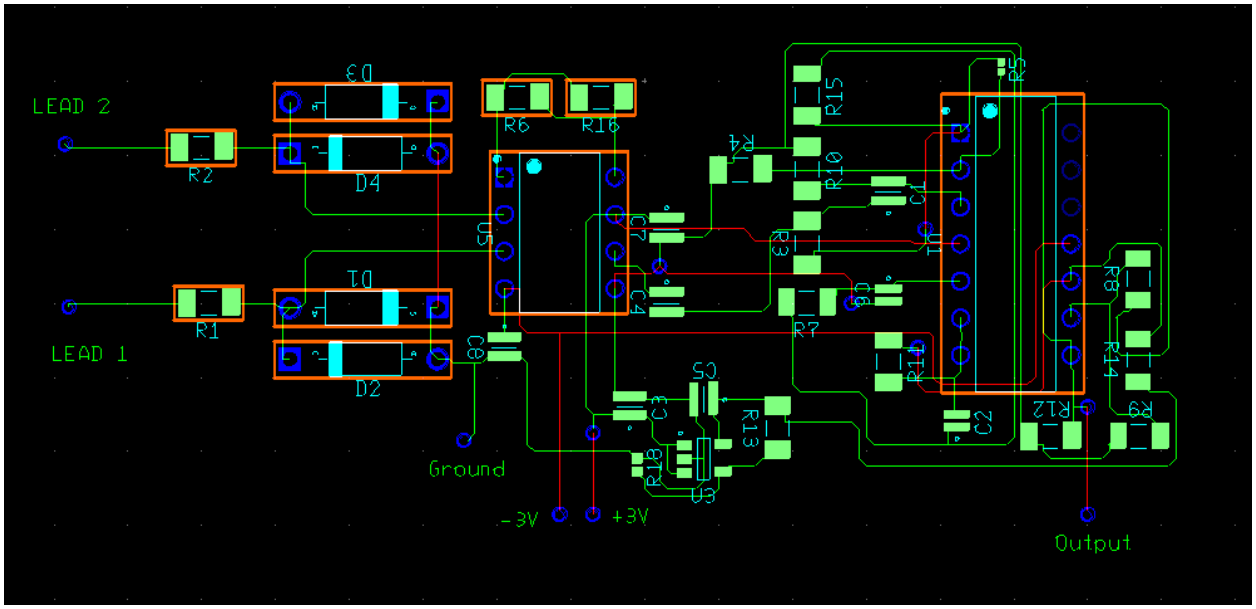


Figure 5.9: Final PCB Layout

Our PCB board was two-layers: one copper top layer and one copper bottom layer. As part of the layout process, we ensured that signals did not cross each other and all decoupling capacitors were placed next to the component that they were connected to. All of our components are through-hole components except our resistors, capacitors, and voltage regulator which are surface-mounted. Since we have a lot of resistors, we chose to make them surface-mounted since they are smaller than through-hole components, which will make our overall PCB smaller. Additionally, all of our grounds are connected to each other and that common ground will be connected to the center tap of the battery. We also have input signals for

two electrodes, which are close to the grounded electrode, and our electrodes can be manually soldered on. Finally, we have our VCC and VEE, which are +3 V and -3 V respectively, also with external ports to be connected to the battery. We left some extra board space next to the connections where two batteries, for +3 V and another battery for -3 V, can be mounted. Figure 5.10 shows what our expected final PCB board looks like.

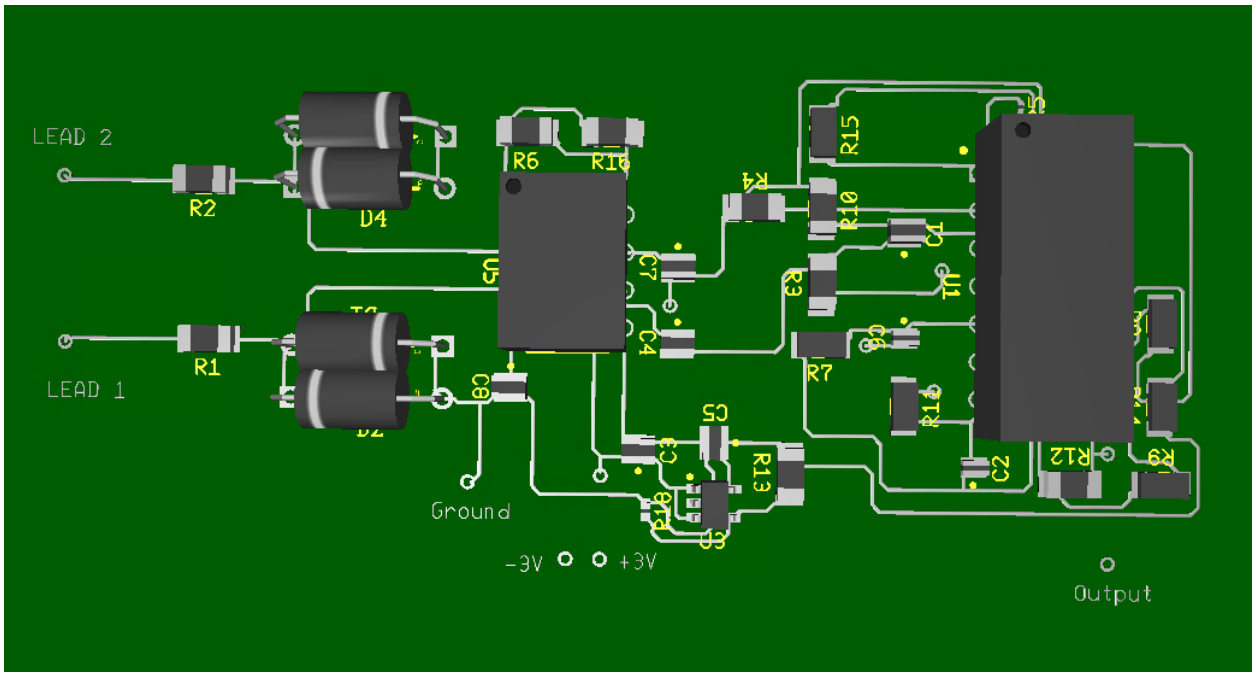


Figure 5.10: Expected Final PCB

Once the PCB design was completed the board was ordered in addition to the surface mount components to be soldered. The board was populated according to the design to create a smaller and more complete version of the circuit on the solderboard. The completed PCB with all of the components soldered is shown in Figure 5.11 below.

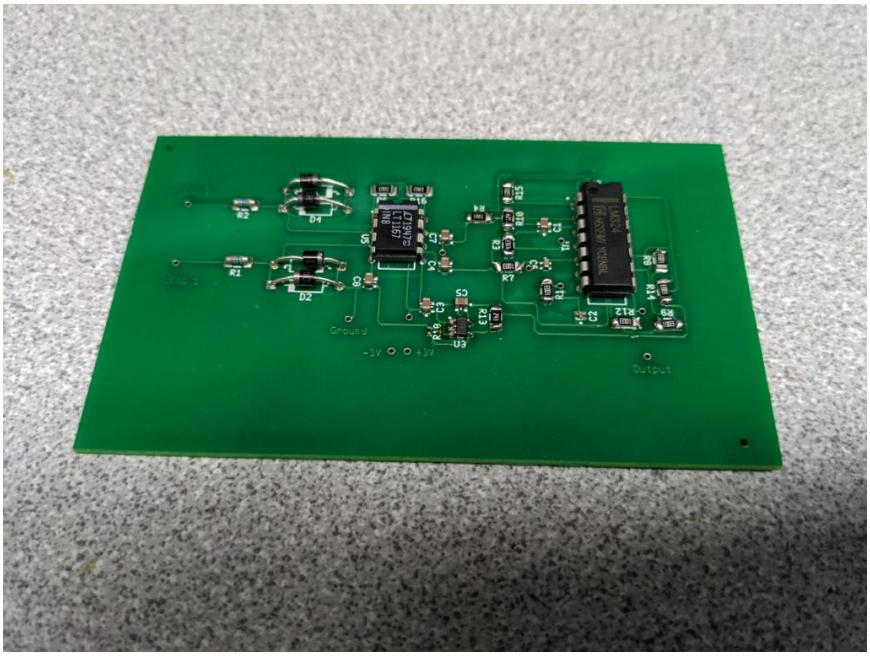


Figure 5.11: Assembled PCB

5.6: Hardware Housing

The housing for the analog design was 3D modeled and printed within WPI's Innovation Studio. The housing, in the shape of a box, consists of two separate pieces: the main body and the lid. The two pieces conjoined have a base area of approximately 30 square inches (6"x5"). The height of the two pieces conjoined is roughly 1.5". The lid sits on top of an extended ridge from the base piece of the housing. Furthermore, a small hole exists on both sides of the box for the wiring of the ECG leads to escape the housing. A detailed image of the 3D modeling can be seen below in Figure 5.12.

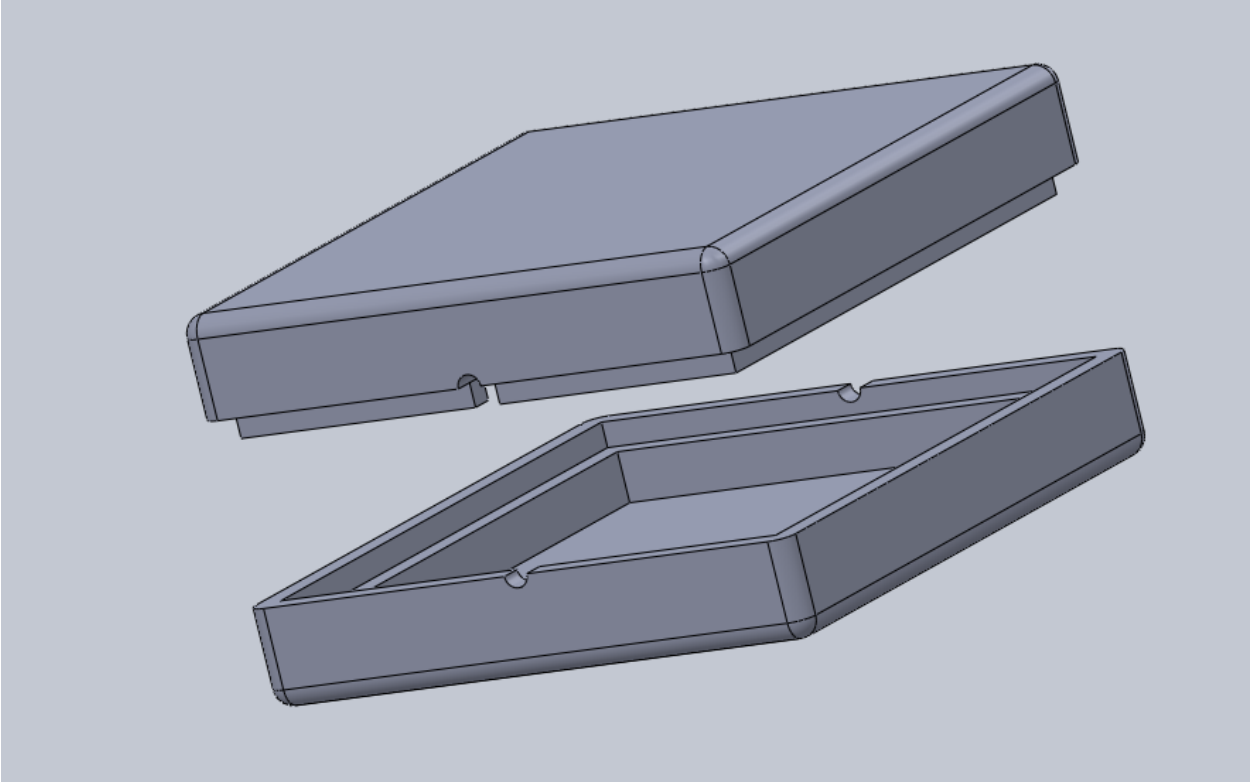


Figure 5.12: Housing 3D Model, Lid (bottom) and Main Unit (top)

Images of the hardware sitting within the open box as well as an image of the printed box enclosing the hardware components can be seen in Figures 5.13 and 5.14 respectively.

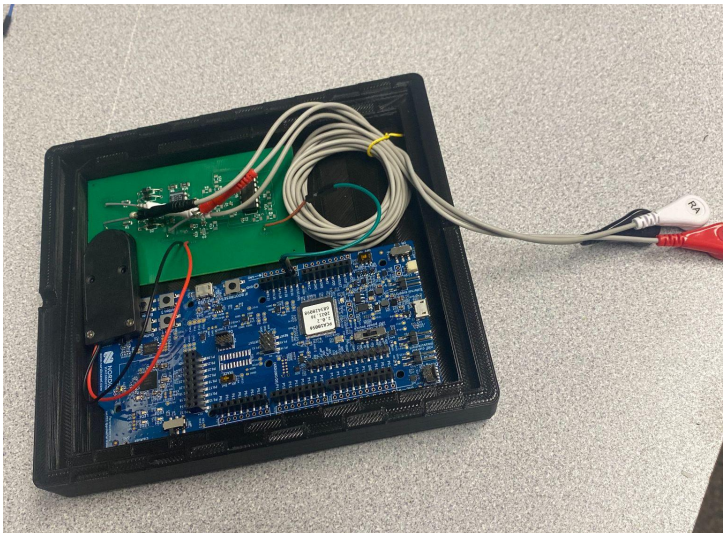


Figure 5.13: Hardware Components within Housing

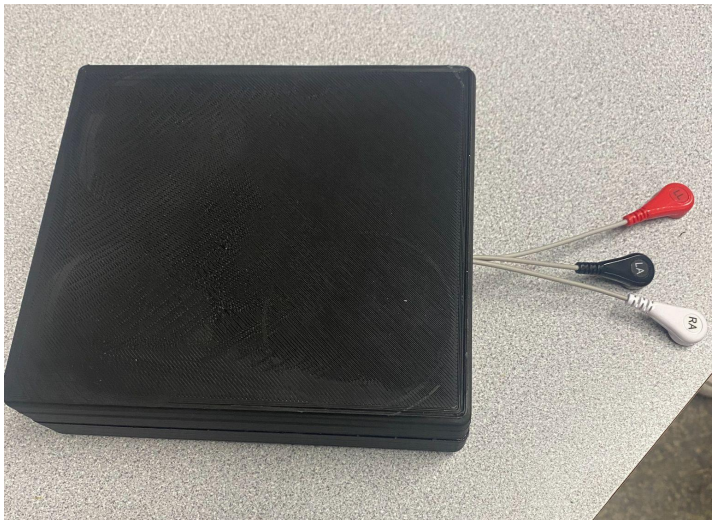


Figure 5.14: Housing Enclosing Hardware Components

5.7: Discussion

Given the analog front end design requirements discussed in section 5.1, a circuit was designed to generate an ECG waveform. The hardware consisted of instrumentation amplifier, highpass filter, lowpass filter, and level shifting stages. Components were chosen for each stage and the circuit was designed in Multisim. With a complete Multisim design, the circuit could be simulated to ensure functionality. Figures 5.3 and 5.4 are the result of a Multisim simulation showing the expected behavior with sine waves applied as inputs. The components were then ordered and the circuit constructed on a breadboard. Figure 5.5 is a picture of the breadboarded circuit and Figure 5.6 is the plot of the output waveform with an ECG simulator device used as input.

Once the operation of the circuit was verified in hardware on a breadboard, the components were soldered for a more finalized device. Figure 5.7 is a picture of the soldered circuit and Figure 5.8 is the output using the ECG simulator. A PCB was designed and ordered according to the designs shown in Figures 5.9 and 5.10. Surface mount components were ordered to populate the PCB and the finished board is shown in Figure 5.11. In addition, housing was designed and 3D printed to contain the electronics. The completed hardware was then connected to the Microcontroller to send the generated ECG signal for processing.

Shown in the table below is a repeat of the intended design requirements table along with the measured performance of the circuit.

Table 5-2: Initial Design Requirements and Measured Performance

System	**Wireless ECG Requirement**	**Measured**
Input Amplitude	± 2 - 4 mV	Satisfies requirement
Input Frequency	0.05 - 150 Hz	Lowpass cutoff of 150 Hz and highpass cutoff of 0.05 Hz
Selectable Gain	600 for Full Scale Range of ADC	300 (half of Full Scale Range)
CMRR	≥ 80 dB @ 60 Hz	≈ 81 dB @ 60 Hz
Input-Referred Voltage Noise	≤ 1 μV RMS RTI full band	≈ 0.8 μV RMS RTI
Size	Approximately the size of two fists	30 in² (5 in x 6 in)
Voltage	3V maximum	No component requires over 3V
ADC	12-bit resolution	12-bit resolution onboard Microcontroller
Isolation	Medically isolated	Medically isolated
Current Consumption	≤ 20 mA @ 3 V	2.1 mA @ 3V

CMRR was measured by taking the comparison of common mode and differential gain of the amplifier. Common mode gain was measured by connecting both inputs of the amplifier to a sine wave generator and measuring the noise at the output. Differential gain was measured by grounding one input of the amplifier and the other input to a sine wave generator, with noise measured at the output. CMRR takes the ratio of common mode to differential gain. Input-referred voltage noise was measured by shorting both inputs of the amplifier to ground and measuring the noise at the output.

Our circuit satisfied each of the initial design requirements outside of the intended gain of 600 for the full scale range of the ADC. While a gain of 600 would utilize the full range of the

ADC, making data collection and future algorithm processing easier, it produced clipping on the output waveform of our circuit. We ultimately decided that the tradeoff of less gain and thus a smaller utilized range of the ADC was acceptable for this application.

Future improvements to the hardware aspect of this project would first include further decreasing the overall size of the device. A small and lightweight device is ideal for a convenient product to be used by a patient. In addition, some of the existing components are powered from a bipolar supply. As a result, two 3V batteries are required to power the system. Using a unipolar design would allow the entire system to only be powered from a single 3V battery.

6: Embedded

The embedded software utilized for this project was developed prior to the onset of this project by graduate students He Wang and Jianan Li. The following sections of the report explain the software environment for the embedded code, installation directions for use on personal laptops or computers, an explanation of bluetooth transmission and the debug process, and data visualization of the transmitted data.

6.1: Details and Environment

To access and operate the code the first step is registering for a Github account. Github is a free code hosting website that provides version control capability within a software environment. The entirety of the embedded code files, including two other MATLAB files for data conversion and visualization, are shared via a privatized GITHUB repository. For the moment, access to the repository is limited by invitation from the graduate students He and Jianan.

Once a Github account has been created and access to the repository has been granted, the other important environmental setup is installing Segger Embedded Studio. Embedded Studio, created by Segger Microcontroller Systems, is a free, cross-platform integrated development environment (IDE). When installing the IDE, it is crucial to select the J-Link/J-Trace option for connection with external devices (the Nordic Microcontrollers). Segger Embedded Studio is a C-based coding environment and is functionally similar to other IDE's utilized within the WPI curriculum such as Code Composer Studio.

6.2: Personal Setup

Unfortunately, trying to run the code within the Segger Embedded Studio environment directly after installation produces a slew of errors. In order to properly run the software a series of adjustments need to be made to the environment. The first requirement is to download a software development kit (SDK) from the Nordic website. The SDK can be located at the following website,

https://devzone.nordicsemi.com/guides/short-range-guides/b/getting-started/posts/introduction-to-nordic-nrf5-sdk-and-softdevice, and should be downloaded by clicking on the specified hyperlink.

Figure 6.1: SDK Download Hyperlink

The development kit is necessary for replacing unrecognized files in the future, along with the added bonus of familiarizing yourself with the basics of the bluetooth transmission and reception based code. Once the SDK is downloaded the first adjustment to the IDE can be made. After attempting to run and debug the code, errors should appear stating that the .c files cannot be located or do not exist. Despite appearing to exist on the left side of the IDE's user interface, Segger does not recognize the pathing for the .c file and therefore it must be rerouted. For each unrecognized .c file, you must delete that file within the interface and drag in the duplicate file from the SDK. The pathing to locate the duplicate file appears in the error message but can also be located by copying the full file path, demonstrated in Figure 6.2.

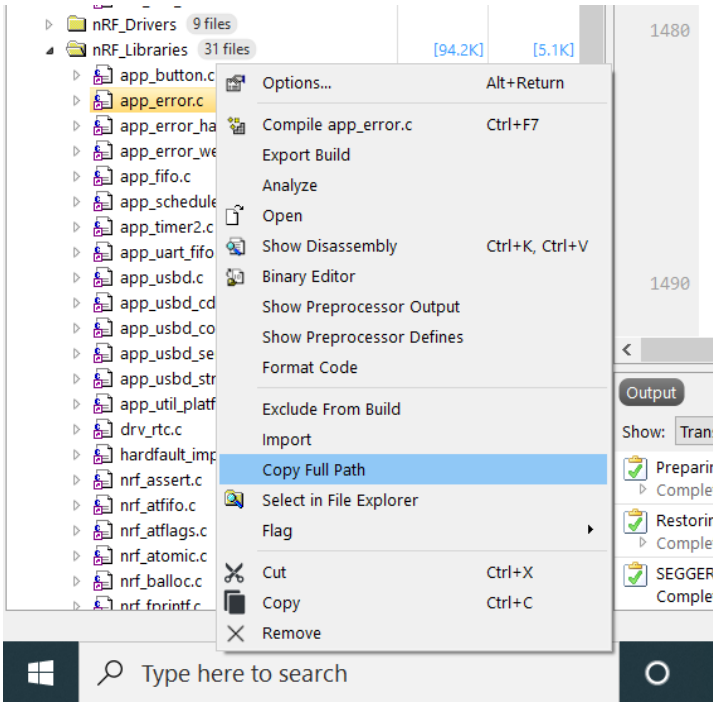


Figure 6.2: File Path Discovery

The pathing within the SDK is identical to the pathing provided from the Github repository project. By selecting to remove the existing .c file and dragging in the duplicate from the SDK, the errors produced for unrecognized .c files should be resolved. After the .c files are recognized, the following errors should state that the .h files cannot be located or are unrecognized. The .h files are located within the dependencies folder of each .c file, as shown in Figure 6.3.

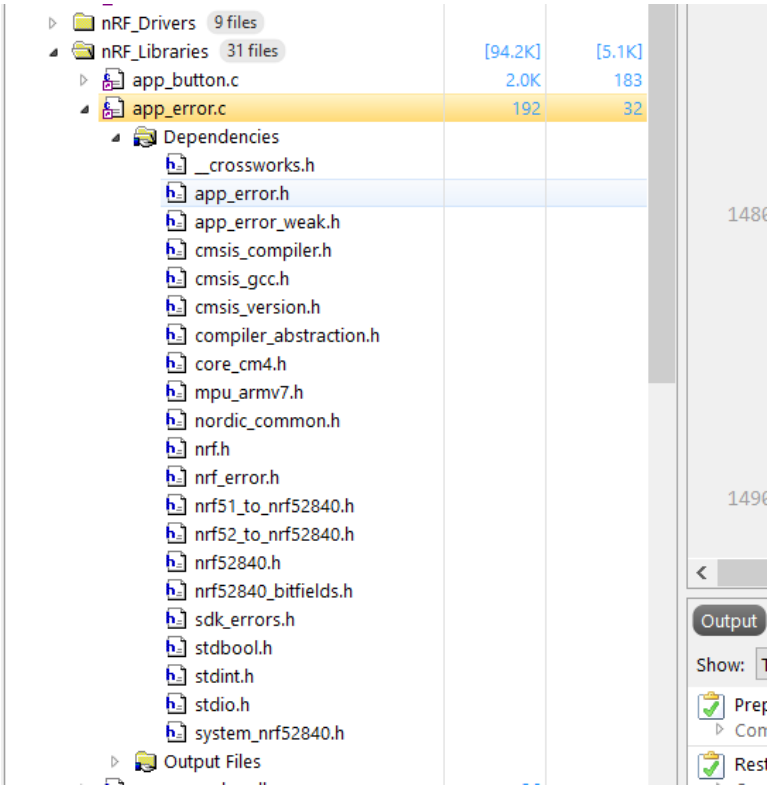


Figure 6.3: Location of .h files

Adjusting for these error messages can be completed in an identical manner as for the .c file error messages. However, given the significantly larger quantity of .h files, fixing these errors in a similar manner would be a monotonous task. Rather, Segger is looking for the root of the dependencies to exist directly in the computer's default local disk. By copying four folders from the SDK (components, external, integration, and modules) directly into the main drive on your local disk the error for unrecognized should be resolved. With the errors regarding .c file and .h files resolved, the environment is fully operational and the code can be downloaded to the

Microcontrollers. It is important to note that USB cables capable of serial data transmission are required to download the code, as opposed to cables solely providing the power source.

6.3: Data Transmission and Reception

The embedded project is divided into code for a peripheral node and code for a central node. Once the code for the peripheral node is downloaded to the peripheral Microcontroller, that Microcontroller no longer needs to be connected to your personal computer. Rather, it can be powered with a USB cable by the USB port of a phone charging block, another laptop or desktop, etc. Data connections are made to the peripheral board via pin P0.03 (A0) and any GND connection (shown in Figure 6.4).

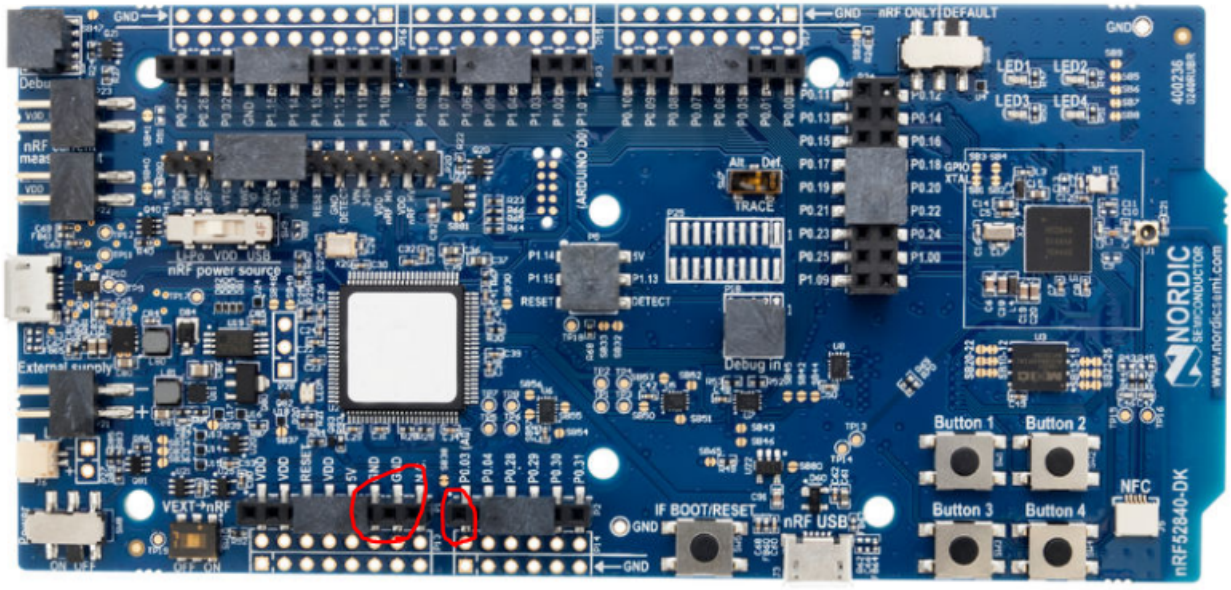


Figure 6.4: Data transmission pin locations marked in red

Our project only utilizes one active peripheral node, however, the software is designed to receive data from two peripherals simultaneously. Therefore, we included a "dummy" second peripheral node in our configuration. The resulting second channel of data will eventually be visualized as a flat, data-less signal. The central node must be connected to your personal laptop for data reception and conversion within MATLAB. The central Microcontroller is powered with

the USB port on the left side of the board and sends data via the USB serial port on the bottom of the board (shown in Figure 6.5).

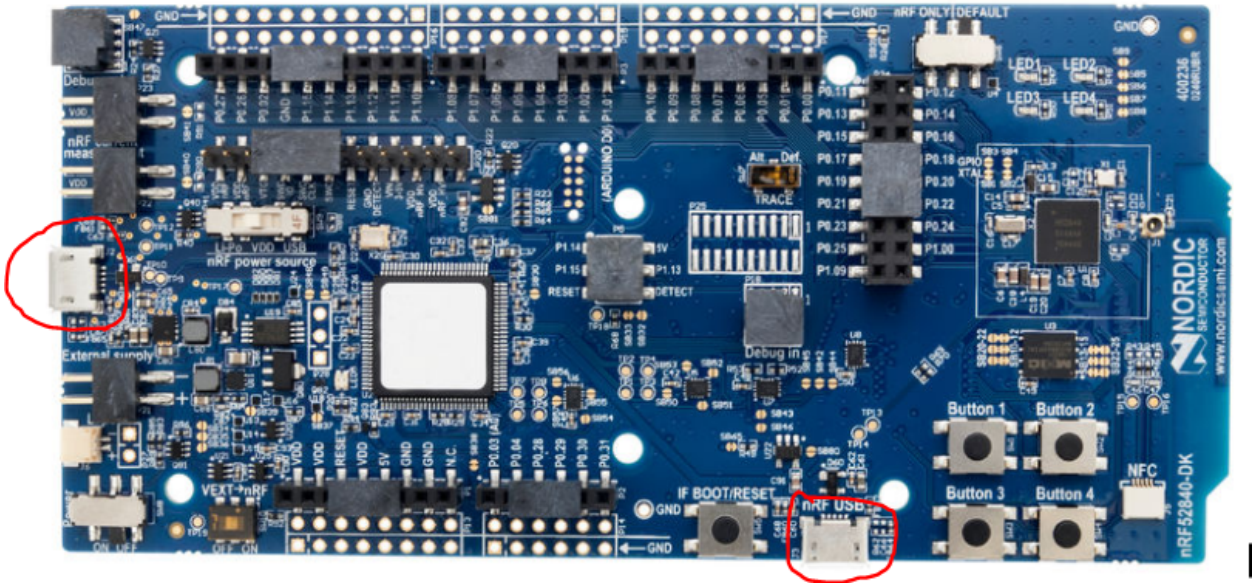


Figure 6.5: Power and data transmission ports marked in red

When in the debugging menu of Segger Embedded Studio for the central node, the console displays messages indicating whether the central and peripheral nodes are connected or not. Furthermore, LED1, located in the top right of all three Microcontrollers, continuously blinks if unconnected and stays lit a solid green once connected. Once connected, the console displays messages indicating the packets of data transmitted from each of the two peripherals. If the packet quantity does not coincide between the two, pressing the BOOT/RESET button to the left of the data transmission USB port on all three Microcontrollers should fix that issue. Finally, a field titled EMG_Processed in the top right of the debug menu indicates data reception in the central node. If the numbers are constantly changing and displaying as red, the bluetooth transmission and reception are working properly.

6.4: Data Visualization

Within the Github repository, along with the code for Segger Embedded Studio, comes a MATLAB code file and a MATLAB app file. The MATLAB code file, *Convert2Float*, converts

the raw data transmitted from the Microcontrollers into readable floating point values for the MATLAB application. That application, *Nordic_RealTime_Test*, is a self-contained MATLAB program that has a user interface. By running *Nordic_RealTime_Test,* and ensuring that the proper COM Port is selected for the data reception (can be checked with your computer's device manager), the data received from the peripheral will be displayed on the screen.

There are, however, some tweaks necessary within the *Nordic_RealTime_Test* application to properly visualize an ECG signal. The MATLAB app was initially developed to convert and visualize electromyogram (EMG) signals. EMG voltage and frequency ranges differ from that of a diagnostic ECG frequency range. Thus, the time period within the MATLAB application and the y-axis limits must be adjusted. This can be accomplished by adjusting the plotbuffer and app.plotline_one statements within the code view of the application. Switching between code and design views can be accomplished by clicking on the tab shown in Figure 6.6.

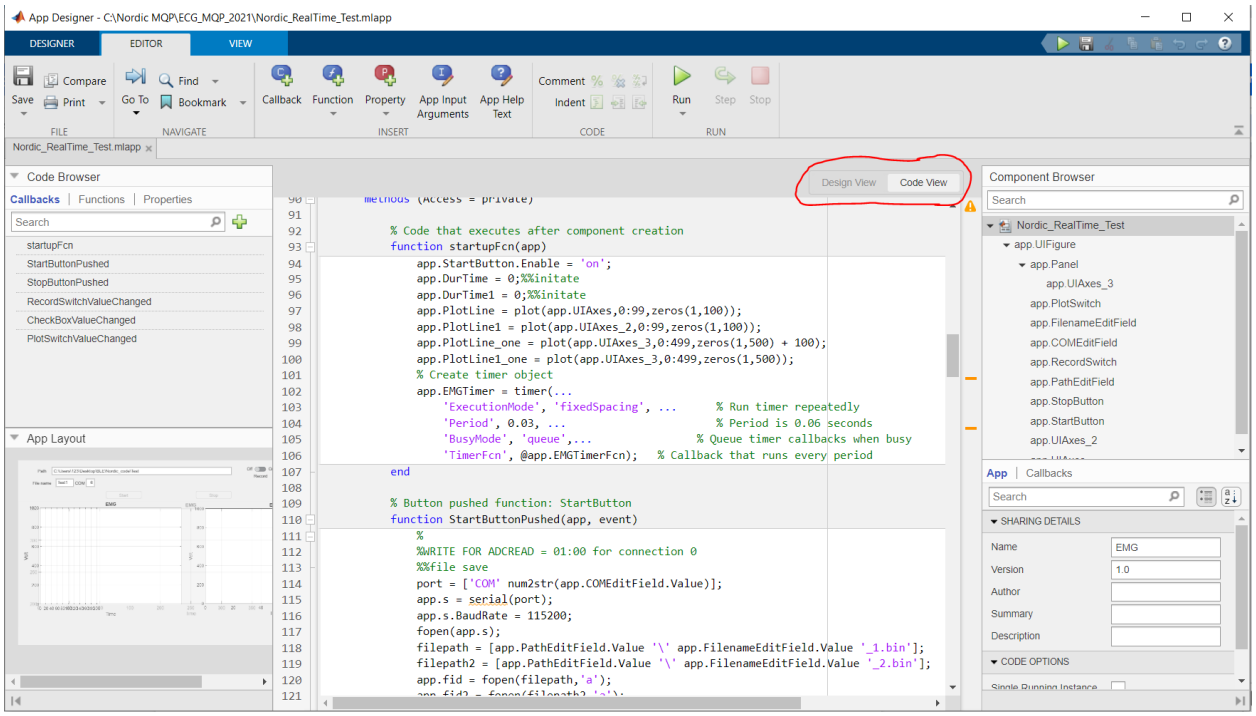


Figure 6.6: MATLAB app layout

Within the design view of the MATLAB app, under the component browser, the xTick and xTick labels should be adjusted to display the proper time scale. Furthermore, the xLim and yLim can be adjusted to view more of the signal along the x-axis while properly focused on the y-axis. The component browser within the design view can be seen in Figure 6.7.

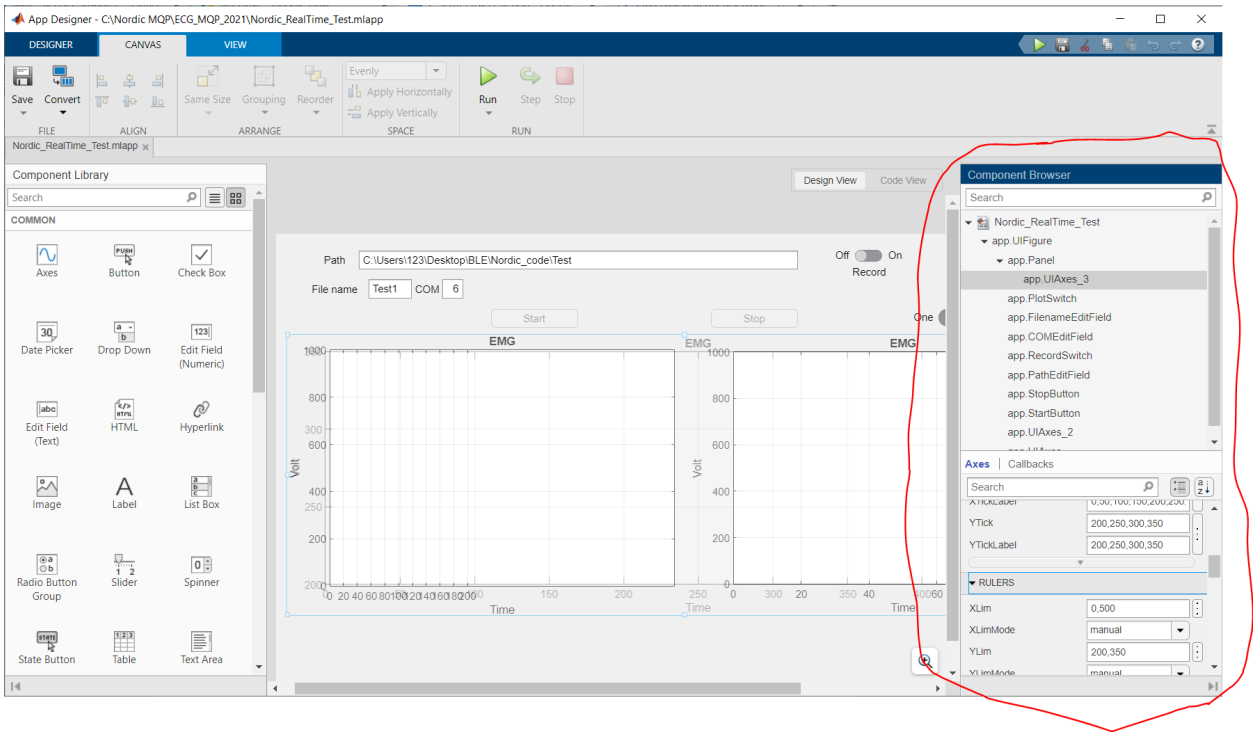


Figure 6.7: MATLAB app design view component browser

By clicking the green run arrow at the top of the MATLAB application the data will now be properly received and visualized. The data can be recorded and stored as a binary file after pressing the record button in the user interface and providing an available path location in your file system. A final MATLAB script, titled *LogData*, is available from the GITHUB repository that plots the stored binary file data across the entire recorded time period. An example of a sample ECG signal after *LogData* processing is shown below in Figure 6.8.

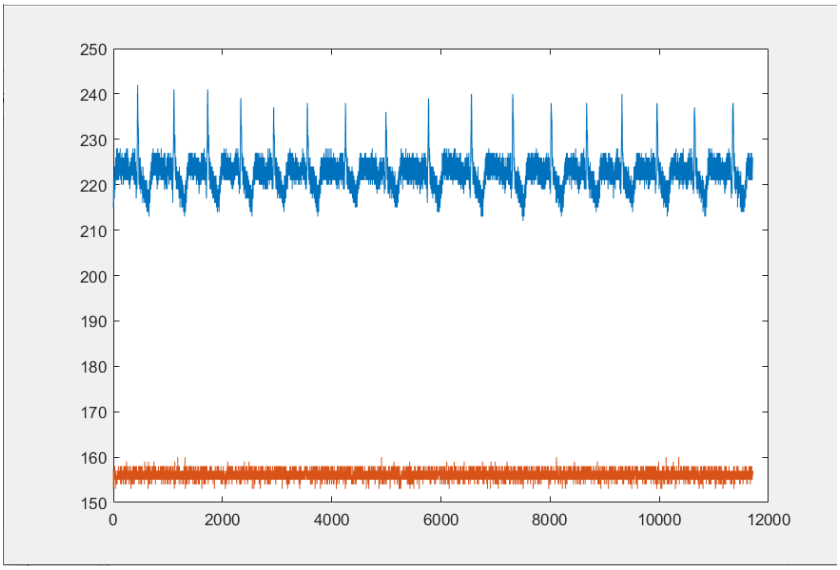


Figure 6.8: Amplitude vs Samples of ECG signal after LogData *processing*

As can be seen in Figure 6.8, while the ECG signal is discernible there is a significant degree of noise present. Further real-time filtering within the embedded side of the circuitry is necessary for a smooth final waveform. A notch filter was implemented to remove potential 60 Hz noise generated and an additional butterworth lowpass filter with a cutoff of 150 Hz was implemented to remove high frequency noise outside of the diagnostic ECG range. Images of the ECG signal after just notch filtering and after complete filtering can be seen in Figures 6.9 and 6.10. It is important to note that the ECG signals seen in Figures 6.8 and 6.10 were all achieved using the same test subject, the same lead configuration, and were all taken within the same half hour window of time.

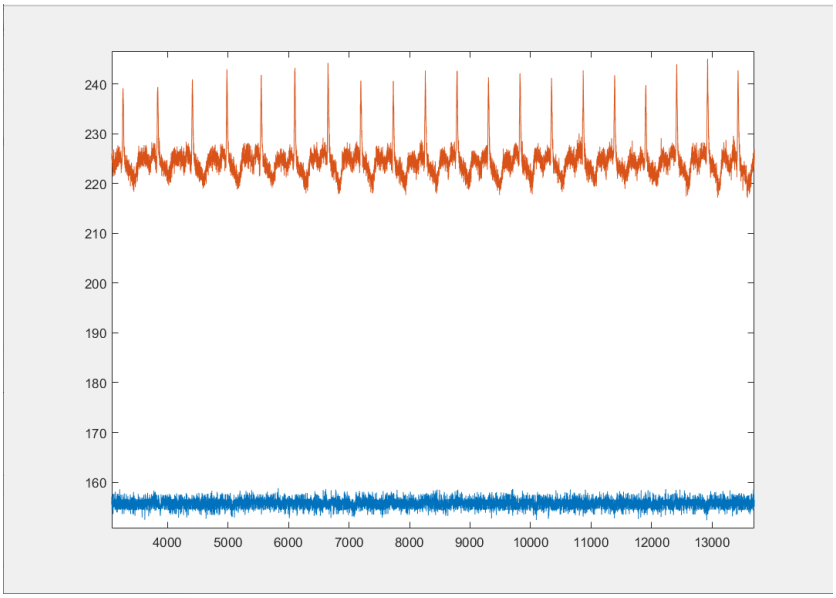


Figure 6.9: Amplitude vs Samples of ECG signal after Notch Filtering

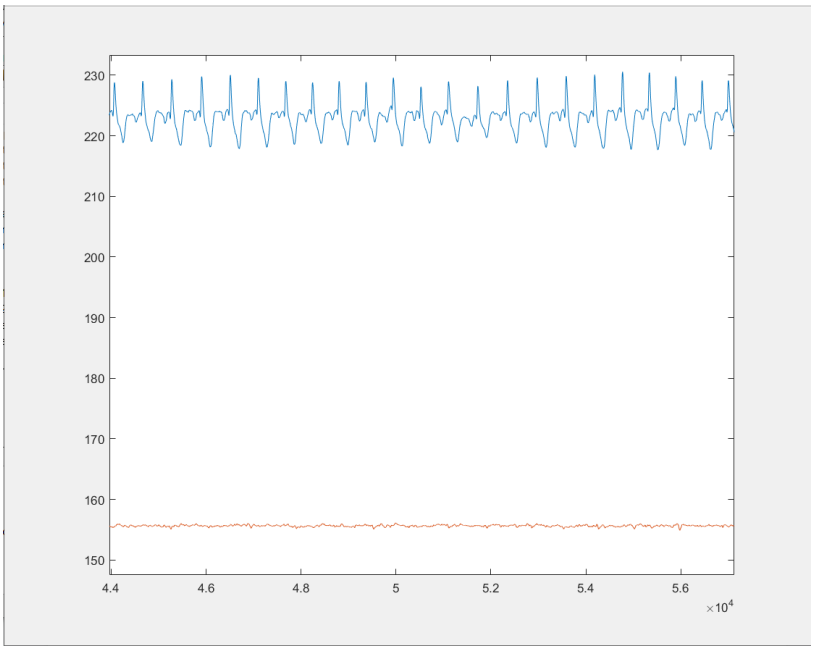


Figure 6.10: Amplitude vs Samples of ECG signal after Notch and Lowpass Filtering

7: R-Wave Detection Algorithm

Our final R-wave detection algorithm was designed in MATLAB and inspired by Pan and Tompkins' QRS Detection Algorithm. The algorithm was tested using MIT/BIH databases, available on Physionet, where we had access to annotation files that allowed us to compare the expected R-wave locations with what our algorithm produced. The completed ECG circuit collected the signals, transmitted the data wirelessly via bluetooth, where the R-wave detection algorithm was run. Annotation files were created to check the accuracy of the detector using our collected data. The goal of this detector was to be as optimized as possible to detect R-waves in different scenarios, especially in noisy data, and to use the output R-waves as the input of the atrial fibrillation detector so that atrial fibrillation can be detected. Our implementation of the R-wave detector can be found in Appendix D.

7.1: Design Process

The R-wave detection algorithm was implemented using Pan-Tompkins' QRS detection algorithm as a reference. Guided by a MATLAB implementation available online [40], we wrote our own version of the algorithm by creating a new bandpass filter, removing the derivative filter, and modifying the moving average from convolution to a Butterworth lowpass filter. The algorithm was also optimized to read in the sampling frequency so that it could be used with any sampling frequency, and not just a 200 Hz sampling rate, which is what Pan-Tompkins designed their algorithm for. Figure 7.1 outlines the steps in the R-wave detector that lead to a resulting index of R-waves.

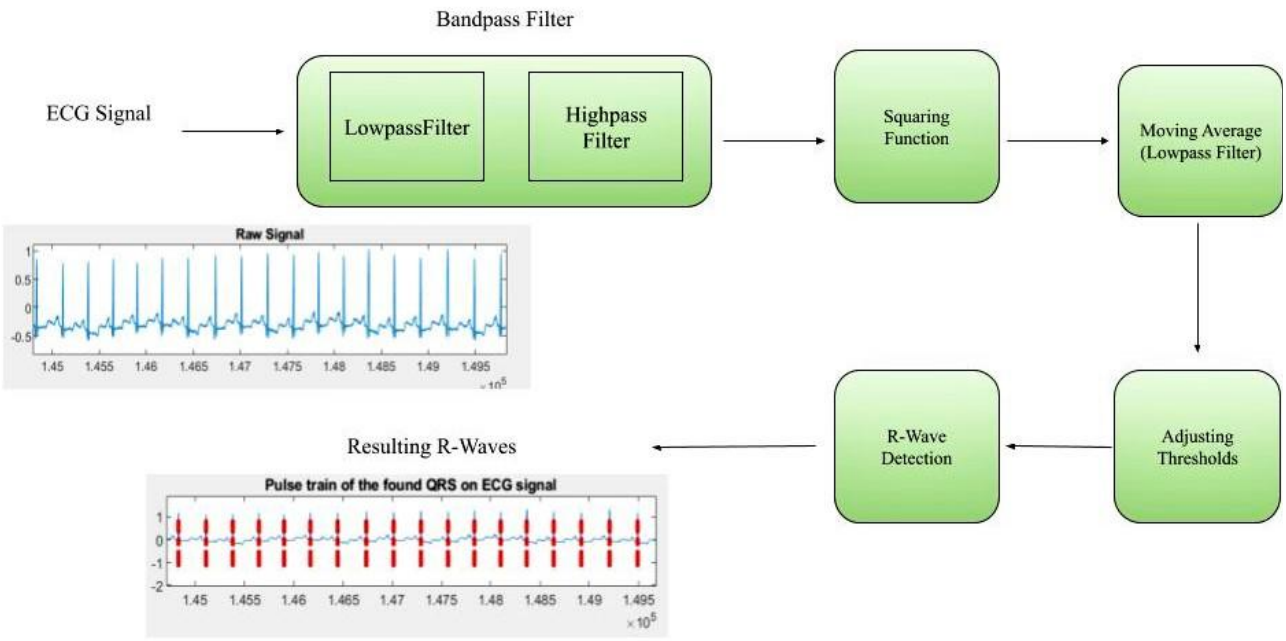


Figure 7.1: Black-box Diagram outlining the steps of the R-wave Detection Algorithm

Since the Pan-Tompkins R-wave detection algorithm was designed over 30 years ago, it was crucial to take into account the extra parameters that they included to run the algorithm over significantly slower systems. Since modern day systems run more efficiently and there are functions in MATLAB available to allow us to design filters, we decided to modify the algorithm to use those. Specifically, the bandpass filter that Pan-Tompkins designed was implemented by cascaded lowpass and highpass filters . In MATLAB, however, we were able to directly create a digital Butterworth bandpass filter using the butter() function. A Butterworth filter has a frequency response that is as flat as possible in the passband and has a monotonic roll-off in the stopband (as compared to other major filter types), which is what we needed for our bandpass filter. We chose to build a fourth-order bandpass filter with a passband between 5-15 Hz since that is the general range of where R-waves are located [25]. Varying this high cutoff frequency allowed signals below 15 Hz to pass through. If we change it to, for example 12 Hz, then signals above 12 Hz will not pass through, which can be better if there is a record with a lot of high amplitude noise, but generally lower amplitude signals.

In Pan-Tompkins' algorithm, they had created a derivative filter after the bandpass to calculate a slope. However, in early testing, we realized that the derivative filter did not have an effect on our results, so we decided to remove it.

We also designed a lowpass filter when performing the moving average after squaring. Initially, Pan-Tompkins used convolution for moving average since that was more efficient on their hardware systems; however, moving average is a poor lowpass filter with slow roll-off and poor stopband attenuation, so we created a filter with similar characteristics to replace convolution. Once again, using the butter() function, we created a second-order lowpass filter with a cutoff frequency of 5 Hz.

Once the filters were created, we needed to set the thresholds for the detection phase. We decided to keep most of this section of the MATLAB implementation because it matched with what Pan-Tompkins' did for their algorithm.

First, we set the fiducial marks that serve as a reference for the locations of potential R-waves. Since the heart has a refractory period of about 200 ms, we ensured that all peaks that are located will be at least 200 ms apart. Next, we initialized the learning phases; the first learning phase is to use the first two seconds of data to set the running estimates of the signal and noise thresholds for both the bandpass filtered and the moving averaged signal. For the second learning phase, we used the first two heartbeats to initialize the RR-interval average and calculate RR-interval limit values. After that, we initialized the two sets of thresholds using the bandpass filtered signal noise peaks and signal peaks and the moving averaged signal noise peaks and signal peaks using equations 2.24-2.31. The higher threshold, for both the bandpass and moving average, is usually the signal height threshold, which is what the detector determines where most of the peaks lie, and the lower threshold, for both the bandpass and moving average, is usually the noise height threshold, which is what the detector determines where most of the noisier peaks lie. The lower one is only used during searchback if there is no peak caught by the higher threshold in the 200 ms time interval.

Additionally, just like Pan-Tompkins, we take the average of the eight most-recent beats and the average of the eight most-recent beats that fall within certain limits, set by equations 2.34-2.36. We determine if any peaks are T-waves, and skip a peak if there is more than one peak detected in less than a 200 ms time interval. Finally, we take the SNR into account and adjust the

thresholds when the signal and noise thresholds are not zero, before plotting our final signal with the R-wave detections overlaying on the plot with a red dashed vertical line.

7.2: Design Verification

To verify that the R-wave detection algorithm works as intended, we compared graphical results to the MATLAB implementation of the Pan-Tompkins algorithm and compared the detected locations of the R-waves to the "truth" locations.

For the graphical results, we plotted the original ECG signal and the resulting R-wave detections determined from our algorithm implementation and the MATLAB implementation of Pan-Tompkins algorithm. Figures 7.2 and 7.3 demonstrate that for normal sinus rhythm, the outputs from the Pan-Tompkins and our algorithm match.

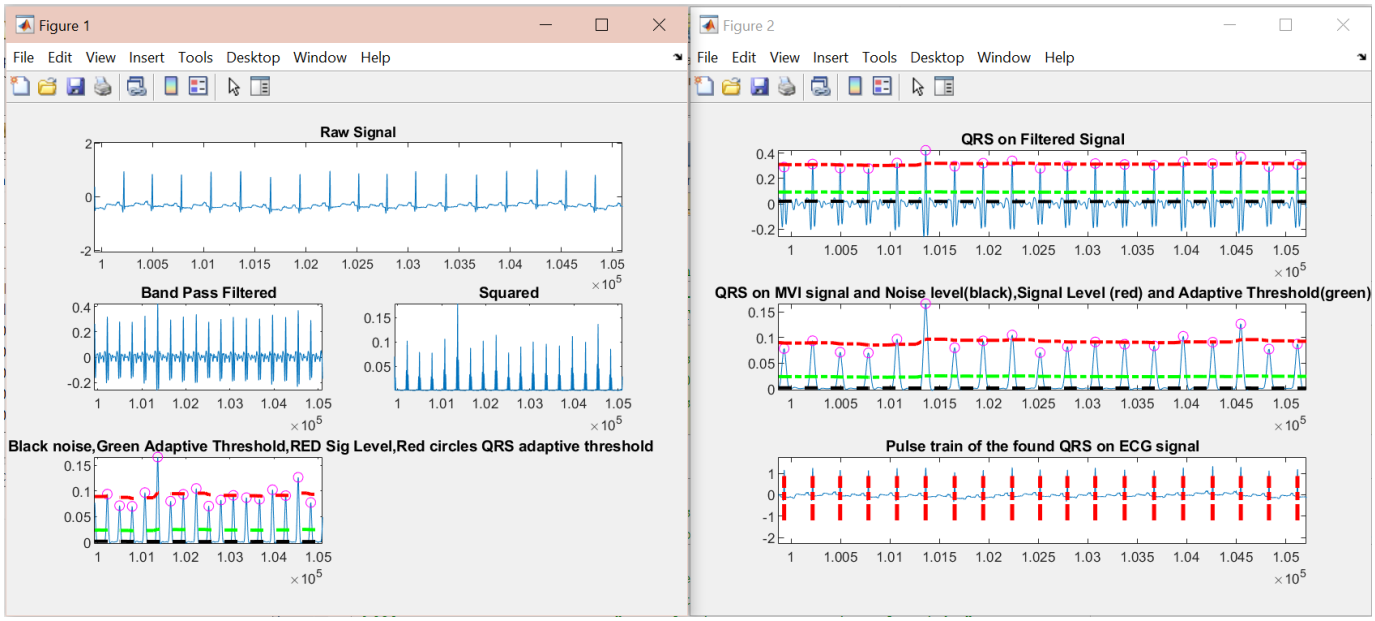


Figure 7.2: Pan-Tompkins Algorithm Implementation reacting to normal sinus rhythm for Lead I of subject 100

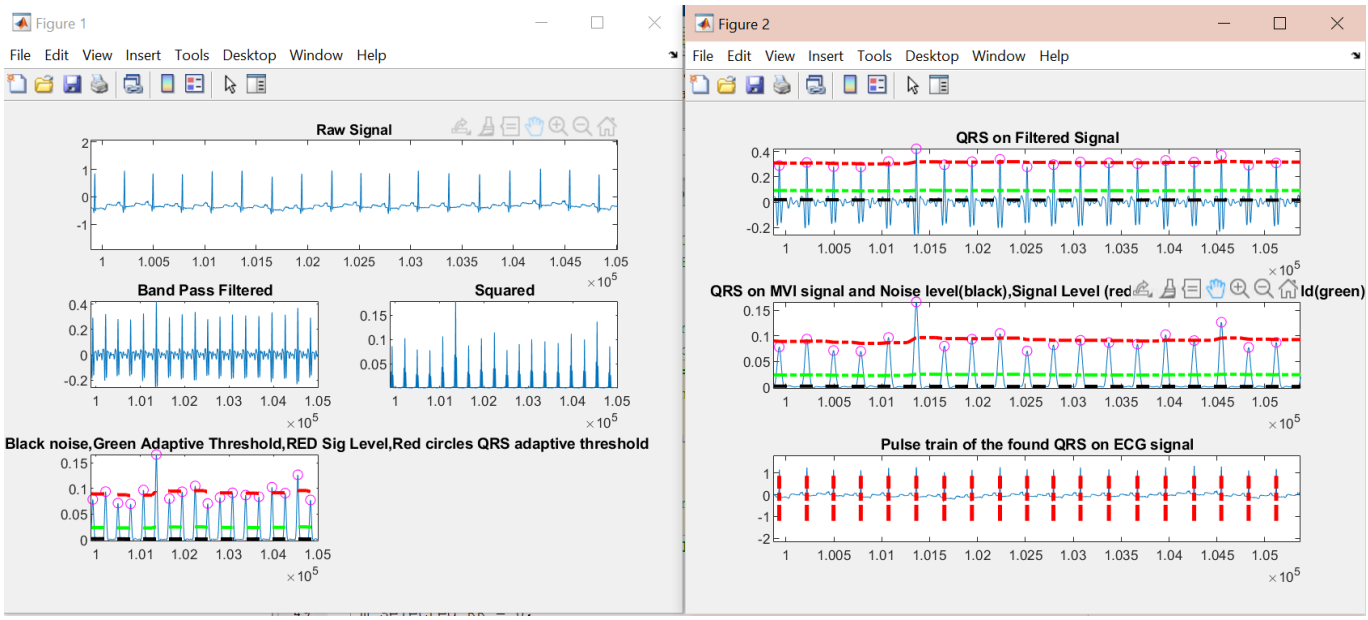


Figure 7.3: Our R-Wave Detection Algorithm reacting to normal sinus rhythm for Lead I of subject 100

As part of testing, we manually reviewed portions of the plotted signal that have artifacts to see how the algorithm performs. Specifically, instances of premature ventricular contractions (PVC) and possible muscle noise have some differing QRS trains when comparing the Pan-Tompkins MATLAB implementation and our algorithm. Figures 7.4 and 7.5 compare the different outputs from Pan-Tompkins and our algorithm.

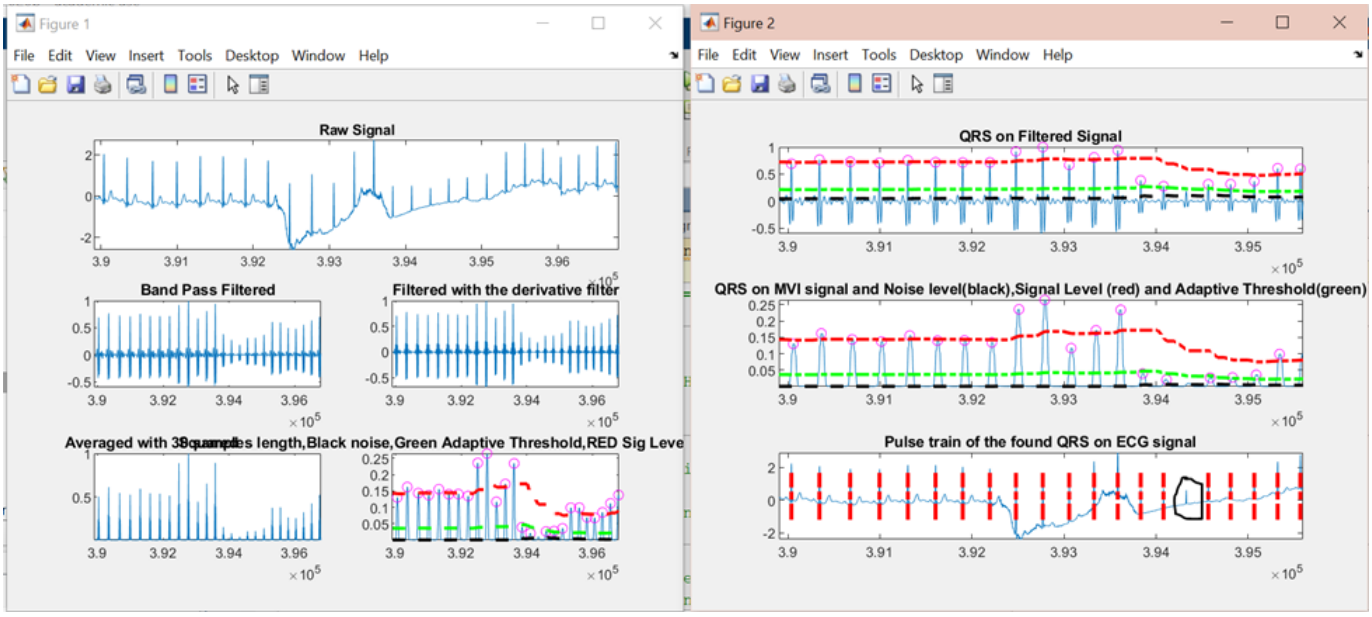


Figure 7.4: Pan-Tompkins Algorithm Implementation reacting to a PVC for Lead I of subject 103

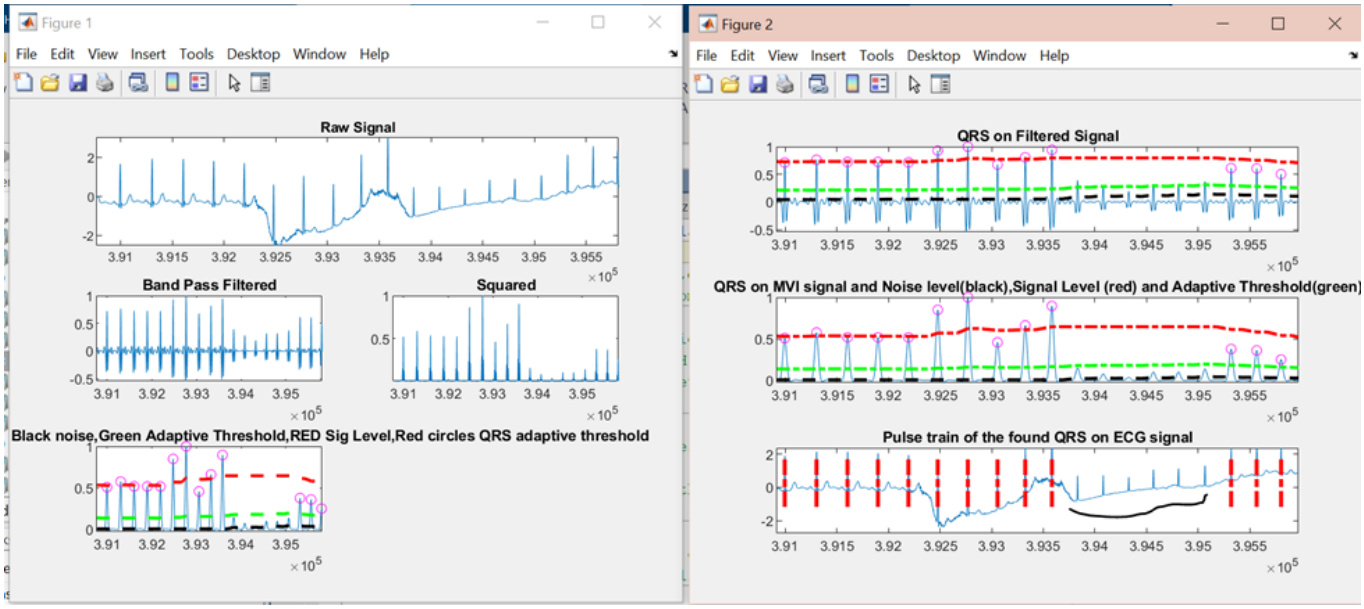


Figure 7.5: Our R-Wave Detection Algorithm reacting to a PVC for Lead I of subject 103

Specifically in Figures 7.4 and 7.5, there are R-waves that were detected in Pan-Tompkins' algorithm implementation between 3.94×10^5 and 3.95×10^5 samples that are not identified in our detection algorithm. This is likely due to the trend where large amplitudes in the dataset increase the threshold, making smaller R-wave peaks negligible.

To verify that the R-wave detection algorithm works as intended, we used the Physionet MIT BIH Arrhythmia Database and the MIT BIH Atrial Fibrillation Database. The Arrhythmia Database contains 48 half-hour long recordings that are sampled at 360 Hz. The Atrial Fibrillation Database contains 25 ten-hour long recordings that are sampled at 250 Hz and have a total of 299 episodes of atrial fibrillation. Both databases come with annotation files that state the truth locations of R-waves and atrial fibrillation. The atrial fibrillation database also comes with some notes about the recordings in the database that might affect our data analysis. We made sure to take those scenarios into account when performing our algorithm on the atrial fibrillation database.

Using the two databases to test the algorithm, we used the corresponding annotation files for each record provided by Physionet to determine the actual location of the R-waves and compared them to the output of the R-wave detection algorithm, using the ANSI/AAMI EC57:1998 standard and the ANSI/AAMI EC38:1998 standard. This was accomplished using the WFDB Toolbox, which reads in information from Physionet. For our purposes, we read in the raw data from the Physionet database and the annotation files that list the exact locations of the R-waves and also contain information about the sampling frequency.

Some key functions from the WFDB Toolbox that we used are “rdann”, “rdsamp”, “wrann”, and “bxb.” The function “rdsamp” is used to read the ECG vector from a data file, read the sampling frequency from the corresponding header file, and determine whether to plot the signal or not. The “rdann” function is used to read the annotation file provided by Physionet, the “wrann” function is used to write to a file, and “bxb” is an ANSI/AAMI-compliant beat-by-beat comparator function which is used to compare the annotation files provided by Physionet and the output pulse train of our detection algorithm. For our purposes, we used “wrann” to write the output R-wave detection locations to a .test file, which we then used “bxb” to compare our annotation file with the “truth” file provided to us by Physionet.

7.3: R-Wave Detection Algorithm Design Validation

We wrote a comparator() function using WFDB MATLAB toolbox functions that creates a bxb() report, which is shown in Figure 7.6. The function can be found in Appendix E.1. In the report, the QRS sensitivity and QRS positive predictivity are noted. QRS sensitivity is

determined by how sensitive the detector is, and QRS positive predictivity is determined by how accurate the QRS detections are (i.e., the detection is within 150 ms of the truth annotation). Additionally, there is also a table that shows the number of R-waves detected and number of R-waves missed by the algorithm. This is denoted by 'N' and 'n', which means that they are beats that were successfully detected, 'Q' which are unknown beats, and 'O', which denote missed beats. Other annotations include 'V', which is premature ventricular contraction (PVC), and 'X' which is to denote incorrect beats.

```
Beat-by-beat comparison results for record 105
Reference annotator: atr
     Test annotator: test

                Algorithm
        n    s    v    f    q    o    x
  ------------------------------------------
 N |  2094    0    0    0    0   27    0
 S |     0    0    0    0    0    0    0
 V |    29    0    0    0    0    0    0
 F |     0    0    0    0    0    0    0
 Q |     3    0    0    0    0    2    0
 O |    19    0    0    0    0
 X |     5    0    0    0    0

           QRS sensitivity:  98.65% (2126/2155)
 QRS positive predictivity:  98.88% (2126/2150)
           VEB sensitivity:   0.00% (0/29)
 VEB positive predictivity:      -  (0/0)
          SVEB sensitivity:      -  (0/0)
SVEB positive predictivity:      -  (0/0)
     RMS RR interval error: 163.37 ms
```

Figure 7.6: bxb Report Example from subject 105

We wanted to automate collecting all the QRS sensitivity and QRS positive predictivity from the individual reports created from each record. Thus, we wrote the bxbreport_table() function, which can be found in Appendix E.2. The function reads the bxb report that is inputted into the function, and then modifies a .csv file that tabulates each of the results. From there, we were able to use Microsoft Excel functions to perform statistical analysis, such as calculating the average and standard deviation, to understand the overall accuracy of our algorithm.

We started by comparing our results to Pan-Tompkins' results for the arrhythmia database as stated in their paper. Their subject-by-subject results are listed in Appendix C.1. All of their

testing was done on Channel 1, as they realized that Channel 2 generally has a lower amplitude ECG signal and has poor SNR. Another observation is that the number of beats our detector found is less than what Pan-Tompkins found, and this could be due to Physionet making some modifications to the annotation files since Pan-Tompkins tested on their algorithm in 1985. The overall accuracy for their algorithm was 99.325%. Appendix C.2 and C.3 have the subject-by-subject results of our algorithm for the two databases, and Table 7-1 shows the accuracy of the QRS sensitivity and QRS positive predictivity.

Table 7-1: Accuracy of the R-Wave Detection Algorithm for Different Databases with a 5-15 Hz Bandpass Filter Cutoff and Second-Order Lowpass Filter

Database	QRS sensitivity (%)	QRS positive predictivity (%)
Arrhythmia	94.26 ± 17.61	99.78 ± 0.86
Atrial Fibrillation	83.40 ± 31.80	94.63 ± 13.58

There were three or four records in both databases that had a very low sensitivity percentage which is why our averages were affected. While they are full readings in themselves, they are outliers in our overall data analysis.

In an attempt to optimize and fix these outliers, our next step was to modify the orders and cutoff frequencies on the bandpass and lowpass filters and determine the effect they have on the performance results. Using that, we determined what the best set of parameters are to detect R-waves. The first modification we tested on was changing the higher cutoff frequency in the bandpass filter from 15 Hz to 12 Hz and changing the order of the lowpass filter for the moving average. As previously mentioned, the moving average is simply a poor lowpass filter, so decreasing the order of the filter deviates the stopband from reaching its ideal characteristics. We initially created a second-order lowpass filter, but to see the effect on the data, we changed it to a first-order. The subject-by-subject results that have the effect of the change on each record can be found in Appendix C.4 and C.5, and the average and standard deviations are displayed in Table 7-2.

Table 7-2: Accuracy of the R-Wave Detection Algorithm for Different Databases with a 5-12 Hz Bandpass Filter Cutoff and First-Order Lowpass Filter

Database	QRS sensitivity (%)	QRS positive predictivity (%)
Arrhythmia	95.44 ± 14.97	99.59 ± 1.63
Atrial Fibrillation	81.48 ± 29.69	94.87 ± 13.51

Some key takeaways we got after implementing the change is that the average QRS sensitivity for the arrhythmia database increased, but the average predictivity got worse. For the atrial fibrillation database, the average predictivity got a bit better but the average QRS sensitivity got worse. Comparing the values in the subject-by-subject data results, record 104 from the arrhythmia database initially had a QRS sensitivity of 27.63%, but after the modification to the bandpass and lowpass filters, it substantially improved to have a QRS sensitivity of 87.29%. Reducing the cutoff frequency in the bandpass filter from 15 to 12 Hz and changing the order of the lowpass filter to a first order greatly helped the detector take in more of the missed lower amplitude QRS signals, which is where the majority of the signals in this record are located.

While there were some improvements to the data after implementing a bandpass filter with a high cutoff at 12 Hz and first-order lowpass filter, some of the data that were previously fine did worse after the change. An example of this is record 07879 from the atrial fibrillation database, where in the original scenario, there was a QRS sensitivity of 99.17%, but after the modification had a QRS sensitivity of 67.53%. Other pitfalls of the modification is that for some records, while the other statistics were close to that for the second-order lowpass filter, the QRS positive predictivity would be affected, usually decreasing.

The second modification that we tested on was using the second channel of the ECG array. We used the same filter parameters as our first round of testing, which was a fourth-order bandpass filter with a passband of 5-15 Hz, and a second-order lowpass filter with a cutoff frequency of 5 Hz. Additionally, the ECG array that Physionet provides has two columns; one for Lead I, and one for Lead II. So far, all of our testing had been on Lead I and we wanted to see the effect that Lead II might have on the statistics.

Some key takeaways from this modification was that all of our outliers except one had a QRS sensitivity over 98% when using Lead II instead of Lead I, but there were other records that did not do well with the modification, some dropping from 95% or higher accuracy to less than 50%. The subject-by-subject results that have the effect of the change on each record can be found in Appendix C.6 and C.7, and the average and standard deviations are displayed in Table 7-3.

Table 7-3: Accuracy of the R-Wave Detection Algorithm for Different Databases by using Channel 2 instead of Channel 1

Database	QRS sensitivity (%)	QRS positive predictivity (%)
Arrhythmia	84.50 ± 28.29	97.29 ± 10.41
Atrial Fibrillation	87.024 ± 27.77	94.41 ± 13.65

The final modification we tested on was changing the higher cutoff frequency in the bandpass filter from 15 Hz to 12 Hz, changing the order of the lowpass filter for the moving average to a first-order, and changing the lowpass filter cutoff frequency to 6 Hz. The subject-by-subject results that have the effect of the change on each record can be found in Appendix C.8 and C.9, and the average and standard deviations are displayed in Table 7-4.

Table 7-4: Accuracy of the R-Wave Detection Algorithm for Different Databases with a 5-12 Hz Bandpass Filter Cutoff and First-Order Lowpass Filter with a 6 Hz Cutoff Frequency

Database	QRS sensitivity (%)	QRS positive predictivity (%)
Arrhythmia	95.50 ± 14.94	99.57 ± 1.67
Atrial Fibrillation	81.92 ± 29.56	94.88 ± 13.50

Our overall conclusion from this study is that while adjusting the cutoff frequencies and the order of the filters might have an effect on the accuracy of the R-wave detections, it is not enough to eliminate all of our outliers. As seen in the third modification, the QRS sensitivity and QRS positive predictivity are almost the same as the first modification where we had a lowpass

cutoff frequency of 5 Hz instead of 6 Hz. The peak detection section will need to be further modified with more complex logic to accommodate R-wave peak amplitudes changing greatly over the course of minutes or hours.

7.4: Discussion

Our initial set of parameters seem to have worked well with the least number of outliers in both databases, since the QRS sensitivity and positive predictivity are quite similar for both databases. The overall performance of the algorithm is about the same for both databases, which makes our algorithm useful for accurately detecting R-waves locations when later detecting atrial fibrillation.

After the initial set of testing, we noticed a QRS sensitivity of 85% or higher for 64/71 of the traces we tested from both the Physionet arrhythmia and atrial fibrillation databases, and a QRS positive predictivity of 91% or higher for 69/71 traces from both databases. In the effort to optimize our algorithm, our goal was to then eliminate the seven data outliers without compromising on the accuracy of those that are producing high accuracy.

After testing with a first-order lowpass filter and a cutoff frequency of 5-12 Hz on the bandpass filter, we noticed a QRS sensitivity of 85% or higher for 61/71 of the traces we tested from both the Physionet arrhythmia and atrial fibrillation databases, and a QRS positive predictivity of 91% or higher for 70/71 traces from both databases. While we were able to remove one of the seven outliers that the first set of testing produced, our overall improvement was not much better.

In another attempt to remove the outliers, we used the filter parameters from our first round of testing, a fourth-order bandpass filter with a 5-15 Hz passband and a second-order lowpass with a cutoff frequency at 5 Hz, and tested it on the second channel from the ECG array for the all records, with specific focus on the ones that were not doing well when using the first channel. Six out of seven of the records performed significantly better, with a QRS sensitivity above 98% and QRS positive predictivity above 97%. However, there were records that previously did well using Channel 1 that did not perform as well when using Channel 2. This is expected however, because generally one channel has a lower amplitude ECG signal and has poor SNR. What was interesting was seeing some results were about the same regardless of the

channel, which could simply mean that the SNR was about the same for both channels. The subject-by-subject results from this can be found in Appendix C.6 and C.7. The only record that has low QRS sensitivity and positive predictivity when using either channel is record 07162 in the atrial fibrillation database.

The final modification we did was change the highpass high cutoff frequency from 15 Hz to 12 Hz, lowpass cutoff frequency from 5 Hz to 6 Hz, and the lowpass filter order from a second to a first-order filter. After testing with this modification, we noticed a QRS sensitivity of 85% or higher for 62/71 of the traces we tested from both the Physionet arrhythmia and atrial fibrillation databases, and a QRS positive predictivity of 91% or higher for 70/71 traces from both databases. This modification was slightly better than the first modification we did, but it is overall almost the same.

Record 07162 was the only record that yielded a QRS sensitivity and positive predictivity of 33% or less for all the different modifications we tried. Zooming in and manually looking at the amplitudes of the signal, the peaks seem to be very erratic, some are at a very large amplitude and others are at a very low amplitude in a short period of time. Additionally, the R-waves appear to be inverted, as shown in Figure 7.7:

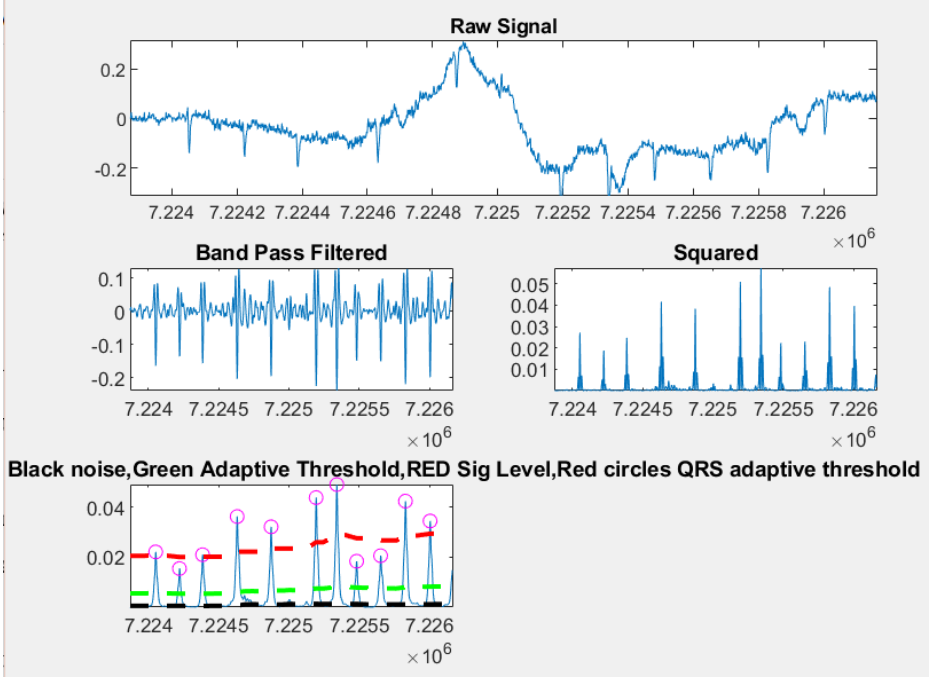


Figure 7.7: Record 07162 Raw Signal- Inverted R-waves in the raw signal for about 2000 samples

We would need to further optimize the R-wave detector, first by trying more combinations of cutoff frequencies and orders, and by implementing more complex logic for these kinds of situations. Another situation that we've noticed is that there are some really large peaks, maybe due to noise, that fall within the filter cutoff frequencies, but affect the thresholds that are set. For example, the 08219 record in the atrial fibrillation database has a higher amplitude in the first third of the data collected than the rest of the data, as shown in Figure 7.8.

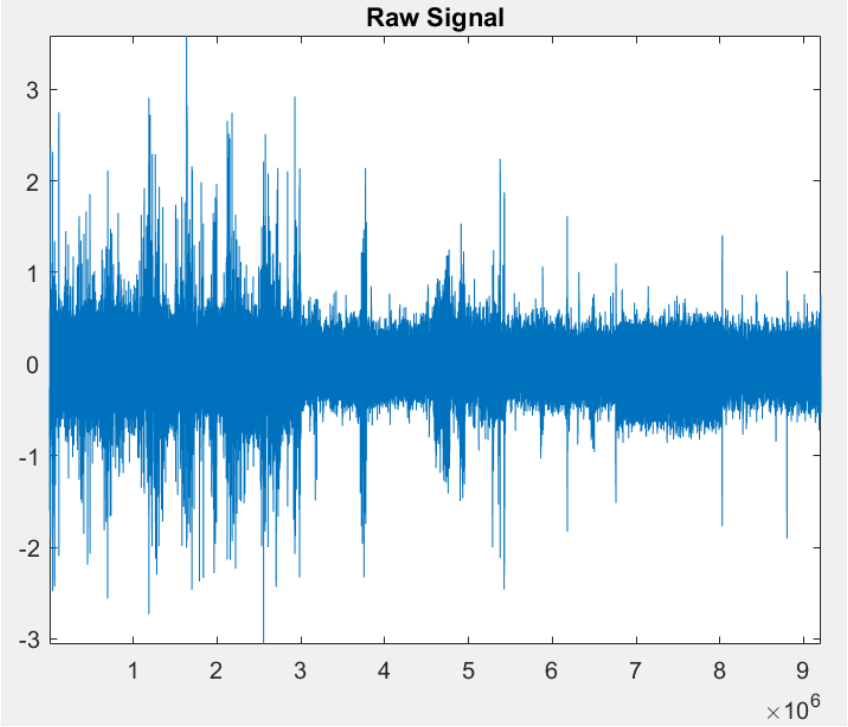


Figure 7.8: Record 08219 Raw Signal- The first $3x10^6$ samples have a significantly higher amplitude than the rest of the $6x10^6$ samples with the exception of some peaks

This is why, when changing the filter cutoff frequencies, the R-waves after one-third of the trace are difficult to catch by the detector. The thresholds may need to be better adjusted with more logic to accommodate special situations like these, since thresholds are determined primarily from the first two seconds of the recording. The rest of the outliers all have a similar pattern, with record 104 in the arrhythmia database being the exception, as described earlier.

A possible solution to fix the thresholds is to take the raw signal and split it into smaller intervals, such as an interval of 1000 samples, and then testing each interval individually to determine what the thresholds are being set to in each learning phase. Additionally, more R-waves would be detected from this method because the threshold from very high amplitude

peaks are not being considered, so it might be more accurate. However, the implementation and testing of this would be needed for a more accurate conclusion.

A reason for the amplitude change that was stable after the change could be due to a lead moving or falling off, which could be skewing the trace. With that said, there are a few spikes later in the graph that reach the earlier trace amplitude, which could be due to noise, so it is also possible that the first one third of the trace was very noisy.

Overall, Physionet was an excellent resource to provide arrhythmia and atrial fibrillation recordings to test our algorithm. It has allowed us to test and compare the effect of different parameters to determine what works best and what could be improved. We were also able to learn a lot about the WFDB toolbox and how to use it, since that was a roadblock early in the project.

8: Atrial Fibrillation Algorithm

This section discusses the methods used to develop the atrial fibrillation detection algorithm. A brief analysis of different RR interval segment lengths, window characteristics, and threshold values was conducted. Plans for future work are outlined at the end of the section.

8.1: Design Process

The atrial fibrillation detection algorithm was based on the algorithm created by Lee et al [1]. The first step in the development of the MATLAB algorithm was to read in the data downloaded from Physionet, specifically the MIT/BIH Atrial Fibrillation database. Physionet is a free database of physiological signals that have been reviewed by medical professionals to ensure the accuracy of all annotations. The MIT/BIH Atrial Fibrillation database contains 25 ECG recordings that are each 10 hours in length. ECG data and its corresponding annotations were read in using functions in the WFDB toolbox. This toolbox interfaces with the Physionet database and allows for easy manipulation of the data files. The WFDB toolbox was originally developed in the "C" programming language and using wrapper functions is compatible with MATLAB, the programming language we used for our algorithm.

Data from an ECG recording was first run through our R-wave detection algorithm and the locations of the waves was then used in the atrial fibrillation detection algorithm. For the purpose of testing the R-wave annotations from Physionet were used instead of our algorithm. This was done so both could be developed in parallel. The AF detection algorithm begins by calculating the differences between successive R-wave locations and creating a vector of RR-intervals. Before any calculations were done, ectopic beats were removed from the vector. This is done by first calculating the RR-interval ratio, using equation 2.48, between the current beat and the previous beat for all beats in the vector. The ratios are stored in a new vector. Next, the 1st, 25th and 99th percentile values of the ratio vector are found. The percentiles are then used as threshold values to indicate if a beat is ectopic using the logic from equations 2.49 - 2.51 in the Background chapter. If the beat is ectopic, it is removed from the vector along with the following beat.

The time varying coherence function was calculated for beat segments with a length of 128 RR intervals. The MATLAB function mscohere() was used to calculate the magnitude squared coherence for adjacent segments of the RR-interval vector. This function uses a Welch-based spectral estimate. A hamming window with 50% overlap was used as part of this function initially. The magnitude squared coherence represents the first half of equation 2.52 in the Background chapter. The two halves of the TVCF equation are complex conjugates, and therefore multiplying the two halves together is the same as squaring the results of one of the halves [41]. For our algorithm we decided to square the magnitude squared coherence function, instead of using the full TVCF equation from Lee et al., because of its simplicity.

The y-intercept of the TVCF for each beat segment was calculated by fitting a line to the data vs. frequency and used as an indicator of atrial fibrillation. Through testing of the algorithm, a y-intercept threshold value was determined to aid in classifying segments as atrial fibrillation or normal sinus rhythm. This replaces the frequency variation parameter used by Lee et al.

Shannon entropy was also found for the same RR-interval segments as the TVCF equation. The Shannon entropy function created was modified from a function found on Github created by Noah Goldstein, Dan Song, and Dan Thompson [42]. The method of Shannon entropy calculation, explained in the Background chapter, was implemented in this function. Once these values were found, the segments were classified as normal sinus rhythm or containing atrial fibrillation initially using the Shannon entropy and the y-intercept threshold values of about 0.79 and 0.6, respectively. For segments with a Shannon entropy above 0.79 and a y-intercept below 0.6, the segment is classified as atrial fibrillation

The results of the algorithm were tested using an episode-by-episode comparator function in the WFDB toolbox, called epicmp, which compares the results to the annotation file. The function compares the two annotation files using the ANSI/AAMI EC38:2007 standard and reports back the accuracy of the detected values versus the truth values. The parameters for segment length and the threshold values were varied to test their effect on the accuracy of the algorithm. The ideal values were determined to create the highest accuracy.

8.2: Design Verification

The first part of verifying the design of the algorithm was to test specific parameters in order to accurately detect atrial fibrillation. These parameters include the RR-interval segment length, the size and amount of overlap in the hamming window, the slope threshold values, the mean threshold values, and the Shannon entropy threshold values.

8.2.1 Optimizing Segment Length

During the initial development of the atrial fibrillation algorithm, a segment length of 128 beats was used. This value was suggested by Lee et al., but when implemented, resulted in significantly noisier TVCF values than those achieved in the journal article (Figure 8.1).

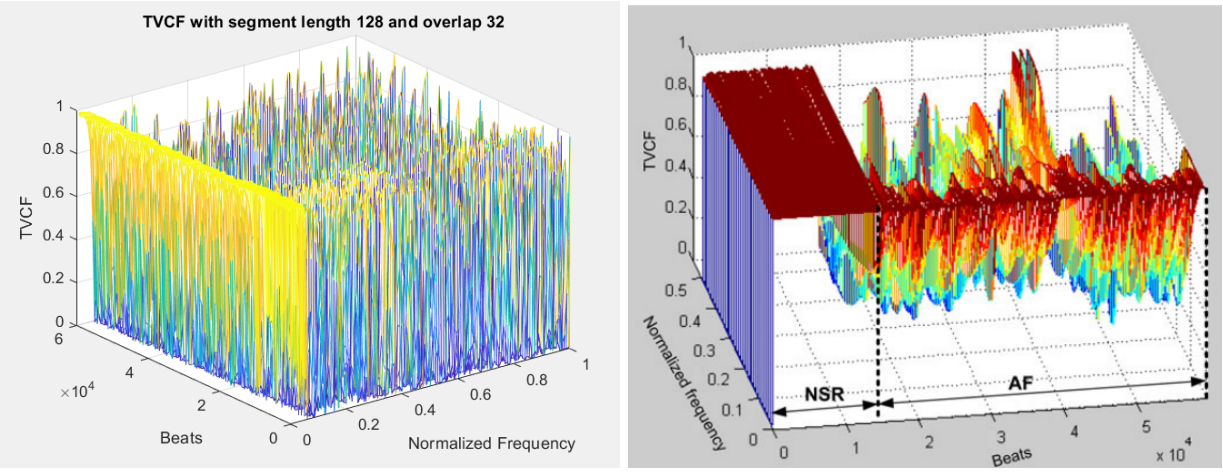


Figure 8.1: Plot of TVCF data for subject 08455 with RR interval length of 128. Our data left and Lee et al.[1] data right

In order to smooth out the data, different segment lengths were tested; the tested RR interval lengths were 128, 150, 192, and 256. The plots looked smoother with each segment increase but with each increase, the magnitude of the TVCF values decreased. This is not ideal because one of the indicators of normal sinus rhythm is the TVCF being near one across the frequency spectrum. The plots for segment lengths 192 and 256 are shown in Figure 8.2.

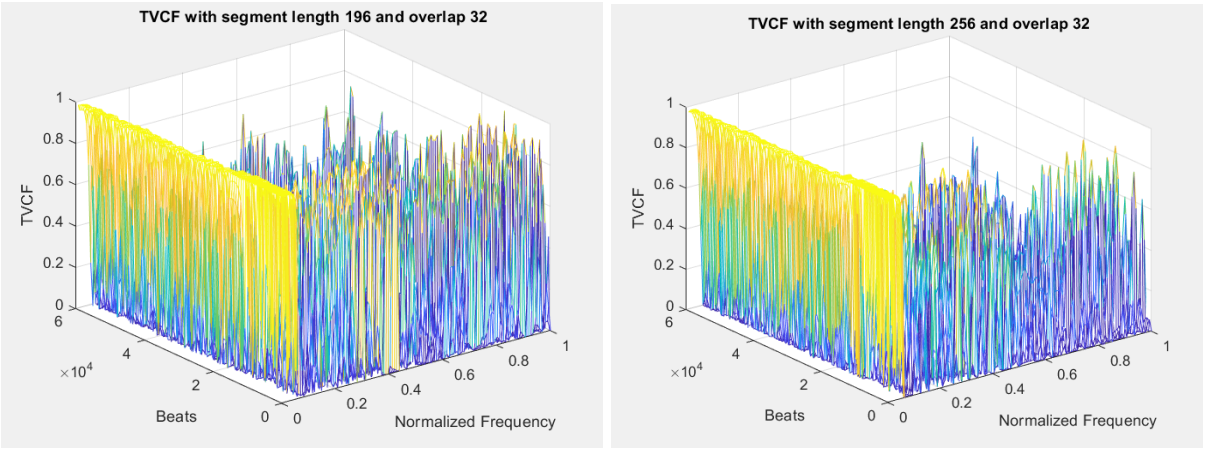


Figure 8.2: Plot of TVCF data for subject 08455. Segment length 196 left and segment length 256 right

Another concern with the increase in segment length is the loss of temporal resolution. For example, within a segment, an individual might go in and out of atrial fibrillation, and this information would be missed by the algorithm. Before a decision can be made on the segment length, the window size needs to also be tested to see the impact it has on the TVCF values.

8.2.2 Varying Window Size

We decided to vary the window size and type used when calculating the coherence in an attempt to reduce noise and create a clearer visual distinction between normal sinus rhythm and atrial fibrillation. The window used was changed from a hamming to a hanning window because in the time domain it reaches zero, and therefore eliminates all discontinuity [43]. The size of the window was reduced to a value of 15 samples instead of 64, the value used by Lee et al., and the amount of overlap was also reduced to 13 because the overlap must be smaller than the window size. The resulting TVCF values are plotted in Figure 8.3 below.

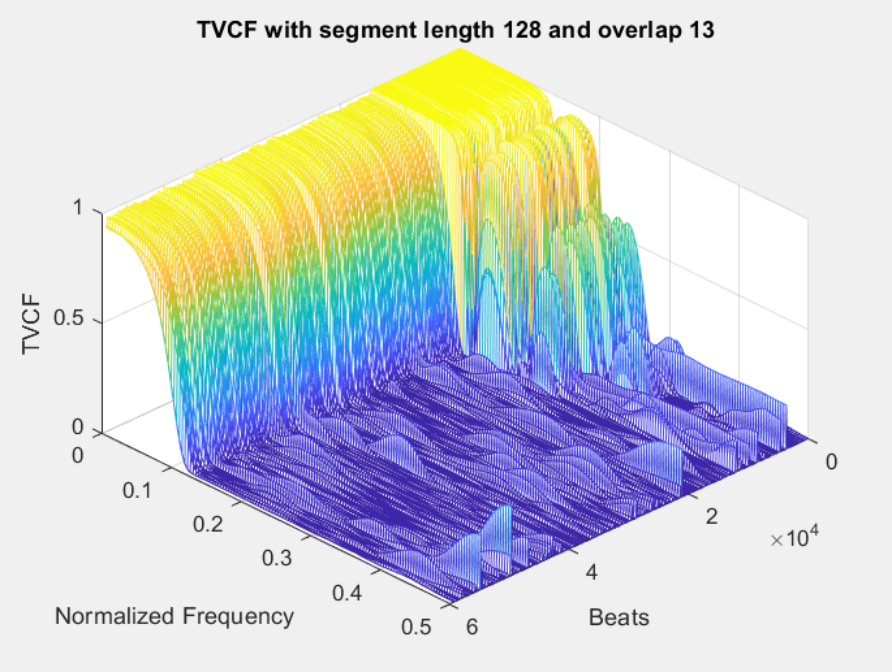


Figure 8.3: Plot of TVCF values for a segment length of 128, window size of 15, and an overlap of 13 subject 08455

The frequency resolution decreased when the window size was reduced. This was expected but the information lost may not be necessary for accurate classification; therefore, this loss of resolution may not affect the overall outcome of the classification algorithm.

8.2.3 Optimizing Window Overlap

During the calculation of the TVCF values from the RR-interval data the cross power spectral density is calculated. The amount of overlap in this window is automatically set to 50% of the window size and can contribute to the smoothness of the TVCF plot. By increasing the overlap, the expected response was a reduction of noise. The overall impact on the TVCF values was minimal but the higher amount of overlap did result in a greater degree of smoothing. The overlap values tested were at 50%, 75%, and roughly 90% of the window size for both a 64-point and 15-point Fourier transform window size.

8.2.4 Classification Parameters

The two parameters we used for the classification of atrial fibrillation were the y-intercept of the TVCF and the Shannon entropy of the RR interval segments. The y-intercept values for subject 08455 are shown in Figure 8.4. For the first 120 segments, this subject is in normal sinus rhythm and for the remaining segments, the subject is in atrial fibrillation. Figure 8.4a is the y-intercept with a segment length of 128, using a hamming window with length 64 and overlap of 60. Figure 8.4b shows the y-intercepts with the 128 beat segment length but using a hanning window size of 15 samples and an overlap of 13. The visual distinction between atrial fibrillation and normal sinus rhythm is significantly more visually distinct using the parameters from Figure 8.4b when tested on subject 08455.

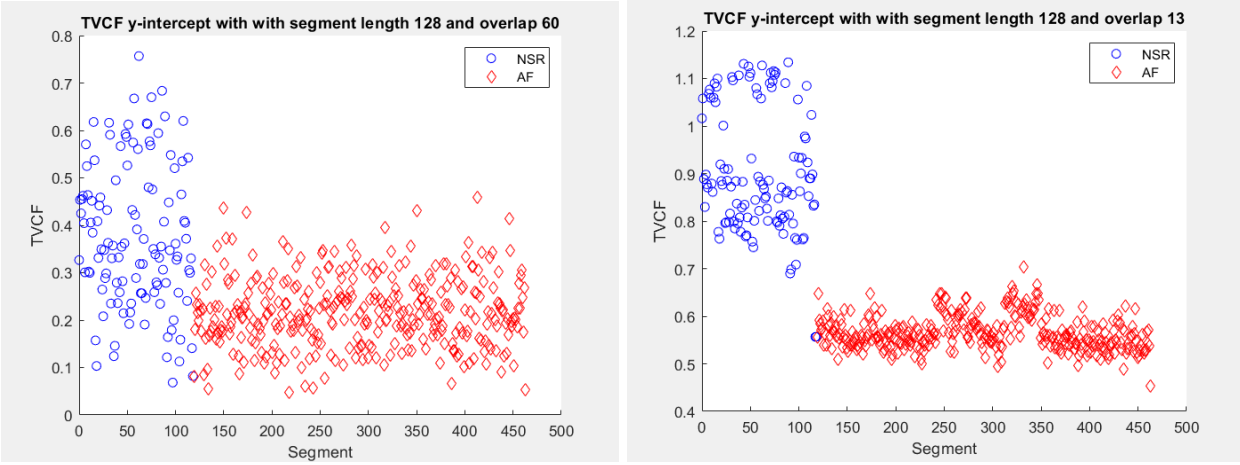


Figure 8.4a (left) and b (right): a)Y-intercept values of TVCF with segment length 128, window type hamming, window size 64, and overlap 60 b) Y-intercept values of TVCF with segment length 128, window type hanning, window size 15, and overlap 13 subject 08455

Shannon entropy for each segment was also calculated with the results plotted in Figure 8.5. The figure plots the results from subject 08455, where the transition from normal sinus rhythm to atrial fibrillation is around segment 120.

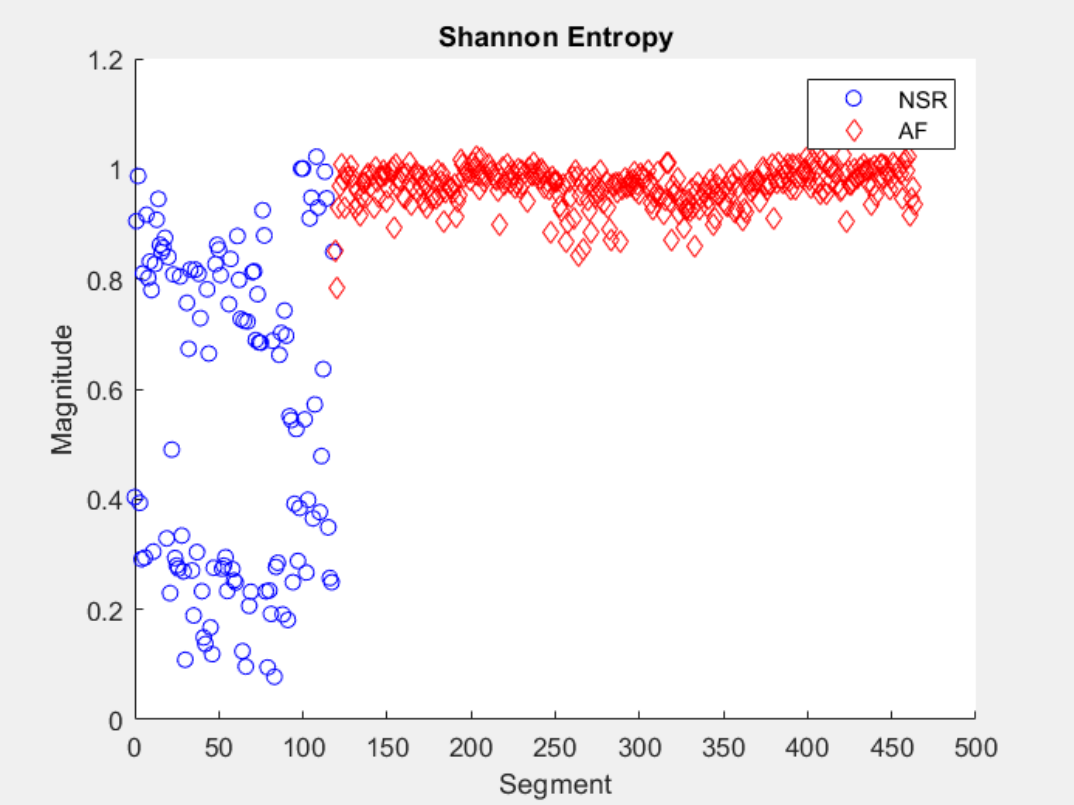


Figure 8.5: Shannon entropy segment length 128 subject 08455

8.3: Design Validation and Testing

For the classification portion of our algorithm, threshold values were set for y-intercept of the TVCF, mean of the TVCF and Shannon entropy. The initial values selected were estimated from Figure 8.6 where NSR is normal sinus rhythm and AF is atrial fibrillation. The thresholds chosen were 0.79 for Shannon entropy, as suggested by Lee et al., and 0.7 for y-intercept. For the purpose of testing the mean of the frequency values was also used as a classification parameter.

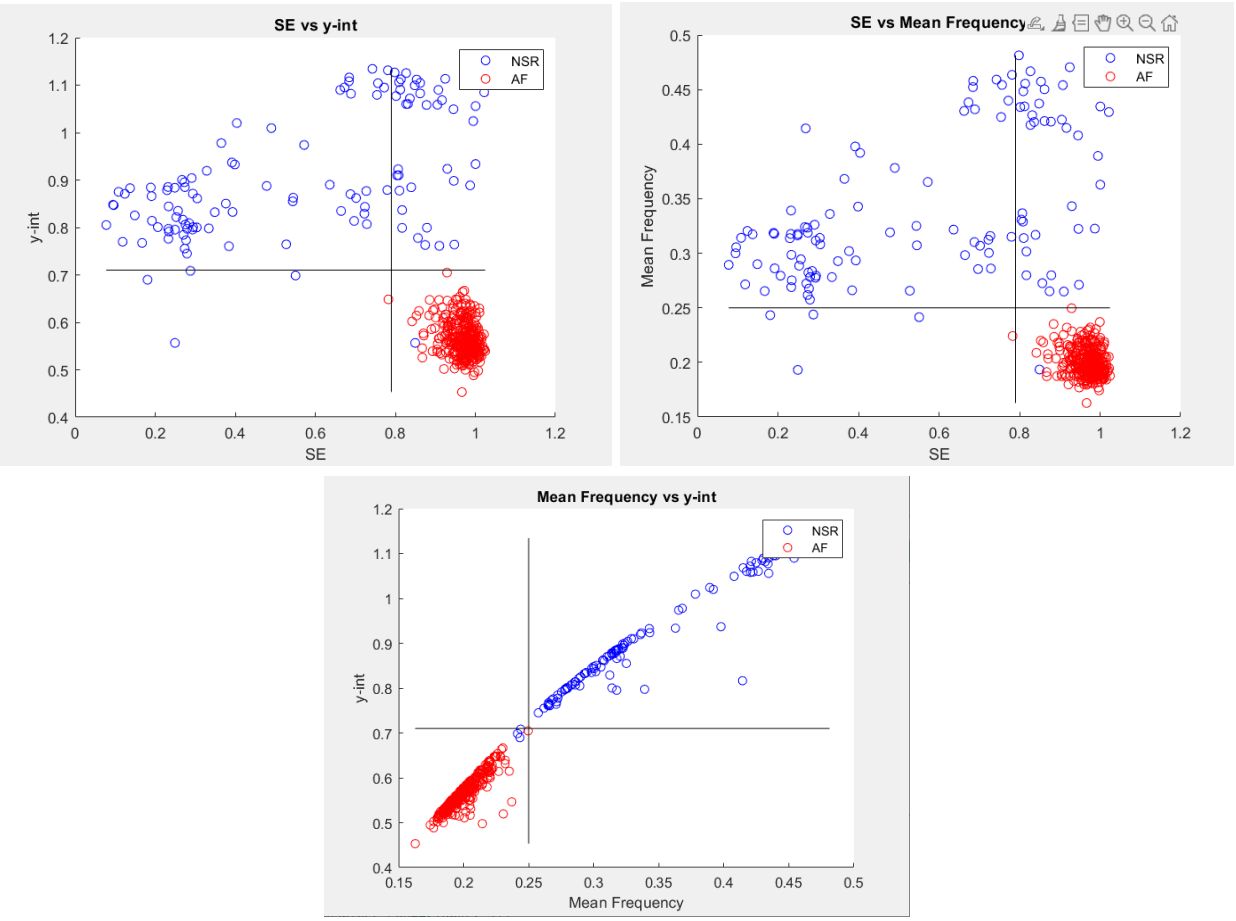


Figure 8.6: Shannon entropy vs y-intercept of TVCF(top left) Shannon entropy vs Mean Frequency of TVCF (top right) and Mean Frequency vs y-intercept of TVCF (bottom) for subject 08455

The first round of testing was performed on all subjects from the MIT/BIH Atrial Fibrillation database using the code in Appendix H. The hanning window size was set to 24 and the overlap set to 22 to begin. The parameters for y-intercept, mean, and Shannon entropy used for classification were 0.6, 0.225, and 0.79 respectively. The algorithm uses these thresholds to classify a segment of 128 beats as containing atrial fibrillation or normal sinus rhythm. If the y-intercept and mean values for the segment are less than the threshold values, and the Shannon entropy is greater than the threshold, the segment is said to be in atrial fibrillation. By using the episode comparator we were able to test the episode detected with the algorithm to the truth values provided in the annotation files. The report from the comparator provided the sensitivity,

positive predictivity, duration sensitivity, and duration positive predictivity. The averages of the sensitivity and positive predictivity were used to judge the effectiveness of the parameter thresholds.

The averages across all subjects for episode sensitivity and positive predictivity were 80.5%±24.25 and 29.36%±32.07 respectively. The y-intercept parameter was then varied between 0.6 and 0.44 in increments of 0.02. The mean parameter was also varied between 0.235 and 0.5 in increments of 0.01. Shannon entropy remained 0.79 for all testing. The parameter values of 0.44 of the y-intercept and 0.16 for the mean produced the best results with average sensitivity and positive predictivity being 75.91±27.99 and 72.23±27.31 respectively. The table of the parameters and their average values is in Appendix F. The average values for sensitivity and positive predictivity across all the subjects is displayed in Figure 8.7.

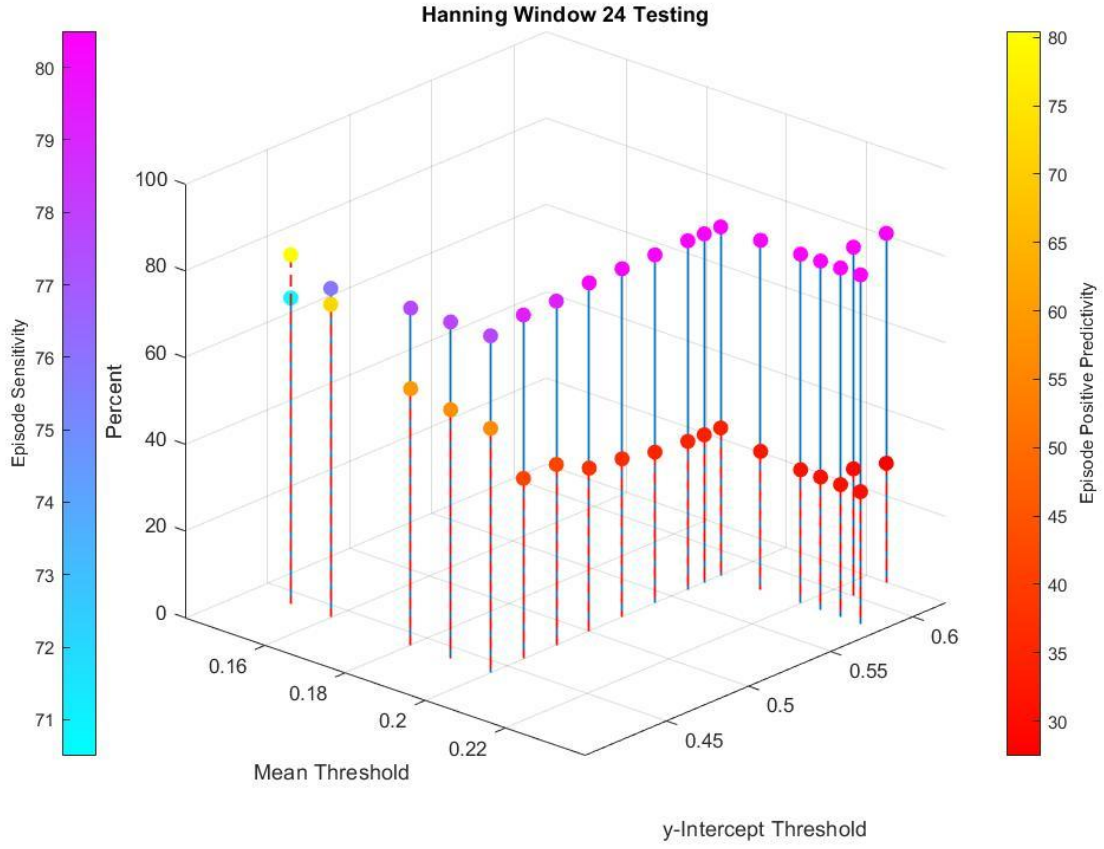


Figure 8.7: Average Sensitivity and Positive Predictivity plot for Hanning Window of 24

After seeing the results of the first round of testing, the window size and overlap were adjusted to improve the classification. The new window size was 16 with an overlap of 15. The altering of these values shifted the location of the y-intercept and mean boundary between NSR and AF. The y- intercept parameter was varied between 0.69 and 0.53 and the mean parameter value was varied between 0.25 and 0.21. The best parameter combination in this round of testing was a y-int of 0.59 and a mean of 0.21, with the average sensitivity being 70.95±33.03 and the positive predictivity being 83.73±26.99. The results for this round of testing are displayed in Figure 8.8.

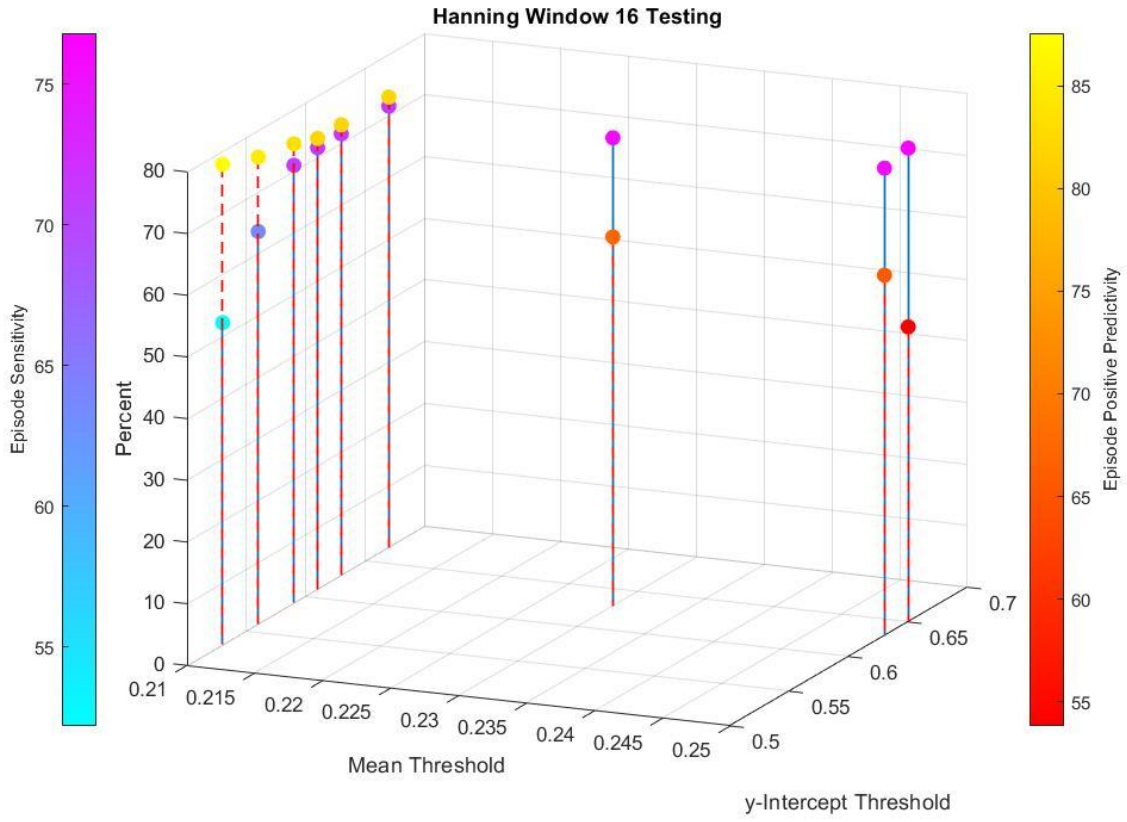


Figure 8.8: Average Sensitivity and Positive Predictivity plot for Hanning Window of 16

The third round of testing kept the window at 16 and the overlap at 15 but changed the frequency range the y-intercept is calculated over. The original frequency range was 0.04 Hz to 0.5 Hz in normalized frequency. Though it did remove some of the DC peak that is present in both NSR and AF, it was unable to remove all of it. The y-intercept was therefore changed to calculate the y-intercept for TVCF data from 0.3 Hz to 0.5 Hz. This modification greatly

changed the threshold value for the y-intercept. With the new parameter values, when the signal is in atrial fibrillation the y-intercept is close to zero (Figure 8.9).

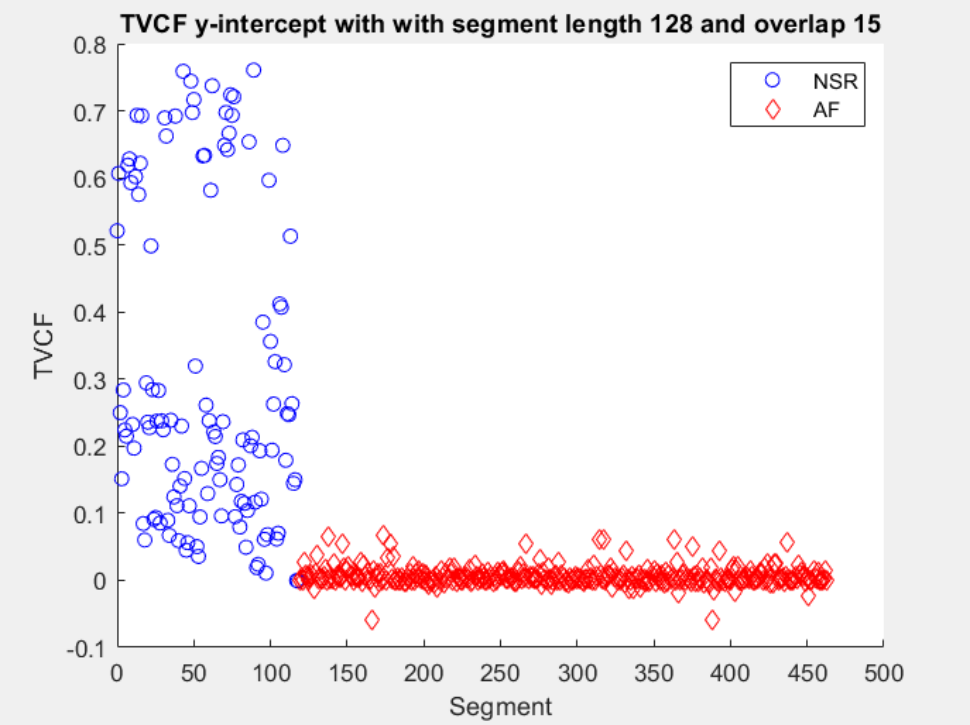


Figure 8.9: y-intercept of TVCF calculated from 0.3 Hz to 0.5 Hz for subject 08455

In this round of testing the y-intercept value was varied from 0.28 to 0.005 and the mean was varied from 0.3 to 0.21. The combination that produced the highest averages was a y-intercept threshold of 0.06 and a mean threshold of 0.21. The average sensitivity was 71.55±32.70 with an average positive predictivity of 82.23±28.01. This did not produce any significant change in the averages compared to the second round of testing. The results are shown in Figure 8.10

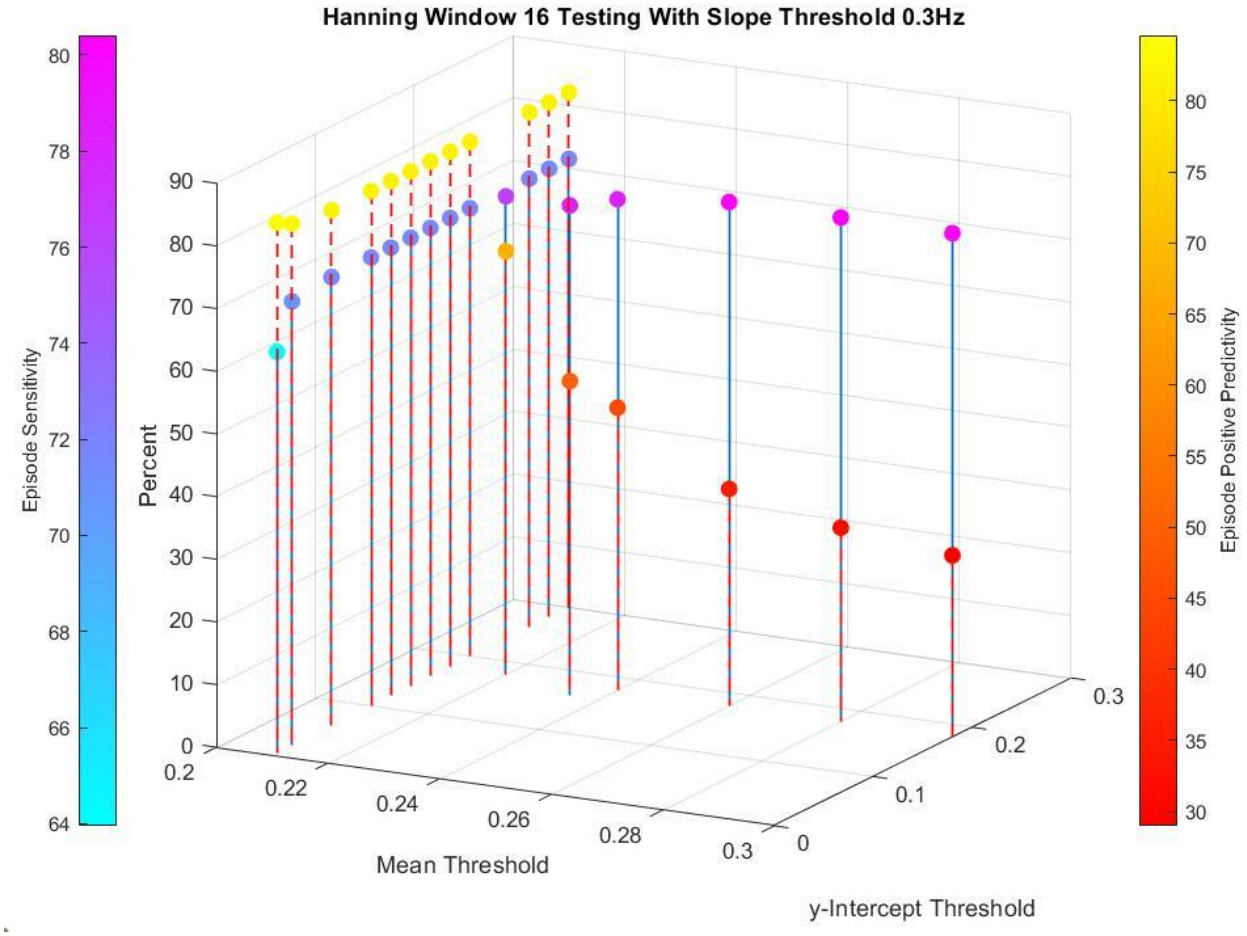


Figure 8.10: Average Sensitivity and Positive Predictivity plot for Hanning Window of 16 and Slope Threshold .3Hz

One general trend was present in all three rounds of testing. When the mean and y-intercept threshold values increased, the sensitivity would also increase, but the positive predictivity would decrease. This is most clearly shown in Figure 8.7. This trend makes sense because sensitivity is measured by the number of episodes detected. The higher threshold values allowed for more of the episodes to be found. Positive predictivity is measured by how often when the algorithm claims a beat is in atrial fibrillation it is truly in atrial fibrillation. Therefore, with the higher threshold values the algorithm falsely says beats are in atrial fibrillation. That is why when the thresholds decrease the positive predictivity increases because the algorithm is more likely to be correct when saying a beat is in atrial fibrillation.

The success of the algorithm on different subjects varied drastically as seen in the high standard deviation values. There were a few subjects that the algorithm correctly classified with

100% sensitivity and positive predictivity. Subjects 06426, 07162, and 08215 for all three test trials had a 100% for both categories when the optimal thresholds identified above were used. This can be attributed to the data following a similar pattern to subject 08455 for which the classification parameters are derived from. This is shown when comparing Figures 8.8 and 8.6. For subject 05191 threshold values used did not correctly classify atrial fibrillation with 0% sensitivity and positive predictivity. This is because the threshold values need to correctly classify atrial fibrillation being higher for this subject than others. The results from the y-intercept vary more than that of other subjects; this is shown when comparing Figure 8.11 and 8.12.

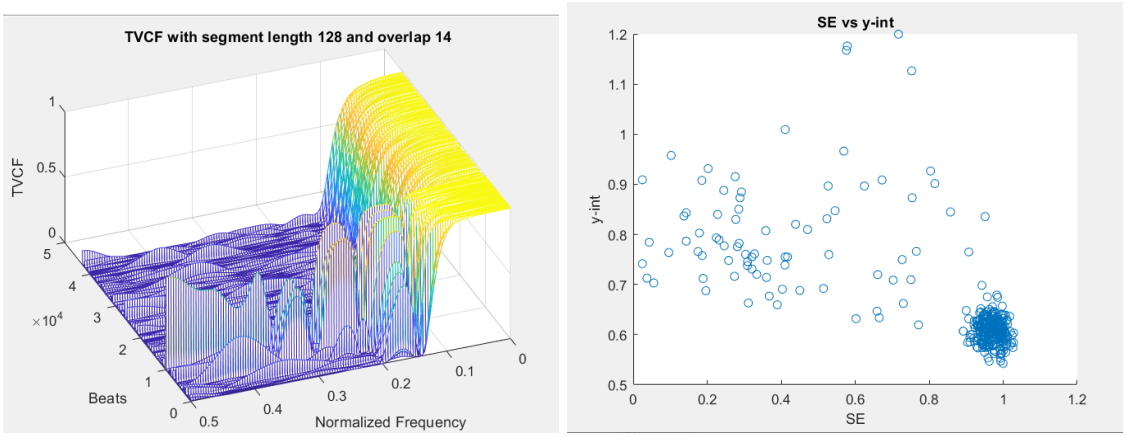


Figure 8.11: Subject 08215 TVCF (left) and y-intercept vs Shannon entropy (right)

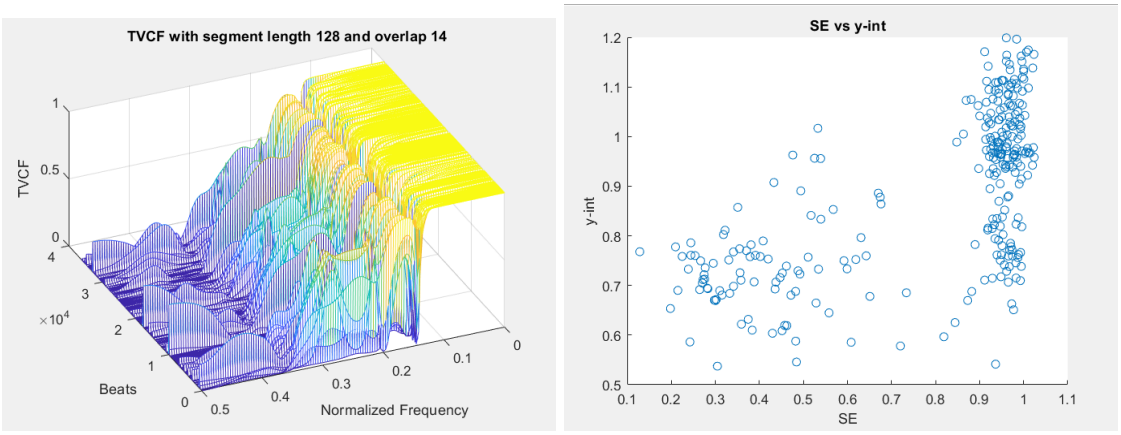


Figure 8.12: Subject 05019 TVCF (left) and y-intercept vs Shannon entropy (right)

With some of the subjects there could be a very high positive predictivity with a relatively low sensitivity. For example, the classification results for subject 08455 were 50% sensitivity and a 95% positive predictivity. The detection algorithm always missed the first episode of AF in the subject because it is very short in comparison to the other AF episode that occurs for the majority of the ECG data. Instances like this, where some episodes of AF are very short, possibly contributed to some of the lower sensitivity percentages.

The accuracy achieved by Lee et al. was not replicated with our atrial fibrillation algorithm. Lee et al.'s algorithm had a sensitivity of 97.41% [1]. The highest average sensitivity achieved using our algorithm was 80.5%. Our algorithm therefore needs significant improvement in order to perform with the same accuracy as Lee et al.

8.4: Discussion

The atrial fibrillation algorithm is able to classify RR-interval segments as normal sinus rhythm or atrial fibrillation using the thresholds described above. The algorithm utilized a RR-interval segment size of 128 and the TVCF function used a hanning window of size 16 and an overlap of 15. The best thresholds for classification were found to be a y-int of 0.59 and a mean of 0.2 with a window size of 16 and an overlap of 15. Testing was conducted over all subjects in the MIT/BIH Atrial Fibrillation database with the exceptions of subjects 07859, 00735, and 03665. Subject 07859 caused an error in the prctile() matlab function due to its size. The subject can be tested on if the ectopic beat remover function is not applied to the data as this

is the function that uses prctile(). For subject 03665, the comment annotations contained “(J” episode annotations. This might have prevented epicmp from producing an accurate report because the algorithm only assigns episode annotations of “(N” and “(AFIB”. For subjects 03665 and 00735 there was information missing from the header files that is present in the rest of the subjects. This missing information is most likely the reason for the issues with these subjects. The report produced by the comparator for this subject, as well as 03665, is blank. The reports are compared in Figure 8.13.

```
AF episode-by-episode comparison results for record 00735
Reference annotator: atr
     Test annotator: test

           Episode sensitivity:    -  (0/0)
 Episode positive predictivity:    -  (0/0)
          Duration sensitivity:    -  (0.000/0.000)
Duration positive predictivity:    -  (0.000/0.000)
Duration of reference episodes:          0.000
     Duration of test episodes:          0.000
```

```
AF episode-by-episode comparison results for record 08455
Reference annotator: atr
     Test annotator: test

           Episode sensitivity:  50% (1/2)
 Episode positive predictivity:  95% (18/19)
          Duration sensitivity:  94% (6:40:57.212/7:04:31.024)
Duration positive predictivity: 100% (6:40:57.212/6:42:34.012)
Duration of reference episodes:   7:04:31.024
     Duration of test episodes:   6:42:34.012
```

Figure 8.13: epicmp() report subject 00735(top) and subject 08455(botton)

The implementation of the epicmp() function from the WFDB toolbox was a challenge. The episode comparator is not a part of the WFDB Physionet MATLAB toolbox and is written only in the C programming language. We tried to compile the function using a C editor but it had too many errors and too much missing information to compile. We then tried to download the WFDB toolbox that contained all of the functions in C. After following the installation instructions we were able to obtain an executable version of the epicmp function. This was then run from MATLAB using the system() command. With this function we were able to compare our results with the annotations provided by Physionet using the ANSI/AAMI EC38:2007 standard. Having access to the standard simplified the testing process significantly because the AF annotations provided by Physionet are not marked on the location of an R-wave. This allowed the results of our algorithm to be off by a number of samples and still be considered correct. The degree to which the samples can be off the truth value is defined in the standard.

Other ways to further optimize the algorithm would be to test the window sizes' effect on classification. Also the investigation of other parameters such as the integral of the TVCF might provide a better indicator of atrial fibrillation. There are many possible ways to further improve this algorithm that future teams can explore. Other methods of classification can be used, such as discriminant analysis, support vector machines, machine learning, and neural networks.

9: Final Design

For the sake of explanation we will divide our ECG system into 4 primary components; electrodes, analog front end, embedded system/software, and algorithms. It is important that each of these components are broken down and examined. This setup can be seen in Figure 9.1.

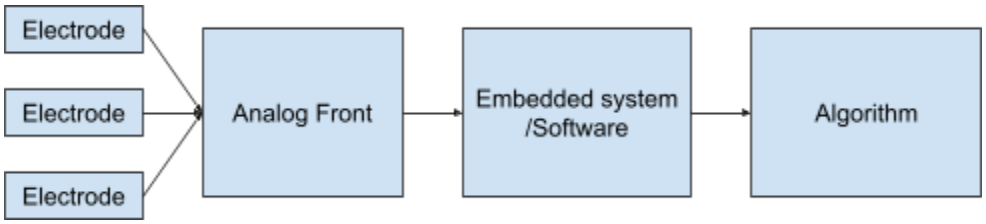


Figure 9.1: Final Design Block Diagram

The first section that we will talk about are the electrodes. For this ECG three electrodes were used, two of the electrodes were used to measure the signal of the heart while the third electrode references their body to the same ground as the electronics allowing a consistent point for measurement. Once the electrodes transduce the signal from the body, it goes through an 80 kΩ resistor and 2 sets of diodes both there to prevent any possible damage to the person or the electronics. The resistor is set up in series with the electrode while the diodes act as a way to limit voltage below 0.7 V, meaning if the voltage in the electrode for any reason goes above 0.7V then the diodes will allow the voltage a direct path to ground avoiding damage to the person or the electronics. This design can be seen in Figure 9.2.

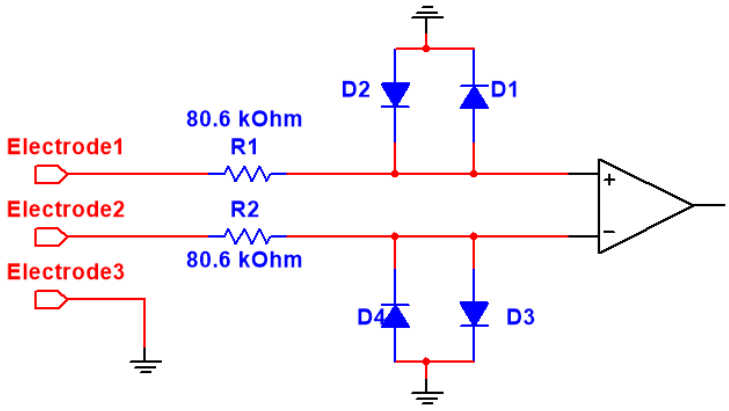


Figure 9.2: Electrode setup

After the signal has passed the resistors and diodes, both of the channels go into a differential amplifier which outputs the difference between the signals and amplifies it 10 times its original amplitude. Next the signal goes to a quad op amp with a highpass and lowpass second order Sallen-Key filter which has a gain of about 30. This leads to an overall gain on the analog front end of around 300 times the original. The last part of the analog front end is the summing amplifier that is still working off the same quad op amp as the filters. This summing amp combines the output from the filters with a roughly 1V DC output from the voltage regulator in order to add an offset to the signal so that the ADC in the Nordic unit can interpret the data. The block diagram design of this front end can be seen in Figure 9.3.

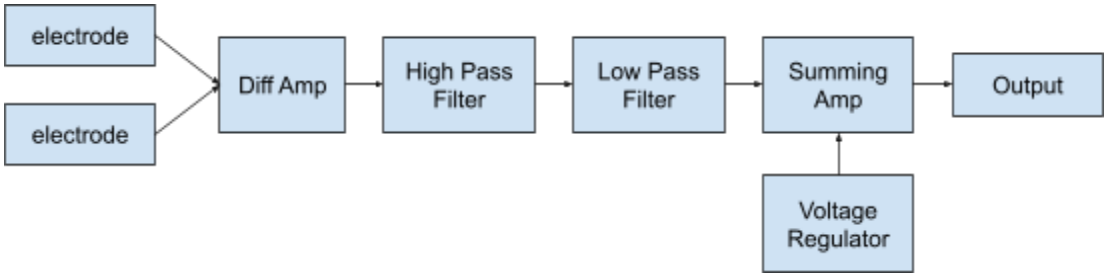


Figure 9.3: Analog Front End Design

Once the ECG is produced at the output of the circuit, it is transmitted through the Microcontroller for processing. The hardware is physically wired to the Nordic Microcontroller and is passed into the onboard ADC. The signal is then transmitted to another Microcontroller configured as the receiver. This receiver MCU is connected to the computer to allow for data to be collected.

The data received and passed into the computer can be recorded and processed. A MATLAB function called *convert2float* converts the raw collected data into float values. The output waveform from the circuit can then be viewed on the computer and the converted data can be applied to the signal processing algorithms. The R-wave and atrial fibrillation detection algorithms discussed in previous sections can be used with the generated data to provide information from the ECG waveform. The R-wave detector used a bandpass filter on the subject data. The data was then squared and a moving average filter applied. Algorithms then used the filtered data to accurately detect the locations of R-waves. The R-wave detection algorithm was able to accurately identify almost all of the R-wave peaks in our recorded ECG signal. The

algorithm did not perform well if there was motion detected or large variation in R-wave magnitude as seen in Figure 9.4.

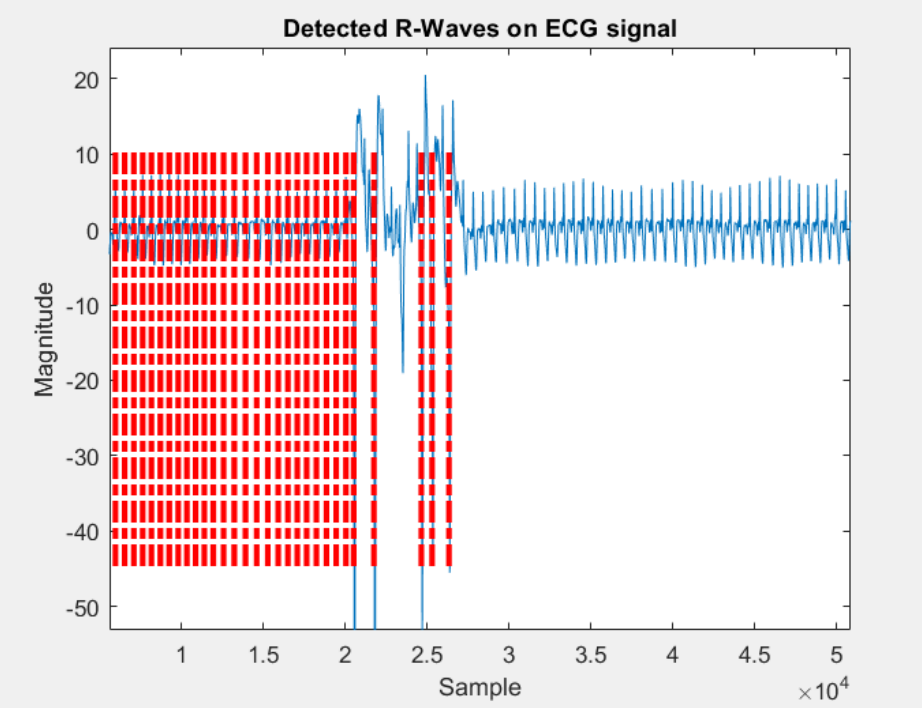


Figure 9.4: ECG waveform with R-Wave detection in red from human testing with motion artifact (1000 Hz sampling rate)

The atrial fibrillation algorithm classifies 128 beat segments as containing atrial fibrillation by calculating the Shannon entropy as well as the y-intercept and mean of the TVCF. Using threshold values defined in chapter 8, the segments are classified as normal sinus rhythm or atrial fibrillation.

With the entire process of generating, transmitting, and processing the ECG signal finalized, testing was conducted on people instead of using a simulator. Testing on humans allowed for real data to be processed and compared to simulations. Figure 9.5 is the resulting plot of data collected from a test subject and Figure 9.6 shows the R-waves found after using the R-wave detection algorithm.

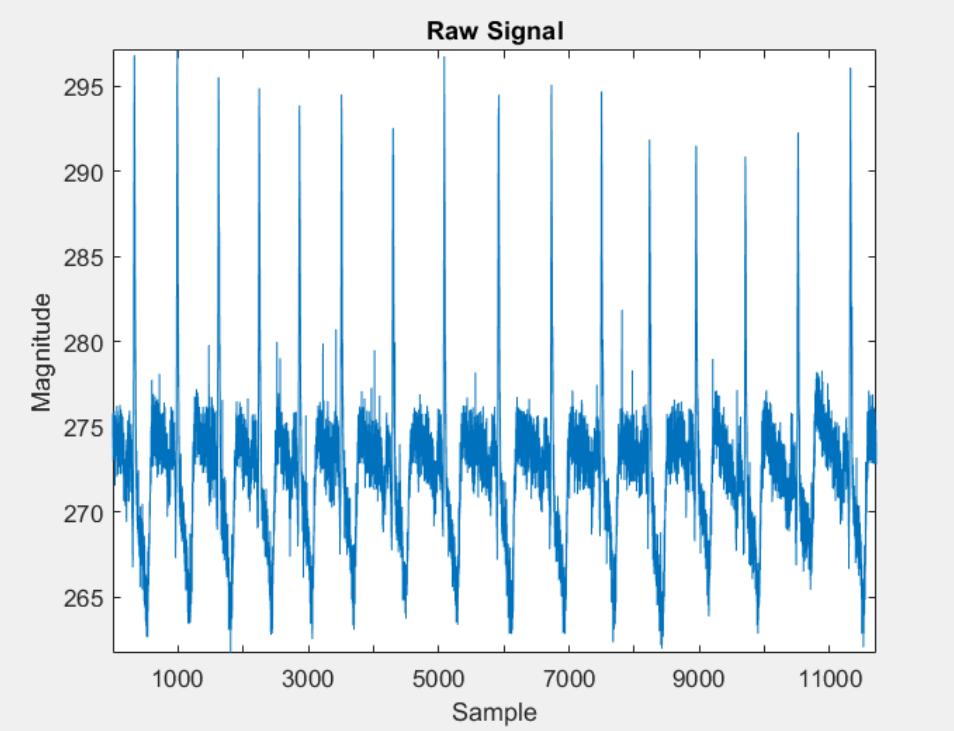


Figure 9.5: ECG Waveform from Human Testing (1000 Hz sampling frequency)

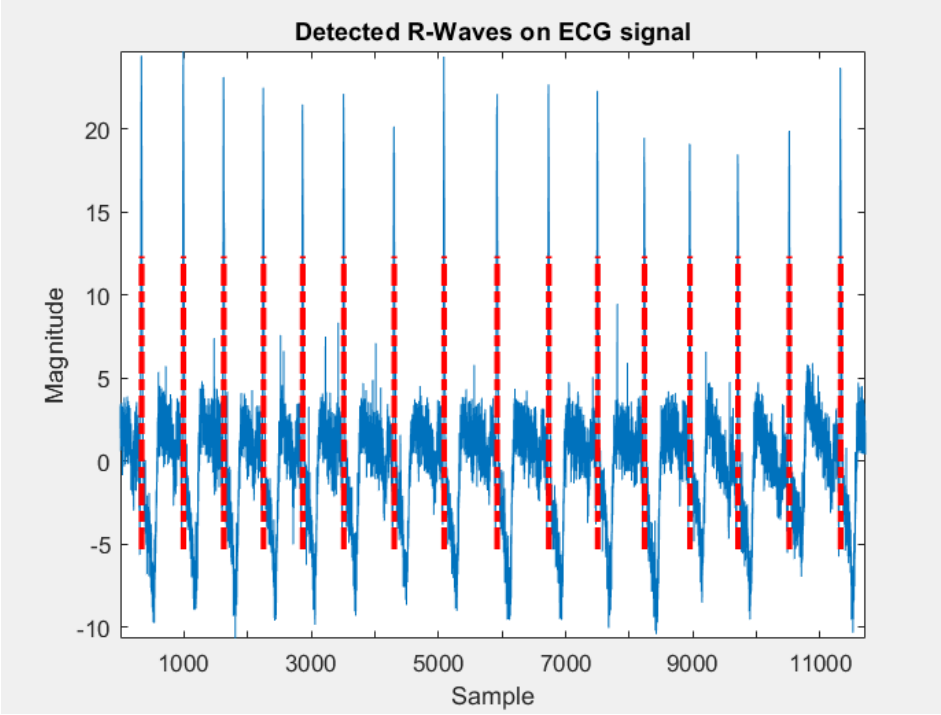


Figure 9.6: R-wave Detection Results (100 Hz sampling frequency)

The recording in Figures 9.5 and 9.6 were very short and therefore could not be tested with the atrial fibrillation detection algorithm due to the recording containing fewer than 128 beats. Longer recordings were obtained with the signal data shown in Figure 9.7. This recording contained the least amount of motion artifact and therefore responded the best to the R-wave detection algorithm. The algorithm only missed three beats in the entire recording. The missed beats are displayed in Figure 9.8. The beats were missed due to the amplitude of the R-waves being smaller than the rest of the surrounding peaks.

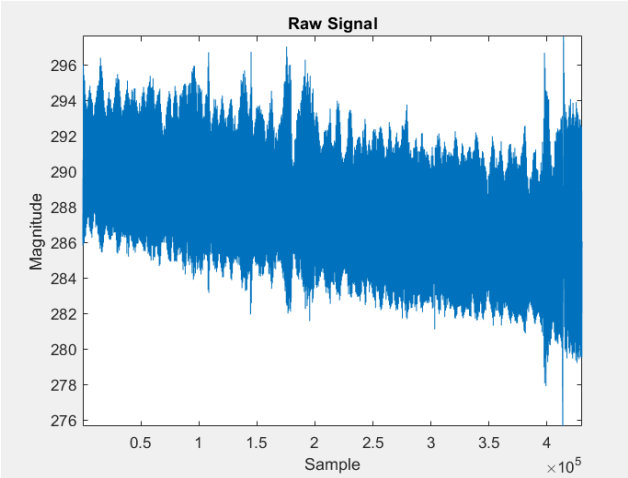


Figure 9.7: Long ECG Waveform from Human Testing (1000 Hz sampling frequency)

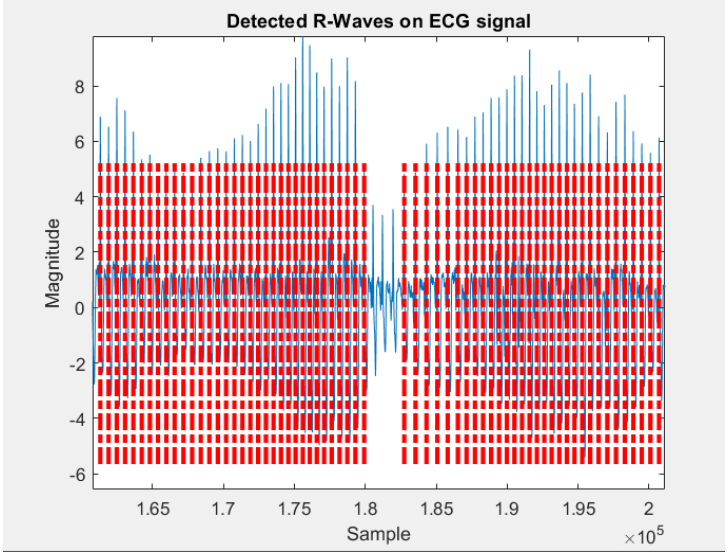


Figure 9.8: Long ECG Waveform from Human Testing with Missed R-Waves (detected waves indicated with a vertical red line) 1000Hz sampling frequency

The atrial fibrillation algorithm was then applied to the difference between the R-waves detected using the code in Appendix G. The missed R-waves were added back in to improve the performance of the AF algorithm. The TVCF of the recorded data is shown in Figure 9.9.

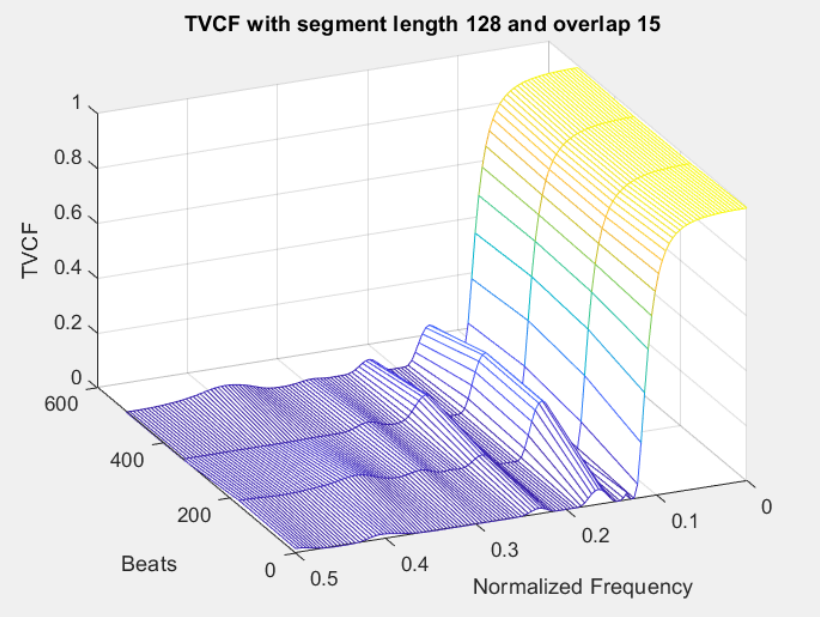


Figure 9.9: TVCF for Human Testing Recording

The algorithm detected only normal sinus rhythm which was expected since the test subject is in good health and does not have any known heart conditions.

9.1: Economic Impact

The cost of a single analog front comes primarily from the instrumentation amplifier and the voltage regulator. The Instrumentation amp, the LT1167, costs $10.22 for each component and each front end only needs one. The second most expensive component on the board is the TPS73615 voltage regulator with a price of $6.88, this component also only appears once on the board. The last of the ICs on the front end is the LM324 op-amp with a cost of only $0.44 and the fact that only one is needed does not contribute significantly to the overall cost. The largest cost in the overall product is the processor Nordic nrf52840-DK with a per unit cost of $50.00. To record a signal only one is needed but if that signal is to be transmitted three will be needed. The overall cost is made with all of the previous components as well as a handful of miscellaneous resistors and capacitors that have a negligible impact on the overall cost. All together the price comes out to roughly $170.00 this does not include any manufacturing costs.

This cost of $170 is higher than similar wireless ecg monitoring products. While other products on the market are occasionally closer to $100 the price of our units could decrease significantly if we were to find a solution other than having to use the full development kit which is the largest portion of our cost.

9.2: Environmental Impact

Every component utilized in measuring the ECG signal can be reused, lessening the negative environmental impact as opposed to a single-use device. Furthermore, the construction of the circuit relies solely on small, low power components, thus reducing the risk of damaging the local environment. Given the small size of the design, potential disposal of our product will produce minimal waste product.

9.3: Societal Influence

This project will primarily be for specific use by doctors and patients with the goal of more mobile heart rate monitoring. The target audience would be people who have a problem with atrial fibrillation. This means that the product will likely not have significant immediate societal impact. This product in the future could help with the monitoring and diagnosis of patients with atrial fibrillation.

9.4: Political Ramifications

A political ramification of this project may be a discussion on patient privacy when storing and sending data collected from the patient. Currently, data collected is sent only to the members of the project team for analysis and is kept confidential. Future improvements to the system could involve patient data being sent directly to a doctor or to an app on the patient's phone. Sending patient data could raise concerns regarding who has access to the data, where it is stored, and how long it will be saved. Patients should be aware of who has access to their private information and data. Doctors or other users of the device should also clearly communicate to their patients how the collected data will be handled.

9.5: Ethical Concerns

Ethical concerns were mostly considered in regard to human testing of our final design. All testing protocols were approved by the Institutional Review Board. Testing protocols followed guidelines set forth by the Belmont Report and 45 CFR 46 also known as the Common Rule. The consent form can be found in appendix B. With the wireless transmission of the data and its storage on computers there is always the change of the information being accessed by unauthorized individuals. In order to maintain the confidentiality of users' medical information all files are coded so that it can not be easily linked back to users.

9.6: Health and Safety Issues

The goal of the wireless ECG device is to improve peoples' health and safety by allowing them to monitor their ECG and be alerted if they experienced an episode of atrial fibrillation. The device itself was designed with safety in mind. The electronic components are capable of transmitting data wirelessly. The maximum current that can be delivered to the body from the device is within a safe limit defined by ANSI/AAMI ES1—1993. The housing for the electronics was designed with no sharp edges with the wearer's comfort in mind.

9.7: Manufacturability

The circuit for the ECG hardware was implemented on a Printed Circuit Board (PCB) with electronic components soldered onto it. This board can be reprinted and the components can usually be soldered by the company that prints the board. The components can also be purchased separately. The housing for the system was 3D printed and can be cheaply remade with access to a 3D printer. Electrodes can also be easily purchased with wires to connect to the circuit. Finally, a Nordic microcontroller is used to control the device. The total cost of all the components required to manufacture the system is under $200 and could be ordered and 3D printed in one to two weeks.

9.8: Sustainability

Sustainability was largely not considered in the design of this project. The device does not utilize renewable energy sources for power. However, the reusability of the device does produce a more sustainable product. Manufacturing of our product occurred at a semiconductor company, a PCB fabrication manufacturer, and additional work within the WPI laboratories. The product is not intended to be mass produced, therefore, sustainability issues concerning mass production are largely subverted.

10: Discussion

The overall testing of the entire system was successful. The analog front end acquired, filtered, and amplified the signal obtained from the test subject. The MCU then used an AD converter and wirelessly transmitted the digital data to a base unit. The data was then saved and passed through the R-wave detection algorithm. As discussed in the previous chapter, motion artifact caused the detection algorithm to miss the majority of beats in some recordings. This is due to the adaptive thresholding in the algorithm being adjusted too high to capture the majority of the R-wave peaks. This was also an issue encountered when testing on the MIT/BIH database.

The atrial fibrillation detection algorithm initially detected atrial fibrillation in the ECG recording where there should be none. The first approach to solving the issue was to include the missed beats when calculating R-wave differences. This helped but was not the main source of the issue. When looking at the parameter values they were significantly different from the average values obtained for normal sinus rhythm when testing on the MIT/BIH atrial fibrillation database. We suspected that the difference in sampling rates between the database and our recording might have contributed to the issue. Therefore, we downsampled from 1000 samples per second to 250 samples per seconds by dividing the R-wave sample locations by four. This improved the algorithm results but not enough for them to be considered accurate. The magnitude of the R-wave differences were still notably different so we divided the sample locations by eight instead of four. This resulted in the R-wave difference being the same magnitude as those in the MIT/BIH database. With this adjustment the algorithm accurately classified the entire recording as being normal sinus rhythm. One way to ensure that everything is uniform is to convert the sample values to time before taking the difference. This can be implemented in future iterations of the atrial fibrillation algorithm.

Our project is just a start and there are a lot of future directions our project can go. On the hardware side, we can incorporate an LCD or a seven-segment display that can display the heart rate in real time while it is mounted on the body. The current analog system is powered using a bipolar supply, but a unipolar supply would allow the hardware to be powered using only one battery. The circuit could also be modified to use only two input electrodes instead of the three electrode input that is now required. Using the system on a chip (SoC) version of the Microcontroller instead of the full development board would allow the entire device to be

significantly smaller. For the R-wave algorithm, it can be further optimized by adjusting thresholds so that special cases such as record 08219, which was described earlier, can detect R-waves despite the amplitude shift in the data. There is also the case of record 07162 where changing filter parameters and trying both channels of the ECG array did not improve the QRS sensitivity and QRS positive predictivity, so we would need to further examine how to produce better results. There are also probably better ways to automate testing the data so that we don't have to manually change the .dat file that we want to analyze, but that is something that future iterations of the detector can add. For the atrial fibrillation detection algorithm, alternate methods for categorizing AF episodes can be investigated in order to increase the sensitivity and positive predictivity of the algorithm. The algorithm should also be tested on the MIT/BIH Normal Sinus Rhythm database to ensure that the algorithm does not identify false AF episodes. This might also aid in better understanding the properties and trends present in normal sinus rhythm.

As a team, we were able to learn a lot from each other and our advisor about what ECG circuits need, how to reduce noise, and concepts such as PCB layout. We first learned about the concept of the ECG signal and the wave that is produced. The method of extracting the waveform from the electrical signals produced from the body was thoroughly investigated. We have also been learning how to understand and modify the embedded code for the MCU. For the algorithms, we learned how to use the WFDB toolbox for MATLAB and the methodology for the algorithms. We had also never heard of the bxb comparator for R-wave detection until our advisor recommended we try it, and it was interesting to see how accurate our algorithm really was, since we were initially relying on manual verification. We also struggled when it came to using the episode comparator provided by Physionet because it was not part of the WFDB toolbox for MATLAB. We had to be creative when it came to calling the function through MATLAB. We ended up using the system command to run the executable version of the function. Overall, Physionet was able to give us a visual understanding of what the signals will look like for normal sinus rhythm, arrhythmias, and atrial fibrillation, which will help when we collect our own data and analyze it. It also allowed us to test the limits of our algorithm so that we knew how to improve it.

Some challenges we faced included the selection of components that were low power but still able to perform as needed for our application. Noise throughout the system, including the hardware and wireless transmission, was a frequent problem. Both analog and digital filtering

was used to reduce noise when possible. Another challenge experienced was understanding how to install and use the functions in the WFDB toolbox. Additionally, while testing the records using our comparator() function, we realized that MATLAB would freeze often, which slowed down our testing a lot. We were able to bypass this hindrance by strategically exiting out of the MATLAB program once our test file was written to and we were still able to produce an accurate bxb report file. By using the executable version of the episode comparator the MATLAB program ran smoothly and efficiently. It might be beneficial to run the bxb function using its executable instead of its MATLAB function to prevent the program from freezing.

If we had more time, the goals that we would like to be able to reach would be to change the peripheral nodes to one that could handle only two electrodes, making it easier to conceal. To accomplish this we would need to modify the circuit by changing a few components in front of the instrumentation amplifier. Making the system smaller would also create a more convenient product for the user. Figure 10.1 shows how, by using the nRF52840 Dongle or the nRF52840 system-on-chip, future teams could reduce the size of the boards and housing. The proposed dimensions for the housing are based on increasing the PCB size to accommodate the additional components while removing the current microcontroller development board. If all components are on a single PCB board the final design could be significantly smaller than our estimates.

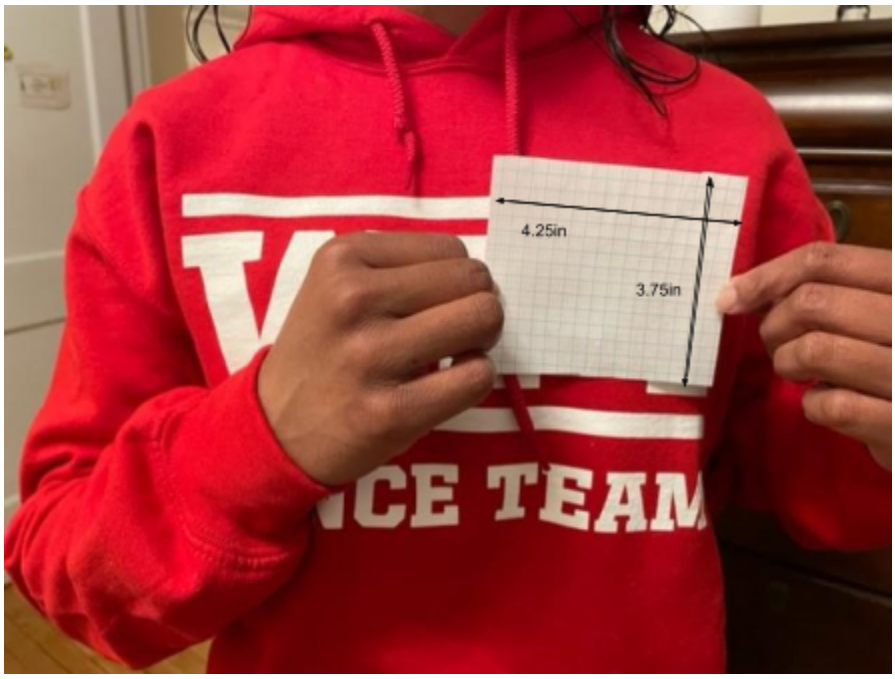


Figure 10.1: Proposed size of future housing with dimensions 4.5inx3.5inx1.5in

A stretch goal for the project is to process the data in real time. This task is made difficult due to slight time delays between the different electrodes and the MCU. The scope of this project will likely include post processing that could take up to a minute to display results.

11: Conclusions and Recommendations

This project presented the team with many challenges that required inventive ideas to solve. . The team was able to create a final product that met the requirements laid out in our revised client statement. Our final design is an ECG system that operates off of a battery and transmits the data from the patient where it is recorded to a base module. The data is stored on a computer where we can run our algorithms to detect heart rate and atrial fibrillation.

One way this project can be improved is by using DIP components on the PCB. While the benefits of surface mount components is clear due to their small size, the increased difficulty in working with them and making repairs does not make it worth using them over more standard pin ICs for this type of academic project. In the lab we did not have access to a good enough soldering iron to work well with surface mount components leading to issues later in the development process.

We would also recommend the use of a solely unipolar powered circuitry. While we were unable to find the necessary parts, in the future we would suggest that this be a more significant priority as it simplifies the system greatly and makes the power transmission of the whole project far easier.

We would recommend looking into how to better power the Microcontrollers. Early in the project the team found that the battery on the Microcontroller was able to power the board, however once the code was uploaded this was no longer true and we needed to pursue other options for powering it. Our recommendation would be to find the cause of this issue or change the power of the overall system to meet the new power needs of the Microcontroller.

There are a few recommendations we have for the two algorithms.In general it would be easier for the C program version of the WFDB toolbox to be installed first. Once it is installed check to make sure that all of the function executable files are present. This will provide all of the functions in the WFDB toolbox unlike the MATLAB WFDB toolbox which only has a portion of the full toolbox. The executable files can be run using the system() function in matlab. Running the functions this way might also avoid issues we encountered using bxb(), where the program would not stop running. For the R-wave detection algorithm specifically we suggest dividing the recordings into smaller sections and then running the section individually. This might prevent large sections of R-waves being missed due to the threshold being too high.

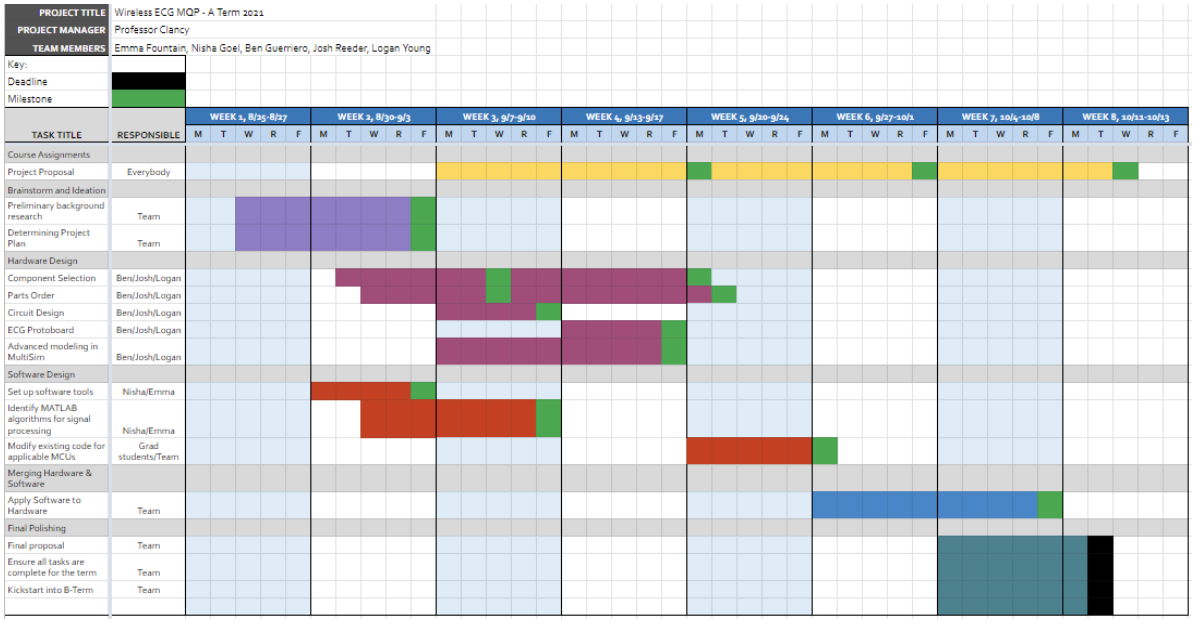
Therefore, motion artifacts or other noise that occurs will only affect the detection on a small section instead of the whole recording. For the atrial fibrillation detection algorithm we suggest pursuing other methods for classification such as discriminant analysis and machine learning. If the difference of RR waves is used in the future, convert samples into time using the sampling frequency so that regardless of the data being tested the differences are calculated using the same scale.

Over the course of three terms we were able to meet the requirements defined in our revised client statement. We created a three electrode ECG analog circuit that interfaces with a microcontroller to wirelessly transmit the ECG data. The size of the analog circuit was reduced through the creation of a PCB equivalent circuit. Which met our goal of minimizing the size of the device. A housing was developed to enclose all of the electrical components for safety during testing. We also developed an R-wave detection algorithm that was able to accurately detect all but three R waves from our recorded ECG signal. The atrial fibrillation algorithm correctly only detected normal sinus in the same ECG recording. Overall, we created one functional node that could be replicated in the future to record and transmit ECG data from multiple nodes. We also created algorithms that are able to process that transmitted data.

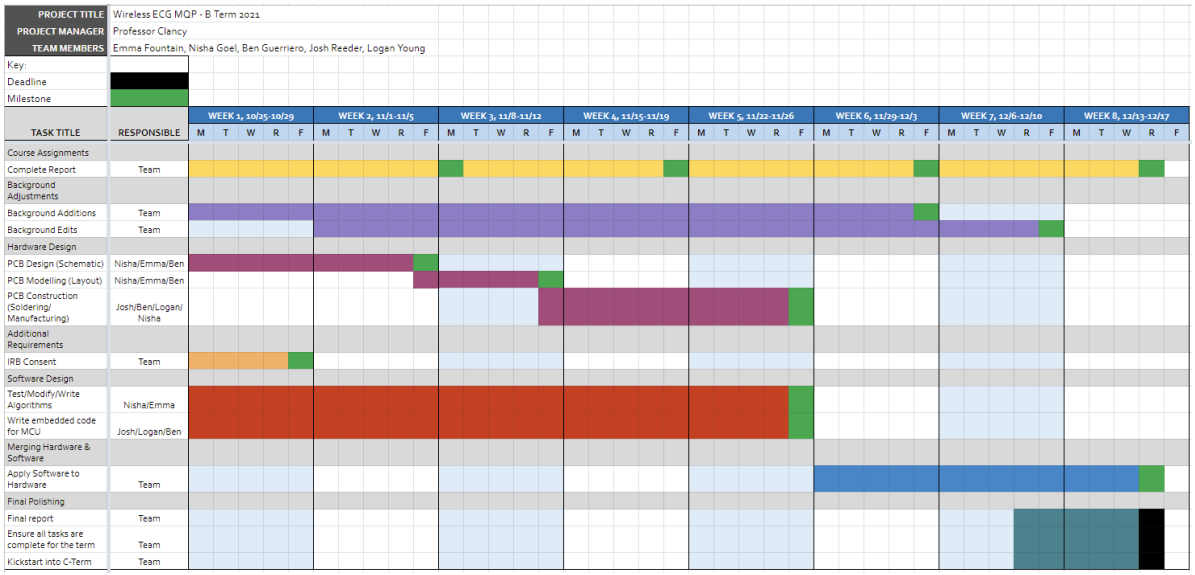
Appendix

Appendix A: Gantt Charts

A.1: A-Term



A.2: B-Term



A.3: C-Term

PROJECT TITLE	Wireless ECG MQP - C Term 2021
PROJECT MANAGER	Professor Clancy
TEAM MEMBERS	Emma Fountain, Ben Guerriero, Josh Reeder, Logan Young, Nisha Goel (in spirit)
Key:	
Deadline	
Milestone	

TASK TITLE	RESPONSIBLE	WEEK 1, 1/12-1/14					WEEK 2, 1/17-1/21					WEEK 3, 1/24-1/28					WEEK 4, 1/31-2/4					WEEK 5, 2/7-2/11					WEEK 6, 2/14-2/18					WEEK 7, 2/21-2/25					WEEK 8, 2/28-3/4				
		M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
Course Assignments																																									
Project Report	Team																																								
Hardware Design																																								
Order PCB	Team																																							
Software Design																																								
Complete A-Fib algorithm	Emma																																							
Modify/Test A-Fib algorithm																																								
Modify MCU code	Team																																							
Merging Hardware & Software																																								
Apply Software to Hardware (Testing)	Team																																							
Final Polishing																																								
Design Housing (3D print... a box)	Team																																							
Ensure all tasks are complete for the term	Team																																							

Appendix B: IRB Informed Consent Document

Informed Consent Agreement for Participation in a Research Study

Investigators: Emma Fountain, Benjamin Guerriero, Joshua Reeder, Logan Young, Professor Edward A. Clancy

Contact Information: gr-ECG-MQP@wpi.edu (student team) or ted@wpi.edu (faculty advisor)

Title of Research Study: Developing a Multi-Lead Wireless Electrocardiogram (ECG)

Sponsor: None

Introduction: You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation

Purpose of the study: This study will contribute to the development of a wireless ECG system (also known as an EKG system). This system will allow the user to monitor their heart rate and notify them if they are experiencing atrial fibrillation or other abnormal beating of their heart. This device will allow the user to monitor their heart rhythm and could possibly be utilized by medical professionals in the future. The goal of this study is to validate that the ECG system works to record, transmit, and process your heart data.

Procedures to be followed: This study will consist of one session of approximately 30 minutes duration. Testing will take place on WPI's campus in the Atwater Kent building. Upon arrival you will be asked to fill out a short questionnaire. You are not required to provide any information you are uncomfortable sharing. We will ask about your: sex, age, height, weight, and any diagnosed heart conditions.

We ask that participants wear a face mask and clothing that easily allows access to the upper chest and back. Women are encouraged to wear a sports bra or bathing suit top underneath a tank top (or T-shirt).. Before the electrodes are placed on the body, the skin at the application sites will be prepared using an alcohol wipe.

We will secure to your upper chest and back up to three of the wireless ECG systems that we are developing. Each will use three ECG electrodes. We may use ace bandage or other flexible wraps

around your trunk and tape to hold the apparatus secure to your body. Each ECG device will be light-weight and small enough to be easily supported (e.g., smaller than the size of your fist).

You will be asked to sit in a chair and remain still during the recording, breathing normally and not speaking. The device(s) will record 15 minutes of data. Once data collection is complete the apparatus and electrodes will be removed.

Risks to study participants: There are no risks to participants. Some subjects experience discomfort when the electrodes are removed, as they can pull on body hair and the skin.

Benefits to research participants and others: There are no benefits to the participants or others.

Record keeping and confidentiality: For this study you will be assigned a subject ID number upon the signing of this document. This document will be the only record connecting your subject ID number to your name and will be stored privately. All electronic records will only refer to your data by your subject ID. Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to confidential data that identify you by name. Any publication or presentation of the data will not identify you.

Compensation or treatment in the event of injury: WPI assumes no responsibility to pay for any injuries that you might receive as a result of participating in this research study. No funds have been set aside for payments or other forms of compensation (such as for lost wages, lost time, or discomfort). If you suffer a physical injury as a result of your participation in this study, you may choose to seek medical care in the same way as you would normally. If your insurance does not cover the cost, then you may be responsible for this cost. However, you do not give up any of your legal rights by signing this consent form.

Incidental findings: This study does not provide medical care. None of the investigators is a health care provider nor will your recorded ECG data be routinely reviewed by a healthcare provider. If you or the investigators note any possible abnormalities on your ECG, you should speak with your own health care provider about any concerns.

Data Reuse and Contribution of Your Data to a Public Data Archive: The data from this experiment will also be contributed to publicly available databases and/or reused by the study investigators in future research. The purpose of this data reuse is to share your data with other researchers (or reuse the data ourselves) to make further advances in medicine, science and teaching. Your data could be used for many different purposes. Most researchers will gain access to your data over the Internet. Before contributing your data, all information that identifies you as a subject in this experiment (including your name) will be coded. The only way to relate the code to your identifying information is by a “key” that the study investigators will maintain private. We will never reveal your

identity, unless required to do so by law. The public database will not provide any direct access to your identity.

Cost/Payment: Healthy subjects will not be paid for their participation in this study. Subjects with a known diagnosis of atrial fibrillation will be paid $20 for completion of this study.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact: MQP team, gr-ECG-MQP@wpi.edu; Edward (Ted) A. Clancy, Professor (WPI), Tel. 508-831-5778, Email: ted@wpi.edu; Ruth McKeogh, IRB Manager, Tel. 508 831- 6699, Email: irb@wpi.edu; and Gabriel Johnson, Human Protection Administrator, Tel. 508-831-4989, Email: gjohnson@wpi.edu

Your participation in this research is voluntary. Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

_____ Date: _____
Study Participant Signature

_____
Study Participant Name (Please print)

_____ Date: _____
Signature of Person who explained this study

Appendix C: R-Wave Detection Algorithm Results

C.1: Pan-Tompkins Results for the Arrhythmia Database [25]

RESULTS OF EVALUATING THE REAL-TIME *QRS* DETECTION ALGORITHM USING THE MIT/BIH DATABASE

Tape (No.)	Total (No. Beats)	FP (Beats)	FN (Beats)	Failed Detection (Beats)	Failed Detection (%)
100	2273	0	0	0	0
101	1865	5	3	8	0.43
102	2187	0	0	0	0
103	2084	0	0	0	0
104	2230	1	0	1	0.04
105	2572	67	22	89	3.46
106	2027	5	2	7	0.05
107	2137	0	2	2	0.09
108	1763	199	22	221	12.54
109	2532	0	1	1	0.04
111	2124	1	0	1	0.05
112	2539	0	1	1	0.04
113	1795	0	0	0	0
114	1879	3	17	20	1.06
115	1953	0	0	0	0
116	2412	3	22	25	1.04
117	1535	1	1	2	0.13
118	2275	1	0	1	0.04
119	1987	1	0	1	0.05
121	1863	4	7	11	0.59
122	2476	1	1	2	0.08
123	1518	0	0	0	0
124	1619	0	0	0	0
200	2601	6	3	9	0.35
201	1963	0	10	10	0.51
202	2136	0	4	4	0.19
203	2982	53	30	83	2.78
205	2656	0	2	2	0.08
207	1862	4	4	8	0.43
208	2956	4	14	18	0.60
209	3004	3	0	3	0.10
210	2647	2	8	10	0.38
212	2748	0	0	0	0
213	3251	1	2	3	0.09
214	2262	2	4	6	0.26
215	3363	0	1	1	0.03
217	2208	4	6	10	0.45
219	2154	0	0	0	0
220	2048	0	0	0	0
221	2427	2	0	0	0.08
222	2484	101	81	182	7.33
223	2605	1	0	1	0.04
228	2053	25	5	30	1.46
230	2256	1	0	1	0.04
231	1886	0	0	0	0
232	1780	6	1	7	0.39
233	3079	0	1	1	0.03
234	2753	0	0	0	0
48 patients	116 137	507	277	784	0.675

C.2: Arrhythmia Database

Filter Parameters:

```
[b, a] = butter(2, [5 15]*2/fs, 'bandpass');  % 5-15 Hz 4th-order Bandpass Filter
[b,a] = butter(2,5*2/fs,'low');  % 2nd-order Lowpass Filter with 5 Hz cutoff freq
```

Record Number	QRS sensitivity (%)	QRS positive predictivity (%)
100	100	100
101	100	99.87
102	85.28	100
103	99.65	100
104	27.63	100
105	98.65	98.88
106	99.29	99.94
107	97.53	100
108	99.19	94.65
109	99.76	100
111	97.75	99.88
112	100	100
113	99.93	100
114	44.95	97.43
115	100	100
116	98.51	99.9
117	100	100
118	100	100
119	100	100
121	99.87	100
122	100	100
123	100	100

124	98.76	100
200	99.82	100
201	95.92	100
202	99.2	100
203	86.58	99.63
205	99.77	100
207	99.75	99.87
208	99.1	99.88
209	100	99.92
210	96.05	99.86
212	100	100
213	99.74	100
214	100	100
215	99.82	100
217	98.64	99.95
219	99.66	100
220	100	100
221	99.65	100
222	90.74	100
223	99.45	100
228	14.21	100
230	100	100
231	100	100
232	99.93	100
233	99.65	100
234	99.91	100

C.3: Atrial Fibrillation Database

Filter Parameters:

```
[b, a] = butter(2, [5 15]*2/fs, 'bandpass');  % 5-15 Hz 4th-order Bandpass Filter
[b,a] = butter(2,5*2/fs,'low');  % 2nd-order Lowpass Filter with 5 Hz cutoff freq
```

Record Number	QRS sensitivity (%)	QRS positive predictivity (%)
08455	99.23	99.72
08434	99.8	97.63
08405	98.86	97.08
08378	98.63	98.24
08219	18.72	97.86
08215	99.71	97.78
07910	99.98	96.75
07879	99.17	97.15
07859	88.2	97.04
07162	31.42	33.05
06995	99.89	97.35
06453	99.89	99.9
06426	92.52	97.62
05261	99.44	97.74
05121	94.18	97.69
05091	93.39	96.78
04936	99.8	97.28
04908	99.9	97.75
04746	6.62	98.9
04126	89.08	96.66
04048	99.68	99.68
04043	99.66	97.72

04015	9.52	88.88

C.4: Arrhythmia Database- Changed Parameters

Filter Parameters:

```
[b, a] = butter(2, [5 12]*2/fs, 'bandpass');  %changed the higher freq from 15 to 12
[b,a] = butter(1,5*2/fs,'low');  % changed from 2nd to 1st order LPF
```

Record Number	QRS sensitivity (%)	QRS positive predictivity (%)
100	100	100
101	100	99.87
102	93.03	100
103	99.65	100
104	87.29	100
105	98.38	98.42
106	99.17	100
107	97.48	100
108	98.99	91.51
109	99.76	100
111	95.5	99.88
112	100	100
113	99.93	100
114	40.59	92.34
115	100	100
116	98.22	99.85
117	100	100
118	100	100
119	100	100
121	99.87	100

122	100	100
123	100	100
124	98.9	100
200	99.72	100
201	97.17	99.8
202	99.2	100
203	84.89	99.67
205	99.64	100
207	99.69	99.87
208	99.22	99.83
209	100	99.84
210	96.55	99.95
212	100	100
213	99.7	100
214	100	100
215	99.36	100
217	97.45	99.94
219	99.66	100
220	100	100
221	99.75	100
222	89.41	99.63
223	99.27	100
228	14.09	100
230	100	100
231	100	100
232	99.93	99.93

233	99.65	100
234	99.96	100

C.5: Atrial Fibrillation Database- Changed Parameters

Filter Parameters:

```
[b, a] = butter(2, [5 12]*2/fs, 'bandpass');  %changed the higher freq from 15 to 12
[b,a] = butter(1,5*2/fs,'low');  % changed from 2nd to 1st order LPF
```

Record Number	QRS sensitivity (%)	QRS positive predictivity (%)
08455	99.19	99.66
08434	99.79	97.64
08405	98.7	97.07
08378	97.4	98.21
08219	20.16	96.99
08215	99.7	97.78
07910	99.97	96.73
07879	67.53	97.1
07859	82.04	97.04
07162	30.93	33.13
06995	98.22	97.09
06453	99.87	99.91
06426	91.81	97.65
05261	99.38	97.66
05121	94.2	97.68
05091	93.46	96.78
04936	99.79	97.29
04908	97.99	97.71
04746	27.25	99.69

04126	68.84	95.49
04048	99.57	99.69
04043	99.54	97.67
04015	8.66	96.32

C.6: Arrhythmia Database- Channel 2

Using the Original Filter Parameters, but testing Channel 2 instead of Channel 1:

```
[b, a] = butter(2, [5 15]*2/fs, 'bandpass');  % 5-15 Hz 4th-order Bandpass Filter
[b,a] = butter(2,5*2/fs,'low');  % 2nd-order Lowpass Filter with 5 Hz cutoff freq
```

Record Number	QRS sensitivity (%)	QRS positive predictivity (%)
100	99.84	100
101	0.13	50
102	99.84	99.95
103	59.4	93.45
104	98.49	99.56
105	79.26	94.16
106	17.33	96.39
107	97.53	100
108	96.96	98.02
109	99.29	100
111	100	100
112	97.44	99.76
113	99.93	100
114	99.88	100
115	76.18	99.36
116	99.45	100
117	100	100

118	99.95	100
119	99.82	99.94
121	99.81	100
122	99.76	100
123	100	100
124	94.37	100
200	1.66	46.75
201	94.81	100
202	99.79	100
203	87.59	97.53
205	43.93	99.38
207	99.62	99.94
208	46.53	98.52
209	30.81	98.35
210	90.65	99.6
212	65.25	100
213	99.85	100
214	96.27	99.83
215	99.71	100
217	98.1	99.78
219	99.83	100
220	100	100
221	15.69	100
222	99.67	100
223	97.91	100
228	99.82	99.82

230	100	100
231	76.37	100
232	99.73	99.73
233	98.01	100
234	99.52	100

C.7: Atrial Fibrillation Database- Channel 2

Using the Original Filter Parameters, but testing Channel 2 instead of Channel 1:

```
[b, a] = butter(2, [5 15]*2/fs, 'bandpass');  % 5-15 Hz 4th-order Bandpass Filter
[b,a] = butter(2,5*2/fs,'low');  % 2nd-order Lowpass Filter with 5 Hz cutoff freq
```

Record Number	QRS sensitivity (%)	QRS positive predictivity (%)
08455	99.16	99.78
08434	99.62	97.78
08405	99.47	97.26
08378	98.07	98.19
08219	99.65	97.6
08215	99.79	97.78
07910	99.8	96.41
07879	99.88	97.52
07859	24.37	97.92
07162	33.8	33.17
06995	5.15	85.06
06453	99.55	99.93
06426	98.91	97.97
05261	99.13	97.54
05121	99.45	97.77
05091	99.79	97.46

04936	99.76	97.2
04908	99.09	97.45
04746	99.99	97.66
04126	59.7	95.27
04048	99.25	99.6
04043	89.94	97.51
04015	98.23	97.56

C.8: Arrhythmia Database- Lowpass Cutoff Frequency Changed

Using the Original Filter Parameters, but testing Channel 2 instead of Channel 1:

```
[b, a] = butter(2, [5 12]*2/fs, 'bandpass');  % changed from 5-15 Hz to 5-12 Hz
4th-order Bandpass Filter
[b,a] = butter(1,6*2/fs,'low');  % changed to 1st-order Lowpass Filter with 6 Hz
cutoff freq
```

Record Number	QRS sensitivity (%)	QRS positive predictivity (%)
100	100	100
101	100	99.87
102	91.76	100
103	99.65	100
104	89.66	100
105	98.65	98.29
106	99.17	99.94
107	97.48	100
108	99.05	91.45
109	99.76	100
111	95.5	99.88
112	100	100
113	99.93	100

114	40.71	92.1
115	100	100
116	98.22	99.85
117	100	100
118	100	100
119	100	99.94
121	99.87	100
122	100	100
123	100	100
124	98.76	100
200	99.72	100
201	97.24	99.26
202	99.2	100
203	85.93	99.67
205	99.68	100
207	99.69	99.87
208	99.22	99.83
209	100	99.84
210	96.55	99.95
212	100	100
213	99.7	100
214	100	100
215	99.43	100
217	97.56	99.94
219	99.66	100
220	100	100

221	99.7	100
222	89.74	99.63
223	99.32	100
228	13.98	100
230	100	100
231	100	100
232	99.93	99.93
233	99.65	100
234	99.96	100

C.9: Atrial Fibrillation Database- Lowpass Cutoff Frequency Changed

Using the Original Filter Parameters, but testing Channel 2 instead of Channel 1:

```
[b, a] = butter(2, [5 15]*2/fs, 'bandpass');  % 5-15 Hz 4th-order Bandpass Filter
[b,a] = butter(2,5*2/fs,'low');  % 2nd-order Lowpass Filter with 5 Hz cutoff freq
```

Record Number	QRS sensitivity (%)	QRS positive predictivity (%)
08455	99.2	99.66
08434	99.79	97.64
08405	98.71	97.07
08378	96.65	98.2
08219	20.36	96.92
08215	99.7	97.78
07910	99.97	96.73
07879	68.08	97.12
07859	84.54	97.12
07162	31.03	33.17
06995	98.32	97.08

06453	99.87	99.9
06426	91.67	97.66
05261	99.38	97.66
05121	94.2	97.69
05091	92.93	96.91
04936	99.73	97.29
04908	99.21	97.74
04746	27.28	99.7
04126	75.68	95.83
04048	99.58	99.7
04043	99.55	97.62
04015	8.68	96.14

Appendix D: R-Wave Detection Algorithm [40]

```
function [qrs_i_raw]=R_Wave_Detector(ecg,fs,gr)
%% function [qrs_i_raw]=R_Wave_Detector(ecg,fs, gr)
%% Inputs
% ecg : raw ecg vector signal 1d signal
% fs : sampling frequency e.g. 200Hz, 400Hz and etc
% gr : flag to plot or not plot (set it 1 to have a plot or set it zero not
% to see any plots
%% Output
% qrs_i_raw : index of R waves
%% References:
%[1] Sedghamiz. H, "Matlab Implementation of Pan Tompkins ECG QRS detector.",2014.
%https://www.researchgate.net/publication/313673153_Matlab_Implementation_of_Pan_To
mpkins_ECG_QRS_detector
%[2] PAN.J, TOMPKINS. W.J,"A Real-Time QRS Detection Algorithm" IEEE
% TRANSACTIONS ON BIOMEDICAL ENGINEERING, VOL. BME-32, NO. 3, MARCH 1985.
%%  ================= Read In Data ==================== %%
[ecg,fs,gr] = rdsamp('100.dat'); %Make sure to modify this prior to each run
%% ================= BioSigKit ==================== %%
ecg = ecg(:,1); % choose ecg column from dataset
if ~isvector(ecg)
  error('ecg must be a row or column vector');
end
if nargin < 3
    gr = 1;   % on default the function always plots
end
%% ======================= Initialize ================================ %
delay = 0;
skip = 0;                                   % becomes one when a T wave is detected
m_selected_RR = 0;
mean_RR = 0;
ser_back = 0;
%%  bandpass filter for Noise cancelation of other sampling frequencies(Filtering)
[b, a] = butter(2, [5 15]*2/fs, 'bandpass');
[Hnew, f] = freqz(b, a, [], fs);
ecg_bp = filtfilt(b, a, ecg);
ecg_bp = ecg_bp/max(abs(ecg_bp));
if gr
 plot(ecg);axis tight;title('Raw Signal'); zoom on;
end
%% ========== Squaring nonlinearly enhance the dominant peaks ========== %%
 ecg_s = ecg_bp.^2;
%% ============  Moving average ================= %%
%Implement a lowpass filter to calculate moving average
[b,a] = butter(2,5*2/fs,'low');
ecg_m = filtfilt(b,a,ecg_s);
```

```
ecg_m = ecg_m/ max(abs(ecg_m));
delay = delay + round(0.150*fs)/2;
%% ==================== Fiducial Marks ============================== %%
% Note : From a physiological point of view no RR wave can occur in less
% than 200 msec distance
[pks,locs] = findpeaks(ecg_m,'MINPEAKDISTANCE',round(0.2*fs));
%% ================== Initialize Some Other Parameters =============== %%
LLp = length(pks);
% ---------------- Stores QRS wrt Sig and Filtered Sig ------------------%
qrs_amp = zeros(1,LLp);         % amplitude of R
qrs_i = zeros(1,LLp);           % index
qrs_i_raw = zeros(1,LLp);       % amplitude of R raw
qrs_amp_raw= zeros(1,LLp);      % Index of raw signal
% ------------------- Noise Buffers ---------------------------------%
noise_c = zeros(1,LLp);
noise_i = zeros(1,LLp);
% ------------------- Buffers for Signal and Noise ----------------- %
SIGL_buf = zeros(1,LLp);
NOISE_buf = zeros(1,LLp);
SIGL_buf1 = zeros(1,LLp);
NOISE_buf1 = zeros(1,LLp);
THRS_buf1 = zeros(1,LLp);
THRS_buf = zeros(1,LLp);
%% initialize the training phase (2 seconds of the signal) to determine the THR_SIG
and THR_NOISE
THR_SIG = max(ecg_m(1:2*fs))*1/3;                       % 0.25 of the max amplitude
THR_NOISE = mean(ecg_m(1:2*fs))*1/2;% 0.5 of the mean signal is considered as noise
SIG_LEV= THR_SIG;
NOISE_LEV = THR_NOISE;
%% Initialize bandpass filter threshold(2 seconds of the bandpass signal)
THR_SIG1 = max(ecg_bp(1:2*fs))*1/3;                     % 0.25 of the max amplitude
THR_NOISE1 = mean(ecg_bp(1:2*fs))*1/2;
SIG_LEV1 = THR_SIG1;                            % Signal level in Bandpassed filter
NOISE_LEV1 = THR_NOISE1;                         % Noise level in Bandpassed filter
%% ============ Thresholding and decision rule ============= %%
Beat_C = 0;                                                          % Raw Beats
Beat_C1 = 0;                                                      % Filtered Beats
Noise_Count = 0;                                                   % Noise Counter
for i = 1 : LLp
   %% ===== locate the corresponding peak in the filtered signal === %%
    if locs(i)-round(0.150*fs)>= 1 && locs(i)<= length(ecg_bp)
          [y_i,x_i] = max(ecg_bp(locs(i)-round(0.150*fs):locs(i)));
       else
          if i == 1
            [y_i,x_i] = max(ecg_bp(1:locs(i)));
            ser_back = 1;
          elseif locs(i)>= length(ecg_bp)
```

```
            [y_i,x_i] = max(ecg_bp(locs(i)-round(0.150*fs):end));
        end
    end
 %% == calculate the mean last 8 R waves to ensure that QRS is not ==== %%
        if m_selected_RR
            test_m = m_selected_RR;              %if the regular RR available use it
        elseif mean_RR && m_selected_RR == 0
            test_m = mean_RR;
        else
            test_m = 0;
        end
    if test_m
          if (locs(i) - qrs_i(Beat_C)) >= round(1.66*test_m)       % it shows a QRS
is missed
              [pks_temp,locs_temp] = max(ecg_m(qrs_i(Beat_C)+
round(0.200*fs):locs(i)-round(0.200*fs))); % search back and locate the max in this
interval
              locs_temp = qrs_i(Beat_C)+ round(0.200*fs) + locs_temp -1; % location
              if pks_temp > THR_NOISE
               Beat_C = Beat_C + 1;
               qrs_amp(Beat_C) = pks_temp;
               qrs_i(Beat_C) = locs_temp;
              % ------------- Locate in Filtered Sig ------------- %
               if locs_temp <= length(ecg_bp)
                  [y_i_t,x_i_t] = max(ecg_bp(locs_temp-round(0.150*fs):locs_temp));
               else
                  [y_i_t,x_i_t] = max(ecg_bp(locs_temp-round(0.150*fs):end));
               end
              % ------------ Band pass Sig Threshold ------------------%
               if y_i_t > THR_NOISE1
                  Beat_C1 = Beat_C1 + 1;
                  qrs_i_raw(Beat_C1) = locs_temp-round(0.150*fs)+ (x_i_t - 1); %
save index of bandpass
                  qrs_amp_raw(Beat_C1) = y_i_t;        % save amplitude of bandpass
                  SIG_LEV1 = 0.25*y_i_t + 0.75*SIG_LEV1;      % when found with the
second thres
               end
               not_nois = 1;
               SIG_LEV = 0.25*pks_temp + 0.75*SIG_LEV ;    % when found with the
second threshold
             end
          else
              not_nois = 0;
          end
    end
    %% ===================  find noise and QRS peaks ================== %%
    if pks(i) >= THR_SIG
```

```
      % ------ if No QRS in 360ms of the previous QRS See if T wave ------%
       if Beat_C >= 3
          if (locs(i)-qrs_i(Beat_C)) <= round(0.3600*fs)
              Slope1 = mean(diff(ecg_m(locs(i)-round(0.075*fs):locs(i))));    %
mean slope of the waveform at that position
              Slope2 =
mean(diff(ecg_m(qrs_i(Beat_C)-round(0.075*fs):qrs_i(Beat_C)))); % mean slope of
previous R wave
              if abs(Slope1) <= abs(0.5*(Slope2))   % slope less then 0.5 of
previous R
                 Noise_Count = Noise_Count + 1;
                 noise_c(Noise_Count) = pks(i);
                 noise_i(Noise_Count) = locs(i);
                 skip = 1;                                      % T wave identification
                 % ----- adjust noise levels ------ %
                 NOISE_LEV1 = 0.125*y_i + 0.875*NOISE_LEV1;
                 NOISE_LEV = 0.125*pks(i) + 0.875*NOISE_LEV;
              else
                 skip = 0;
              end

           end
        end
        %---------- skip is 1 when a T wave is detected -------------- %
        if skip == 0
          Beat_C = Beat_C + 1;
          qrs_amp(Beat_C) = pks(i);
          qrs_i(Beat_C) = locs(i);
        %--------------- bandpass filter check threshold --------------- %
          if y_i >= THR_SIG1
              Beat_C1 = Beat_C1 + 1;
              if ser_back
                 qrs_i_raw(Beat_C1) = x_i;                 % save index of bandpass
              else
                 qrs_i_raw(Beat_C1)= locs(i)-round(0.150*fs)+ (x_i - 1);   % save
index of bandpass
              end
              qrs_amp_raw(Beat_C1) =  y_i;             % save amplitude of bandpass
              SIG_LEV1 = 0.125*y_i + 0.875*SIG_LEV1;        % adjust threshold for
bandpass filtered sig
           end
          SIG_LEV = 0.125*pks(i) + 0.875*SIG_LEV ;             % adjust Signal level
        end
    elseif (THR_NOISE <= pks(i)) && (pks(i) < THR_SIG)
         NOISE_LEV1 = 0.125*y_i + 0.875*NOISE_LEV1;   % adjust Noise level in
filtered sig
         NOISE_LEV = 0.125*pks(i) + 0.875*NOISE_LEV;    % adjust Noise level in MVI
```

```
    elseif pks(i) < THR_NOISE
        Noise_Count = Noise_Count + 1;
        noise_c(Noise_Count) = pks(i);
        noise_i(Noise_Count) = locs(i);
        NOISE_LEV1 = 0.125*y_i + 0.875*NOISE_LEV1; % noise level in filtered signal
        NOISE_LEV = 0.125*pks(i) + 0.875*NOISE_LEV;     % adjust Noise level in MVI
    end
    %% ================== adjust the threshold with SNR ============= %%
    if NOISE_LEV ~= 0 || SIG_LEV ~= 0
        THR_SIG = NOISE_LEV + 0.25*(abs(SIG_LEV - NOISE_LEV));
        THR_NOISE = 0.5*(THR_SIG);
    end
    %------ adjust the threshold with SNR for bandpassed signal -------- %
    if NOISE_LEV1 ~= 0 || SIG_LEV1 ~= 0
        THR_SIG1 = NOISE_LEV1 + 0.25*(abs(SIG_LEV1 - NOISE_LEV1));
        THR_NOISE1 = 0.5*(THR_SIG1);
    end
%--------- take a track of thresholds of smoothed signal -------------%
SIGL_buf(i) = SIG_LEV;
NOISE_buf(i) = NOISE_LEV;
THRS_buf(i) = THR_SIG;
%-------- take a track of thresholds of filtered signal ----------- %
SIGL_buf1(i) = SIG_LEV1;
NOISE_buf1(i) = NOISE_LEV1;
THRS_buf1(i) = THR_SIG1;
% ----------------------- reset parameters ------------------------- %
skip = 0;
not_nois = 0;
ser_back = 0;
end
%% ======================= Adjust Lengths ============================= %%
qrs_i_raw = qrs_i_raw(1:Beat_C1);
qrs_amp_raw = qrs_amp_raw(1:Beat_C1);
qrs_i = qrs_i(1:Beat_C);
qrs_amp = qrs_amp(1:Beat_C);
%% ================== overlay on the signals ========================= %%
 if gr
   figure;
   plot(ecg-mean(ecg));
   title('Detected R-Waves on ECG signal'); axis tight; zoom on;
   line(repmat(qrs_i_raw,[2 1]),...
       repmat([min(ecg-mean(ecg))/2; max(ecg-mean(ecg))/2],size(qrs_i_raw)),...
       'LineWidth',2.5,'LineStyle','-.','Color','r');
 end
%% Optional Functionality: Compares output of this function to an annotation file
% comparator(qrs_i_raw);
end
```

Appendix E: Miscellaneous Functions Written For Testing

E.1: comparator() Function

```
function []=comparator(qrs_i_raw)
%% Input
% qrs_i_raw : one of the RWave_Detection() function outputs; the R-wave
% pulse train returned by the function
%% Output
% a bxb report is created as a .txt file
%% Be sure to modify the record number for each run
%read in the R-wave locations from Physionet annotation file
[sig_ann, ann_type,  subtype, chan, num, comments] = rdann('100', 'atr');

%write a .test file that has the locations of the R-waves as determined by the
RWave_Detection() function
wrann('100', 'test', qrs_i_raw, 'N', zeros(length(qrs_i_raw),1),
ones(length(qrs_i_raw),1), zeros(length(qrs_i_raw),1), []);

clear; clc;
bxb('100','atr','test','100_bxbReport.txt');
end
```

E.2: bxbreport_table() Function

```
function [CM, Sense, PosPr] = bxbreport_table(Rfile)
%% bxbrdrpt: Read report generated by Physionet bxb().
%% Inputs
% Rfile: (string) Full name of the report file (including any needed
%        path information and filename extension).
%% Outputs
% CM: Confusion matrix.  A 7x7 (double) confusion matrix, as printed by bxb().
%     Columns are labeled n, s, v, f, q, o and x (truth). Columns are
%     same order, but label test results.  Lower right 2x2 submatrix
%     not used, so set to zero values.
% Sense: QRS sensitiviy, as listed in the report file.
% PosPr: QRS positive predictivity, as listed in the report file.
% a test.csv file with all the values from all the report files compiled
%% Be sure to modify the record number for each run in the first two lines
Rfile = '100_bxbReport.txt';
record_nums = 100;

fid = fopen(Rfile, 'rt');  % Open report file.
if fid<0, error(['bxbrdrpt: Cannot open ' Rfile]); end

for i=1:7  % Read unused header lines.
  tline = fgetl(fid); if tline == -1, error(['bxbrdprt: Bogus header in ' Rfile]);
end
end

%% Read 7 lines of tabular result data into confusion matrix CM.
%   Number of entries per line differ.  So, customize read for each line.
Label = 'NSVFQOX';       % Row labels, from bxb report table.
Entry = [7 7 7 7 7 5 5]; % Expected columns per row.
CM = zeros(7,7);         % Pre-allocate and initialize confusion matrix.
for row = 1:7  % Loop over each row in report file confusion matrix.
  tline = fgetl(fid);  % Read full line.
  if tline == -1, error(['bxbrdprt: Bogus row' Label(row) ' in ' Rfile]); end
  for col = 1:Entry(row)  % Build format.  Skip first two strings.
    format = '%*s %*s'; for i=1:Entry(row), format = [format ' %d']; end
%#ok<AGROW>
  end
  Avec = sscanf(tline, format, [1 Entry(row)]); % Read confuse matrix entries.
  CM(row, 1:Entry(row)) = Avec; % Insert into confusion matrix.
end

% Read QRS sensitivity and positive predictivity.
%   Empty line.
tline = fgetl(fid);
if tline == -1, error(['bxbrdprt: Bogus empty line after CM in ' Rfile]); end
```

```
%   QRS sensitivity.
tline = fgetl(fid);
if tline == -1, error(['bxbrdprt: Bogus QRS sensitivity line in ' Rfile]); end
Sense = sscanf(tline, '%*s %*s %f', [1 1]);
%   QRS positive predictivity.
tline = fgetl(fid);
if tline == -1, error(['bxbrdprt: Bogus QRS positive predictivity line in '
Rfile]); end
PosPr = sscanf(tline, '%*s %*s %*s %f', [1 1]);

% Close the file.
fclose(fid);

%% Tabulate all results into a csv
data = {'Record Number' 'QRS sensitivity' 'QRS positive predictivity'};
data1= {record_nums Sense PosPr};
Data = [data;data1];
if isfile('test.csv')
    read_arr = readmatrix('test.csv');
    [m,n] = size(read_arr);
    if m >= 1
        writecell(data1, 'test.csv','WriteMode','append')
    end     % File exists.
else
     writecell(Data, 'test.csv'); % File does not exist.
end
```

Appendix F: Atrial Fibrillation Algorithm Testing Result Averages

F.1: Testing with hanning window 24 and overlap 22

Y-Intercept Threshold	Mean Threshold	Average Sensitivity	Sensitivity Standard Deviation	Average Positive Predictivity	Positive Predictivity Standard Deviation
0.6	0.225	80.5	24.25	29.36	32.07
0.62	0.225	80.5	24.25	27.45	30.93
0.58	0.225	80.5	24.25	30.68	31.90
0.58	0.23	80.5	24.25	30.54	31.95
0.58	0.235	80.5	24.25	30.5	31.99
0.58	0.22	80.5	24.25	30.77	31.84
0.58	0.21	80.5	24.25	31.81	31.49
0.58	0.2	80.41	24.25	34.04	31.08
0.57	0.2	80.41	24.25	34.04	31.08
0.56	0.2	80.41	24.25	34.13	31.03
0.54	0.2	80.36	24.25	34.86	30.93
0.52	0.2	80.36	24.25	36.54	30.40
0.5	0.2	80.31	24.25	37.59	30.05
0.48	0.2	79.31	25.42	41.63	30.63
0.46	0.2	79.31	25.42	41.63	30.63
0.44	0.2	77.72	27.05	56.36	31.38
0.44	0.19	77.72	27.05	57.5	31.49
0.44	0.18	77.72	27.05	59.13	31.37

0.44	0.16	75.90	27.99	72.22	27.31
0.44	0.15	70.5	31.63	80.45	28.56

F.2: Testing with window 16 overlap 15

Y-Intercept Threshold	Mean Threshold	Average Sensitivity	Sensitivity Standard Deviation	Average Positive Predictivity	Positive Predictivity Standard Deviation
0.65	0.25	76.81	27.99	53.81	32.09
0.63	0.25	75.81	28.71	65.77	31.81
0.63	0.23	75.81	28.71	67.22	31.22
0.63	0.21	71.54	32.70	82.13	28.00
0.61	0.21	71.54	32.70	82.18	28.00
0.59	0.21	70.95	33.03	83.72	26.99
0.56	0.21	63.59	33.23	85.09	27.46
0.53	0.21	52.18	33.63	87.57	25.45
0.67	0.21	71.54	32.70	82.13	28.00

F.3: Testing with hanning window 16 overlap 15 and TVCF Slope calculated from 0.3Hz - 0.5Hz Normalized Frequency

Y-Intercept Threshold	Mean Threshold	Average Sensitivity	Sensitivity Standard Deviation	Average Positive Predictivity	Positive Predictivity Standard Deviation
0.2	0.21	71.54545	32.70	82.13636	28.00
0.18	0.21	71.54545	32.70	82.13636	28.00
0.18	0.3	80.40909	24.25	29	30.91
0.18	0.28	80.40909	24.25	30.90909	31.78

0.18	0.26	80.36364	24.25	34.59091	30.84
0.18	0.24	78.31818	25.76	45.04545	31.13
0.18	0.22	76.27273	28.70	67.5	30.38
0.16	0.21	71.54545	32.70	82.13636	28.00
0.14	0.21	71.54545	32.70	82.13636	28.00
0.12	0.21	71.54545	32.70	82.13636	28.00
0.1	0.21	71.54545	32.70	82.13636	28.00
0.06	0.21	71.54545	32.70	82.22727	28.01
0.02	0.21	70.81818	32.58	83.22727	28.35
0.005	0.21	63.95455	32.56	84.59091	27.53
0.26	0.21	71.54545	32.70	82.13636	28.00
0.28	0.21	71.54545	32.70	82.13636	28.00
0.3	0.21	71.54545	32.70	82.13636	28.00
0.16	0.235	78.27273	25.77	50.22727	32.04

Appendix G: Atrial Fibrillation Detection Algorithm MATLAB Code

G.1: AFIB_Detector Function

```
function [Samples,Comment_array]=AFIB_Detector(Rsample,yThresh,mThresh,pL)
%detects atrial fibrillation for the sample locations of R waves from ECG
filterMask = ones(length(Rsample),1);% creates mask that is used for finding R-wave
samples
%from the segmented signal after classification
close all %closes all plots
%RR- interval
rrint = diff(Rsample);
filterMask(end) = 0;%zeros added to mask for removed samples
%% filter ectopic beats
[rr_NBeats, ectBeat]=ectopicBeat(rrint);
filterMask(ectBeat) = 0;%zeros added to mask for removed samples
%%
%find TVCF
TVCFwind = 128;%RR interval segment length
overlap = 14;%window overlap
TVCF = findTVCFM(rr_NBeats, TVCFwind,overlap,pL);

%% slopes
CoeffSlope = Slope(TVCF);
if pL == true
figure
scatter(0:length(CoeffSlope)-1,CoeffSlope(:,2),'r')%scatter plot of y-intercepts
title(['TVCF y-intercept with with segment length ', num2str(TVCFwind), ' and
overlap ', num2str(overlap)])
xlabel('Segment')
ylabel('TVCF')
end
%% Shannon Entropy
SHwind = TVCFwind;%RR- interval segemnt same as for TVCF calculations
bin_size = 16;
se = Shannon(rr_NBeats, SHwind, bin_size);
if pL == true
figure
scatter(0:length(se)-1,se)
title('Shannon Entropy')
xlabel('Segment')
ylabel('Magnitude')
end
%% Mean
```

```
meanTVCF = mean(TVCF(:,35:50),2);
if pL == true
figure
scatter(0:length(meanTVCF)-1,meanTVCF)%scatter plot of the mean values
title('mean TVCF across frequency')
xlabel('Segment')
ylabel('Mean')
end
%% SE vs y-int
if pL == true
figure
scatter(se(1:end-1),CoeffSlope(:,2))
title('SE vs y-int')
xlabel('SE')
ylabel('y-int')
end
%% Mean vs SE
if pL == true
figure
scatter(se(1:end-1),meanTVCF)
title('SE vs Mean Frequency')
xlabel('SE')
ylabel('Mean Frequency')
end


%% y-int vs mean
if pL == true
figure
scatter(meanTVCF,CoeffSlope(:,2))
title('Mean Frequency vs y-int')
xlabel('Mean Frequency')
ylabel('y-int')
end


%% Classification
SeThresh = 0.79;
Size= size(TVCF);
Beat_type = zeros(1,Size(1));
for i = 1:Size
    if (CoeffSlope(i,2) < yThresh) && (meanTVCF(i) < mThresh) && (se(i) >=
SeThresh)
        Beat_type(i) = 1;%a-fib
    else
        Beat_type(i) = 0;%NSR
    end
```

```
end
%% Sample finder
BeatSwitch = diff(Beat_type);%finds the difference of array values so start of afib
is 1 and start of normal
%rythm is -1
afib_seg = (find(BeatSwitch == 1) + 1) * TVCFwind;%segment location times segment
length to calculate sample location
NSR_seg = (find(BeatSwitch == -1) + 1) * TVCFwind;
afib_count = ones(length(afib_seg),1);
NSR_count = ones(length(NSR_seg),1);

%because ectopic beats are removed the sample locations can not be
%referenced to the original data.
%following code takes into account the removed data to give the correct
%data locations
for i = 1:length(afib_seg)
    found = false;
    afib_MaskSum = 0;
    while found == false
        %when sum of mask  = afib_seg(i)
        %return afib_count
        afib_MaskSum = afib_MaskSum + filterMask(afib_count(i));

        if afib_MaskSum == afib_seg(i)
            found = true;
        end
        afib_count(i) = afib_count(i)+1;
    end
end

for i = 1:length(NSR_seg)
    found = false;
    MaskSum = 0;
    while found == false
        MaskSum = MaskSum + filterMask(NSR_count(i));
        if MaskSum == NSR_seg(i)
            found = true;
        end
        NSR_count(i) = NSR_count(i)+1;
    end
end

afib_samp = ones(length(afib_count),2);%1 indicates atrial fibrillation
NSR_samp = zeros(length(NSR_count),2);%zero indicates NSR
afib_samp(:,1) = Rsample(afib_count);%adds afib data values to array
NSR_samp(:,1) = Rsample(NSR_count);%adds NSR data values to array
samp_locs = [afib_samp;NSR_samp];%combines both arrays
```

```
samp_locs = sortrows(samp_locs);%orders array by column one values
%% AFIB tables
%creates arrays that are used in epicmp command in the episode comparitor
Samples = zeros(length(samp_locs)+1,1);
Samples(1) = Rsample(1);
Samples(2:end) = samp_locs(:,1);

Comment_array = cell(length(Samples),1);
if Beat_type(1) == 0
    Comment_array{1}='(N';
else
    Comment_array{1} = '(AFIB';
end

for i = 1:length(Samples)-1
    if samp_locs(i,2) == 0
    Comment_array{i+1}='(N';
    else
    Comment_array{i+1} = '(AFIB';
    end
end
```

G.2: ectopicBeat function

```
function [rr_NBeats,ectBeat]=ectopicBeat(rrint)
%removes ectopic beat data from rrint data returns locations of ectopic
%beats and array with ectopic beats removed
rrRatios = zeros(1,length(rrint)-1);%allocating space for the array of interval
ratios
for j = 2:length(rrint)
    rrRatios(j-1) = rrint(j)/rrint(j-1);%array of the ratios of rrint
end
%finding the 1st 25th and 99th percentile of the ratio array
perc1 = prctile(rrRatios,1);
perc25 = prctile(rrRatios,25);
perc99 = prctile(rrRatios,99);
rr_NBeats = zeros(1,length(rrint));%array of new rrintervals
counter = 2;
for i = 2:length(rrint)-2
    %1)RR(i)/RR(i-1)<perc1, 2) RR(i + 1)/RR(i)>perc99 and 3) RR(i+ 1)/RR(i
    %+ 2)>perc25,ectopic beat conditions
    if (rrint(counter)/rrint(counter-1))<perc1 &&
(rrint(counter+1)/rrint(counter))>perc99 &&
(rrint(counter+1)/rrint(counter+2))>perc25
        counter = counter+1;
```

```
    else
        rr_NBeats(counter-1) = rrint(counter);%when it is a normal beat the
interval is added to the array
    end
    counter = counter +1;
    if counter+2 == length(rrint)%prevents indexing error
        break
    end
end
ectBeat=find(rr_NBeats==0);%finds the locations of all the ectopic beats
rr_NBeats(ectBeat) = [];%removes the ectopic beats
end
```

G.3: findTVCF function

```
function TVCF = findTVCFM(Signal, seg_length,overlap,pL)
%Signal is the RR-interval
%seg_length is the length of the segments to be compared in cpsd
%pL if true plots mesh plot of TVCF
x=Signal(1:(end-seg_length));%creates x array used in cspd function from input
signal
y=Signal(seg_length+1:end);%is the same as x only shifted by size of the seg_length
%this is so the adjacent segments can be compared
Size = floor(length(y)/seg_length);
counter = 1;
TVCF = zeros(Size,129);

for i = 1:Size
    %Magnitude squared coherence calculated
    X =
mscohere(x(counter:(counter+seg_length-1)),y(counter:(counter+seg_length-1)),hann(1
5),overlap);
    TVCF(i,:) = X.^2;%array is squared
    counter = counter+seg_length;%shifts index values by seg_length
end
if pL == true
    %mesh plot of data
    figure
    Y = linspace(0,.5,129);
    X = linspace(0,Size*seg_length,Size);
    Z = TVCF.';
    mesh(X,Y,Z)
    title(['TVCF with segment length ', num2str(seg_length), ' and overlap ',
num2str(overlap)])
    ylabel('Normalized Frequency')
    xlabel('Beats')
```

```
    zlabel('TVCF')
end

end
```

G.4: Slope function

```
function CoeffSlope = Slope(TVCF)
%TVCF is the time varying coherence function
%returns slopes and y-intersepts of frequency vs TVCF data
Size = size(TVCF);
CoeffSlope = zeros(Size(1),2);%prealocating space
X = linspace(0,1,129-5);%normalized frequency spectrum
for i=1:Size(1)
    CoeffSlope(i,:)= polyfit(X,TVCF(i,6:end),1);%line fot from frequency 5 to 129
end
end
```

G.5: Shannon function [42]

```
function se = Shannon(signal, seg_length, bin_size)
%This function calculates the shannon entropy of a signal. It has an input
%signal, segment length input, and a bin size input
%the output of shannon entropy is an array of values
    Size = floor(length(signal)/seg_length);
    shift = 1;
    se = zeros(1,Size);

    for i=1:Size
        %Create vector for signal without outliers
        signal_minus_outliers = signal(shift:shift+seg_length-1);

        %Identify maximum outliers
        for n = 1:8%Identifies top 8 outliers
            [max_value, max_index] = max (signal_minus_outliers);
            signal_minus_outliers (max_index) = 0;
        end

        %Identify minimum outliers
        for n = 1:8%Identifies bottom 8 outliers
            [min_value, min_index] = min (signal_minus_outliers);
            signal_minus_outliers (min_index) = 0;
```

```
        end

        %Remove outliers
        signal_minus_outliers (signal_minus_outliers == 0) = [];

        %Calculate histogram bins
        h = hist (signal_minus_outliers, bin_size);

        %Create empty vector for bin probabilities
        probabilities = zeros (bin_size, 1);

        %Calculate probability of each bin
        for n = 1:bin_size
            probabilities (n) = h (n) / (seg_length - bin_size);
        end


        %Calculate shannon entropy
        for n = 1:bin_size
            if probabilities (n) ~= 0
                se(i) = se(i) + probabilities (n) * (log (probabilities (n)) / log
(1 / bin_size));
            end
        end
        shift = shift+seg_length;
    end


end
```

Appendix H: Atrial Fibrillation Detection Algorithm Testing Code

H.1: AFIB_Detect function for testing

```
function se = Shannon(signal, seg_length, bin_size)
%This function calculates the shannon entropy of a signal. It has an input
%signal, segment length input, and a bin size input
%the output of shannon entropy is an array of values
    Size = floor(length(signal)/seg_length);
    shift = 1;
    se = zeros(1,Size);

    for i=1:Size
        %Create vector for signal without outliers
        signal_minus_outliers = signal(shift:shift+seg_length-1);

        %Identify maximum outliers
        for n = 1:8%Identifies top 8 outliers
            [max_value, max_index] = max (signal_minus_outliers);
            signal_minus_outliers (max_index) = 0;
        end

        %Identify minimum outliers
        for n = 1:8%Identifies bottom 8 outliers
            [min_value, min_index] = min (signal_minus_outliers);
            signal_minus_outliers (min_index) = 0;
        end

        %Remove outliers
        signal_minus_outliers (signal_minus_outliers == 0) = [];

        %Calculate histogram bins
        h = hist (signal_minus_outliers, bin_size);

        %Create empty vector for bin probabilities
        probabilities = zeros (bin_size, 1);

        %Calculate probability of each bin
        for n = 1:bin_size
            probabilities (n) = h (n) / (seg_length - bin_size);
        end
```

```
        %Calculate shannon entropy
        for n = 1:bin_size
            if probabilities (n) ~= 0
                se(i) = se(i) + probabilities (n) * (log (probabilities (n)) / log
(1 / bin_size));
            end
        end
        shift = shift+seg_length;
    end


endfilterMask(end) = 0;%zeros added to mask for removed samples
%% filter ectopic beats
[rr_NBeats, ectBeat]=ectopicBeat(rrint);
filterMask(ectBeat) = 0;%zeros added to mask for removed samples
%%
%find TVCF
TVCFwind = 128;%RR interval segment length
overlap = 14;%window overlap
TVCF = findTVCFM(rr_NBeats, TVCFwind,overlap,pL);

%% slopes
CoeffSlope = Slope(TVCF);
%plot y-intercept values
if pL == true
figure
scatter(0:length(CoeffSlope)-1,CoeffSlope(:,2),'r')%scatter plot of y-intercepts
title(['TVCF y-intercept with with segment length ', num2str(TVCFwind), ' and
overlap ', num2str(overlap)])
xlabel('Segment')
ylabel('TVCF')
end
%% Shannon Entropy
SHwind = TVCFwind;%RR- interval segemnt same as for TVCF calculations
bin_size = 16;
se = Shannon(rr_NBeats, SHwind, bin_size);
if pL == true
figure
scatter(0:length(se)-1,se)
title('Shannon Entropy')
xlabel('Segment')
ylabel('Magnitude')
end
%% Mean
meanTVCF = mean(TVCF(:,35:50),2);
```

```
if pL == true
figure
scatter(0:length(meanTVCF)-1,meanTVCF)%scatter plot of the mean values
title('mean TVCF across frequency')
xlabel('Segment')
ylabel('Mean')
end
%% SE vs y-int
if pL == true
figure
scatter(se(1:end-1),CoeffSlope(:,2))
title('SE vs y-int')
xlabel('SE')
ylabel('y-int')
end
%% Mean vs SE
if pL == true
figure
scatter(se(1:end-1),meanTVCF)
title('SE vs Mean Frequency')
xlabel('SE')
ylabel('Mean Frequency')
end
%% y-int vs mean
if pL == true
figure
scatter(meanTVCF,CoeffSlope(:,2))
title('Mean Frequency vs y-int')
xlabel('Mean Frequency')
ylabel('y-int')
end
%% Classification
Size= size(TVCF);
Beat_type = zeros(1,Size(1));
for i = 1:Size
    if (CoeffSlope(i,2) < yThresh) && (meanTVCF(i) < mThresh) && (se(i) >=
SeThresh)
        Beat_type(i) = 1;%a-fib
    else
        Beat_type(i) = 0;%NSR
    end
end
%% Sample finder
BeatSwitch = diff(Beat_type);%finds the difference of array values so start of afib
is 1 and start of normal
%rythm is -1
afib_seg = (find(BeatSwitch == 1) + 1) * TVCFwind;%segment location times segment
```

```
length to calculate sample location
NSR_seg = (find(BeatSwitch == -1) + 1) * TVCFwind;
afib_count = ones(length(afib_seg),1);
NSR_count = ones(length(NSR_seg),1);

%because ectopic beats are removed the sample locations can not be
%referenced to the original data.
%following code takes into account the removed data to give the correct
%data locations
for i = 1:length(afib_seg)
    found = false;
    afib_MaskSum = 0;
    while found == false
        %when sum of mask  = afib_seg(i)
        %return afib_count
        afib_MaskSum = afib_MaskSum + filterMask(afib_count(i));

        if afib_MaskSum == afib_seg(i)
            found = true;
        end
        afib_count(i) = afib_count(i)+1;
    end
end

for i = 1:length(NSR_seg)
    found = false;
    MaskSum = 0;
    while found == false
        MaskSum = MaskSum + filterMask(NSR_count(i));
        if MaskSum == NSR_seg(i)
            found = true;
        end
        NSR_count(i) = NSR_count(i)+1;
    end
end

afib_samp = ones(length(afib_count),2);%1 indicates atrial fibrillation
NSR_samp = zeros(length(NSR_count),2);%zero indicates NSR
afib_samp(:,1) = rWave(afib_count);%adds afib data values to array
NSR_samp(:,1) = rWave(NSR_count);%adds NSR data values to array
samp_locs = [afib_samp;NSR_samp];%combines both arrays
samp_locs = sortrows(samp_locs);%orders array by column one values

%% tables
%creates arrays that are used in epicmp command in the episode comparitor
Samples = zeros(length(samp_locs)+1,1);
Samples(1) = rWave(1);
```

```
Samples(2:end) = samp_locs(:,1);

Comment_array = cell(length(Samples),1);
if Beat_type(1) == 0
    Comment_array{1}='(N';
else
    Comment_array{1} = '(AFIB';
end

for i = 1:length(Samples)-1
    if samp_locs(i,2) == 0
    Comment_array{i+1}='(N';
    else
    Comment_array{i+1} = '(AFIB';
    end
end
%writes file
wrann(subject, 'test', Samples, '+', 0, 1, 0, Comment_array);
%runs episode comparitor
command2 = ['epicmp -r ',num2str(subject),' -a atr test -A ',Rfile];
status_epicmp = system(command2);

end
```

H.2: epicmpreport_reader function

```
function [Sense, PosPr, DurSense, DurPosPr] = epicmpreport_reader(Rfile, record_nums,
yThresh,mThresh,SeThresh)
%reads reports from epicmp and adds them to a csv file
fid = fopen(Rfile, 'rt');  % Open report file.
if fid<0, error(['epicmprdrpt: Cannot open ' Rfile]); end

for i = 1:4
    tline = fgetl(fid); if tline == -1, error(['epicmprdrpt: Bogus header in ' Rfile]); end
end
tline = fgetl(fid);
if tline == -1, error(['epicmprdrpt: Bogus QRS sensitivity line in ' Rfile]); end
Sense = sscanf(tline, '%*s %*s %f %*s %*c %d %*c %d', [1 3]);

tline = fgetl(fid);
if tline == -1, error(['epicmprdrpt: Bogus QRS positive predictivity line in ' Rfile]); end
PosPr = sscanf(tline, '%*s %*s %*s %f %*s %*c %d %*c %d', [1 3]);

tline = fgetl(fid);
if tline == -1, error(['epicmprdrpt: Bogus QRS sensitivity line in ' Rfile]); end
```

```
DurSense = sscanf(tline, '%*s %*s %f', [1 1]);

tline = fgetl(fid);
if tline == -1, error(['epicmprdrpt: Bogus QRS positive predictivity line in ' Rfile]); end
DurPosPr = sscanf(tline, '%*s %*s %*s %f ', [1 1]);

fclose(fid);

%% Tabulate all results into a csv
data = {'Record Number' 'Episode sensitivity' 'Episode positive predictivity' 'Duration
sensitivity' 'Duration positive predictivity' 'yThresh' 'mThresh' 'SeThresh'};
data1= {record_nums Sense PosPr DurSense DurPosPr yThresh mThresh SeThresh};
Data = [data;data1];
if isfile('test.csv')
    read_arr = readmatrix('test.csv');
    [m,n] = size(read_arr);
    if m >= 1
        writecell(data1, 'test.csv','WriteMode','append')
    end     % File exists.
else
     writecell(Data, 'test.csv'); % File does not exist
end
end
```

References

[1] J. Lee, Y. Nam, D. D. McManus, and K. H. Chon, "Time-Varying Coherence Function for Atrial Fibrillation Detection," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 10, pp. 2783–2793, Oct. 2013, doi: 10.1109/TBME.2013.2264721.

[2] "Ch. 1 Introduction - Anatomy and Physiology | OpenStax." https://openstax.org/books/anatomy-and-physiology/pages/1-introduction (accessed Dec. 14, 2021).

[3] Nyq, *Diagram of the human heart*. [Online]. Available: https://commons.wikimedia.org/wiki/File:Diagram_of_the_human_heart.svg

[4] *Anatomy and Pysiology*, vol. 2, 3 vols. 2013. Accessed: Oct. 13, 2021. [Online]. Available: https://textbookequity.org/Textbooks/anatomy+phys+vol2a.pdf

[5] Madhero88, *English: Electrical conduction system of the heart*. Accessed: Sep. 28, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Conductionsystemoftheheart.png

[6] S. Toinga, C. Carabali, and L. Ortega, "Development of a didactic platform for teaching the Einthoven's Triangle," in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, Oct. 2017, pp. 1–6. doi: 10.1109/ETCM.2017.8247542.

[7] J. Aspuru *et al.*, "Segmentation of the ECG Signal by Means of a Linear Regression Algorithm," *Sensors*, vol. 19, no. 4, p. 775, Feb. 2019, doi: 10.3390/s19040775.

[8] R. V. Andreao, B. Dorizzi, and J. Boudy, "ECG signal analysis through hidden Markov models," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 8, pp. 1541–1549, Aug. 2006, doi: 10.1109/TBME.2006.877103.

[9] D. Nabil and F. Bereksi Reguig, "Ectopic beats detection and correction methods: A review," *Biomed. Signal Process. Control*, vol. 18, pp. 228–244, Apr. 2015, doi: 10.1016/j.bspc.2015.01.008.

[10] D. S. Desai and S. Hajouli, "Arrhythmias," in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2021. Accessed: Dec. 10, 2021. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK558923/

[11] Springhouse, *ECG Interpretation: A 2-in-1 Reference for Nurses*. Philadelphia, UNITED STATES: Wolters Kluwer Health, 2004. Accessed: Oct. 04, 2021. [Online]. Available: http://ebookcentral.proquest.com/lib/wpi/detail.action?docID=2032544

[12] L. Sörnmo and P. Laguna, "Chapter 6 - The Electrocardiogram—A Brief Background," in *Bioelectrical Signal Processing in Cardiac and Neurological Applications*, L. Sörnmo and P. Laguna, Eds. Burlington: Academic Press, 2005, pp. 411–452. doi: 10.1016/B978-012437552-9/50006-4.

[13] Jmarchn, *English: Precordial electrodes necessary for a 12-lead ECG*. Accessed: Oct. 05, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Precordial_Leads_2.svg

[14] M. Cadogan, "ECG Lead positioning," *Life in the Fast Lane • LITFL*, May 20, 2021. https://litfl.com/ecg-lead-positioning/ (accessed Sep. 12, 2021).

[15] Jmarchn, *English: ECG electrodes*. Accessed: Oct. 05, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Limb_leads_2_ENG.svg

[16] Npatchett, *English: Derivation of the limb leads*. 2015. Accessed: Oct. 05, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Limb_leads_of_EKG.png

[17] "Holter Monitor."

https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/holter-monitor (accessed Oct. 12, 2021).
[18] L. G. Tereshchenko and M. E. Josephson, “Frequency Content and Characteristics of Ventricular Conduction,” *J. Electrocardiol.*, vol. 48, no. 6, pp. 933–937, 2015, doi: 10.1016/j.jelectrocard.2015.08.034.
[19] W. J. Tompkins and M. L. Ahlstrom, “Digital Filters for Real Time ECG,” *IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/4122146/citations#citations (accessed Dec. 06, 2021).
[20] R. P. Sallen and E. L. Key, “A Practical Method of Designing RC Active Filters,” *IEEE Xplore*.
[21] Electrical4u, “Summing Amplifier or Op Amp Adder,” Oct. 28, 2020. https://www.electrical4u.com/summing-amplifier/ (accessed Dec. 15, 2021).
[22] B. Le, T. W. Rondeau, J. H. Reed, and C. W. Bostian, “Analog-to-digital converters,” *IEEE Signal Process. Mag.*, vol. 22, no. 6, pp. 69–77, Nov. 2005, doi: 10.1109/MSP.2005.1550190.
[23] F. Shaffer, R. McCraty, and C. L. Zerr, “A healthy heart is not a metronome: an integrative review of the heart’s anatomy and heart rate variability,” *Front. Psychol.*, vol. 5, p. 1040, 2014, doi: 10.3389/fpsyg.2014.01040.
[24] D. Chabot, M. Bayer, and A. de Roos, “Instantaneous heart rates and other techniques introducing errors in the calculation of heart rate,” *Can. J. Zool.*, vol. 69, pp. 1117–1120, Apr. 1991, doi: 10.1139/z91-156.
[25] J. Pan and W. J. Tompkins, “A Real-Time QRS Detection Algorithm,” *IEEE Trans. Biomed. Eng.*, vol. BME-32, no. 3, pp. 230–236, Mar. 1985, doi: 10.1109/TBME.1985.325532.
[26] “Heart Rate Variability | Circulation.” https://www.ahajournals.org/doi/full/10.1161/01.cir.93.5.1043 (accessed Oct. 05, 2021).
[27] S. Dash, K. H. Chon, S. Lu, and E. A. Raeder, “Automatic Real Time Detection of Atrial Fibrillation,” *Ann. Biomed. Eng.*, vol. 37, no. 9, pp. 1701–1709, Sep. 2009, doi: 10.1007/s10439-009-9740-z.
[28] D. Cantillon MD and R. Amuthan MD, “Atrial Fibrillation,” *Cleveland Clinic Center for Continuing Education*, Aug. 2018. https://www.clevelandclinicmeded.com/medicalpubs/diseasemanagement/cardiology/atrial-fibrillation/#fig03
[29] W. B. Kannel, P. A. Wolf, E. J. Benjamin, and D. Levy, “Prevalence, incidence, prognosis, and predisposing conditions for atrial fibrillation: population-based estimates 11Reprints are not available.,” *Am. J. Cardiol.*, vol. 82, no. 7, Supplement 1, pp. 2N-9N, Oct. 1998, doi: 10.1016/S0002-9149(98)00583-9.
[30] K. Tateno and L. Glass, “Automatic detection of atrial fibrillation using the coefficient of variation and density histograms of RR and ARR intervals,” *Biol. Eng.*, vol. 39, p. 8, 2001.
[31] R. Peck, C. Olsen, and J. L. Devore, *Introduction to statistics and data analysis*, 3rd ed. Australia ; Belmont, CA: Thomson Brooks/Cole, 2008.
[32] J. Lee, D. McManus, and K. Chon, “Atrial Fibrillation detection using time-varying coherence function and Shannon Entropy,” in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, MA, Aug. 2011, pp. 4685–4688. doi: 10.1109/IEMBS.2011.6091160.
[33] International Electrotechnical Committee, “Medical electrical equipment - Part 2-25:

Particular requirements for the basic safety and essential performance of electrocardiographs." https://webstore.iec.ch/publication/2636 (accessed Sep. 22, 2021).
[34] "IEEE 802.15.1-2002 - IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN - Specific Requirements - Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)." https://standards.ieee.org/standard/802_15_1-2002.html (accessed Sep. 22, 2021).
[35] Analog Devices, "AD624 Datasheet." Accessed: Oct. 07, 2021. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/AD624.pdf
[36] Analog Devices, "AD8227 Datasheet." Accessed: Nov. 30, 2021. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/AD8227.pdf
[37] Linear Technology, "LT1167 Datasheet." [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/1167fc.pdf
[38] Texas Instruments, "LM324-n Datasheet." Accessed: Dec. 01, 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm324-n.pdf?ts=1638353644532&ref_url=https%253A%252F%252Fwww.google.com%252F
[39] H.-Y. Hsieh, C.-H. Luo, and C.-C. Tai, "Wireless potential difference electrocardiogram constituted by two electrode-pairs wearing comfort," *J. Instrum.*, vol. 15, no. 08, pp. P08011–P08011, Aug. 2020, doi: 10.1088/1748-0221/15/08/P08011.
[40] "Complete Pan Tompkins Implementation ECG QRS detector - File Exchange - MATLAB Central." https://www.mathworks.com/matlabcentral/fileexchange/45840-complete-pan-tompkins-implementation-ecg-qrs-detector (accessed Oct. 11, 2021).
[41] A. Papoulis, *Probability, RandomVariables, and Stochastic Process*, Second. McGraw-Hill Book Company, 1984.
[42] N. Goldstein, D. Song, and D. Thompson, *Atrial-Fibrillation-Detection-from-BIH-MIT-Database*. 2015. Accessed: Dec. 16, 2021. [Online]. Available: https://github.com/danthompson41/Atrial-Fibrillation-Detection-from-BIH-MIT-Database
[43] S. Braun, "WINDOWS," in *Encyclopedia of Vibration*, S. Braun, Ed. Oxford: Elsevier, 2001, pp. 1587–1595. doi: 10.1006/rwvb.2001.0052.
[37] Conover, M. B. (2002). *Understanding Electrocardiography*. Elsevier Health Sciences. Pp. 4-7
[38] Babusiak, B., Borik, S., & Smondrk, M. (2020). Two-Electrode ECG for Ambulatory Monitoring with Minimal Hardware Complexity. *Sensors*, *20*(8), 2386. https://doi.org/10.3390/s20082386
[39] Texas Instruments, "TPS73615MDBVREP Datasheet." Accessed: Dec. 10, 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps73615-ep.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1639185179581&ref_url=https%253A%252F%252Fwww.mouser.com%252F
[40] Texas Instruments, "LP38841T-1.5/NOPB Datasheet." Accessed: Dec. 10, 2021. Available: https://www.ti.com/lit/ds/symlink/lp38841.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1639165339792