UNIVERSITY COLLEGE DUBLIN

IoT Device for Electrocardiography Summary Statistics Monitoring

by

Sam Luby

A thesis report submitted in partial fulfillment for the research project as part of
the ME degree in Electronic and Computer Engineering

in the
School of Electrical, Electronic and Communications Engineering

April 2018

Declaration of Authorship

I, Sam Luby, declare that this thesis titled, 'IoT Device for Electrocardiography Summary Statistics Monitoring' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSITY COLLEGE DUBLIN

Abstract

School of Electrical, Electronic and Communications Engineering

by Sam Luby

This thesis presents a method for monitoring and analysing electrocardiograph data captured from a patient. The methodology uses an economically viable implementation of measuring and capturing ECG data, and uses Pan Tompkins algorithm for feature identification on the ECG signal. Heart rate variability analysis is performed on the processed signal to assess the condition of the patient, and the analysis results are sent to the cloud.

Acknowledgements

Firstly I would like to thank my supervisors Dr. Deepu John and Dr. Barry Cardiff for their continued support and guidance throughout the duration of this project. I thoroughly enjoyed working on such an interesting topic under their supervision.

I would like to thank my parents for their support throughout my time in UCD.

Finally, I would like to thank my friends and colleagues in UCD for the enjoyable experience over the past five years, in particular: Darren Coughlan, Conor Power, Chris Hayes & Kevin Keegan.

Contents

Chapter 1

Introduction

Electrocardiography (ECG) is the process of measuring the small electrical signals that are a result of the activity of the heart's muscles, namely the processes of depolarization and repolarization. It is one of the most basic forms of non-intrusive diagnosis in medicine and is usually performed by placing small electrodes on the surface of a patient's skin.

ECG monitoring measures the tiny electrical potential changes, known as biopotentials , due to the hearts activity. The electrodes of an ECG machine record these changing biopotentials and the signal is represented in the form of a moving voltage versus time graph.

Ideally, the process of electrocardiography would record signals that were only a result of the cardiac cycle. In reality, the recorded signal is corrupted with various unwanted artefacts, which are a result of noise or other muscular contractions in

the body. Thus, one of the steps required when performing heart monitoring is some sort of filtering process to remove this unwanted noise.

Various analysis techniques can be performed on a patient's ECG signal to gain an understanding of their cardiovascular health, and to perform diagnosis of cardiovascular arrhythmias and diseases.

1.1 Motivation

The motivation behind this project is about introducing a novel solution to the growing problem of cardiovascular diseases. Cardiovascular diseases, or CVDs, are the leading cause of mortality in the world, amounting to a total of approximately 17 million deaths globally - or 32% of all moralities[1]. However, there is a disproportionality here in that of these figures, more than 80% are in low to middle-income countries[2] such as parts of Asia, Eastern Europe and North Africa. It is clear that the less wealthy are more at risk because many do not have access to proper healthcare.

It is clear that current implementations for monitoring are not enough. The problem of heart failure in Ireland is set to dramatically increase according to the Irish Heart Foundation[1] and without proper monitoring, heart failure can lead to cardiac arrest which can cause permanent brain damage after just a few minutes,

[1] https://irishheart.ie/our-mission/our-policies/heart-disease-irelands-no-1-killer/

eventually leading to a total shutdown of the body's vital organs. Regular monitoring and early diagnosis can help to expose underlying issues in the circulatory system.

1.2 Project Aim and Approach

The problem this project aims to address is how to easily monitor a patient's circulatory system to tackle the increasing issue of heart disease. The ability to automate the monitoring process could help to reduce cardiovascular disease mortality rate, as well as to reduce the healthcare costs associated. The aim of this project is to develop a novel and cost-effective solution for the heart monitoring process, to provide an initial diagnosis to uncover any abnormalities or underlying issues.

The project has two main objectives that I wish to focus on. Firstly is the hardware used to capture and monitor the patient's heart health, to accurately measure the biopotential signal at a fixed sampling rate and accumulate the data for a given period of time. The hardware must also be capable of transmitting data to the cloud, namely the results of HRV analysis.

The second objective is to develop an algorithm to perform analysis on the patient's ECG data. The aim here is to uncover any abnormalities in the patient data, such as irregular or high heart rates or arrhythmias. Pan Tompkins algorithm is used to detect the QRS complexes of an ECG signal, and heart rate variability analysis (HRV) is performed on the data to determine the patient's heart condition.

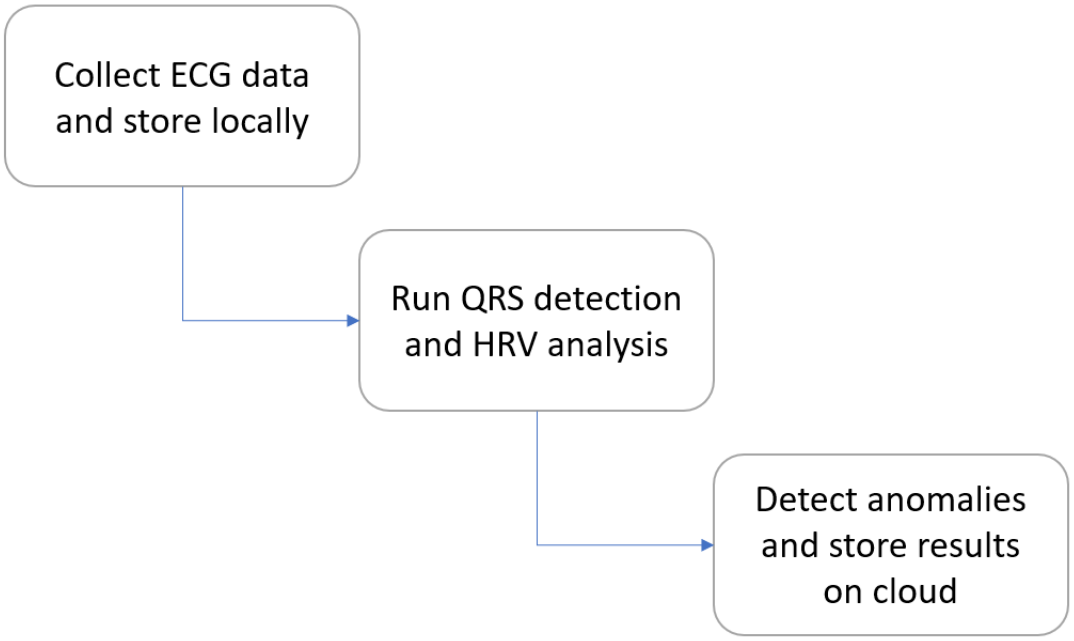

FIGURE 1.1: The basic project approach

A sample result is shown in Figure 1.2. The figure shows the output of the implemented Pan Tompkins algorithm, which has correctly identified the QRS complexes of the given input signal. The measured heart rate is also displayed.

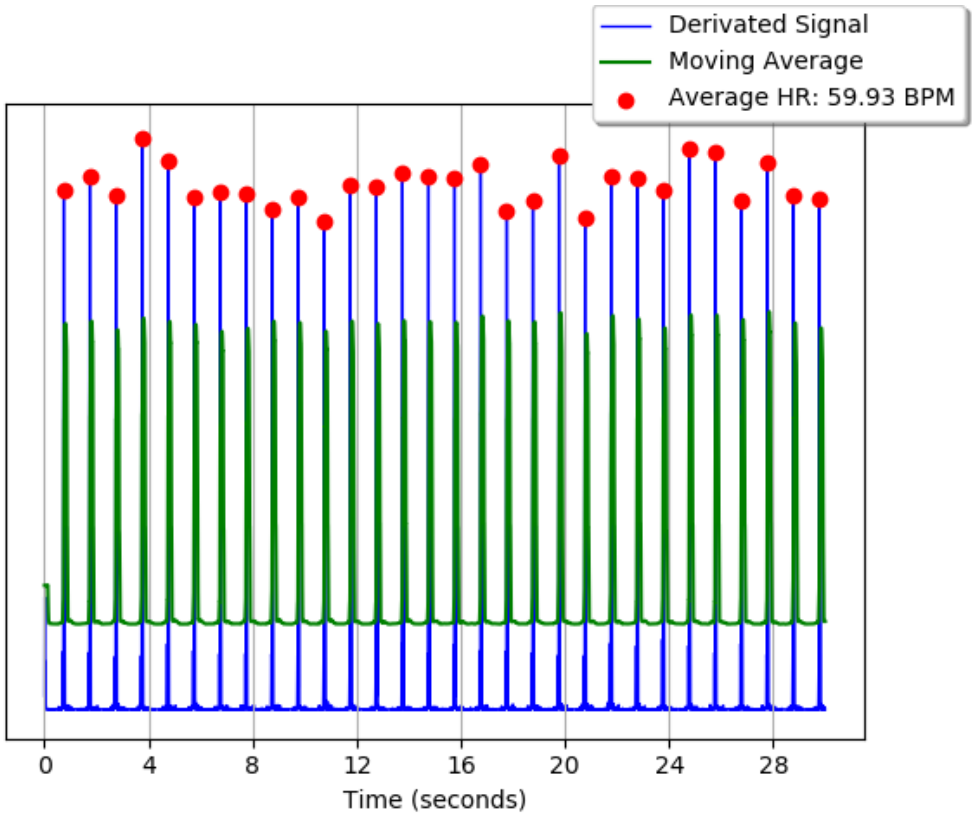

FIGURE 1.2: An example output from the Pan Tompkins algorithm, which detects QRS complexes.

Chapter 2

Background & Literature Review

2.1 Similar Work

2.1.1 Commercial Solutions

Traditionally, ECG monitoring is performed using senors that a patient wears for 24-48 hours, known as a Holter monitor. After this period, the physician would extract the data from the device and perform analysis.

Over the past few years, there has been an increase in the number of commercial heart monitoring systems. This can be largely attributed to advances in technology, allowing for low-power ultra-portable solutions for wearables, and the rise in popularity of the Internet of Things for connectivity.

MoMe Kardia: The MoMe Kardia system by InfoBionic is an automated cardiac monitoring ECG system developed in Massachusetts. The system performs remote

monitoring for patients, allowing physicians to view the ECG data and make a diagnoses. The device uses cellular data to upload directly to the cloud. The aim of the device is to speed up the diagnosis process by allowing the physician to diagnose remotely.

HeartCheck ECG Device: The HeartCheck ECG device is another portable commercial solution for ECG monitoring. It is a hand-held device, rather than using the traditional ECG leads, and is designed to be used to test for arrhythmias at regular intervals at different activity levels. The patient uses their hands to provide and ECG reading, and the device stores the data locally. The data must then be transferred manually to a computer where the physician can perform analysis.

Different to proposed system: The two aforementioned devices, as well as many other commercial devices, provide accurate ECG measuring. They do not however, provide an all-in-one solution to heart monitoring. In the case of the HeartCheck ECG DDevice the applications are limited since the data is stored locally and must be manually transferred and analysed. The patient also has to hold the device, so it isn't designed for continuous monitoring. The MoMe Kardia System is more similar to this project's design, in that it allows for remote monitoring. It does not, however, provide automated analysis of the ECG data.

2.1.2 Research-based Solutions

ECG monitoring and automated analysis has also been the focus of various research-based projects in recent years.

One such technique was developed by a research team at the Pukyong National University in South Korea, who created a non-contact system for ECG monitoring[3]. The system uses capacitive coupled sensors placed on an office chair, which collect ECG data from the patient non-intrusively. The system benefits from being less invasive, but the user is confined to the chair. Additionally, although the patient's data is uploaded to the cloud, only heart rate monitoring is performed.

Carelton University conducted research on an IoT-based remote patient monitoring ECG system[4]. The system uses an IOIO microcontroller to collect data from the patient using an ECG sensor, and a mobile device which connects to the board via Bluetooth. The mobile device acts as a tether to the internet and can also be used for visualisation of the patient's signal. The data is stored locally on an SD card. The system shares similarities to this project's design, in that it uses a small-form, low-cost hardware implementation for measuring and collecting the ECG data, but it lacks the ability to analyse the collected data and stream results to the cloud.

2.2 Internet of Things

One of the key aspects of this project revolves around the Internet of Things (IoT). The IoT is the idea of "anything that can be connected, will be connected". The idea was first introduced by Kevin Ashton [5] to describe a system where the physical world is connected to the internet using sensors. Since then, the IoT has grown tremendously in popularity, with the last decade seeing billions of new connected devices. The rise in popularity can be attributed to better networking infrastructure to wirelessly connect devices, the introduction of cloud computing for easily scaling processing resources and, most notably in the wearable industry is Moore's Law, which is an observation of the number of transistors on an integrated circuit doubling approximately every two years, resulting in small but powerful devices.

2.3 Heart Rate Monitoring

The two most common forms of patient cardiac monitoring are photoplethysmography (PPG) and electrocardiography (ECG). PPG is an optical based monitoring system, which detects blood volume changes by illuminating the skin and uses a photodetector to measure the variations in light absorption associated with the changes in blood perfusion. Many modern smartphones use PPG as the form of heart rate measurement.

PPG is a completely non-intrusive form of heart rate monitoring and is therefore considered more convenient to ECG, however it is less commonly used in the medical field. The reason for this is that ECG is more accurate due to directly measuring the electrical signals produced by the heart, whereas PPG uses electrical signals which are derived from reflected light changes due to heart activity. In addition, ECG data is also capable of delivering more heart metrics compared to PPG, so we can gain a better understanding of the patient's condition.

2.4 Electrocardiography

Electrocardiography (ECG) is a method of heart monitoring that measures the electrical activity of the heart, the tiny biopotential signals that are a result of the cardiac cycle. The first device capable of accurately measuring an ECG signal was a string galvanometer, invented by Dutch physician Willem Einthoven[6], and was the birth of clinical electrocardiography as we know it today. Einthoven's research built on work done by previous physicians and his discovery revealed the ECG signal we know today. He later won the Nobel prize in Medicine for his work, and his convention for the placement of ECG leads on a patient is still used.

The traditional method for ECG measuring is by using electrodes placed on the surface of the skin. The conventional 12-lead ECG places ten electrodes on the patient, and measures the amplitude of the electrical potential from twelve separate angles. The output ECG signal is an average of the individual cardiac muscle cell's potentials. Variations of the 12-lead ECG exist, including the 3-lead and

5-lead ECG. These variations provide less information compared to the standard configuration, but are sufficient for some types of heart rate monitoring, such as heart rate variability (HRV) analysis which is used in this project.

2.5 Electrocardiogram Analysis

An ECG signal is represented in the form of a PQRST waveform, where each letter represents a different section in the cardiac cycle. When discussing electrocardiography, we are mainly concerned with the upper and lower chambers of the heart - the atria and ventricles respectively. In the typical cardiac cycle, the following waveform components can be found in a patient's ECG signal:

- The P wave is due to the depolarisation of the atria of the heart.
- The QRS complex is a result of the depolarisation of the left and right ventricles. The increased amplitude of the QRS complex is due to the ventricles' larger muscle mass compared to the atria, thus producing larger biopotentials.
- The T wave represents the repolarization of the ventricles.

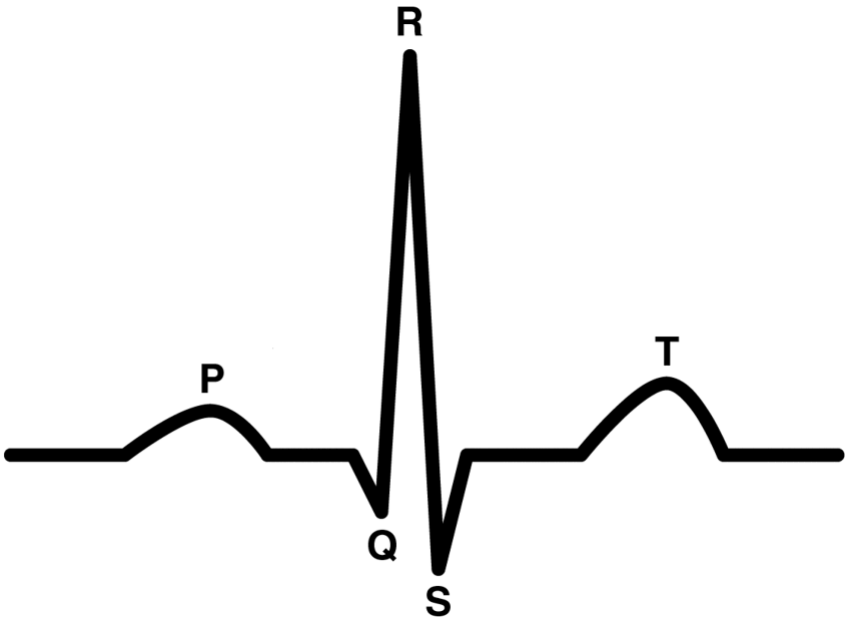

FIGURE 2.1: Representation of the various components of a typical ECG signal

Figure 2.1 shows the general shape of a normal ECG signal, however it is unrealistic due to the absence of any noise. A normal ECG tracing is usually corrupted with noise in some of the following three main ways:

- Baseline wander (BW) is a low-frequency noise artefact which is a result of movement or respiration of the patient.
- Power line interference is a high-frequency contamination due to electromagnetic fields from power lines or nearby equipment. The noise depends on the locality - the lines or mains frequency is 50Hz in Europe and 60Hz in the USA and parts of Asia.
- Electromyography (EMG) or muscle noise is caused by the contractions of other muscles near the heart, which is picked up by the electrodes of the ECG sensor.

Figure 2.2 shows the relative power spectra of an ECG signal. A typical signal is comporised of multiple sources, including noise artefacts.

Analysing the ECG signal can reveal a lot about a patient's health. ECG analysis can uncover abnormal heart rhythms, previous heart attacks, indications of cholesterol and more.

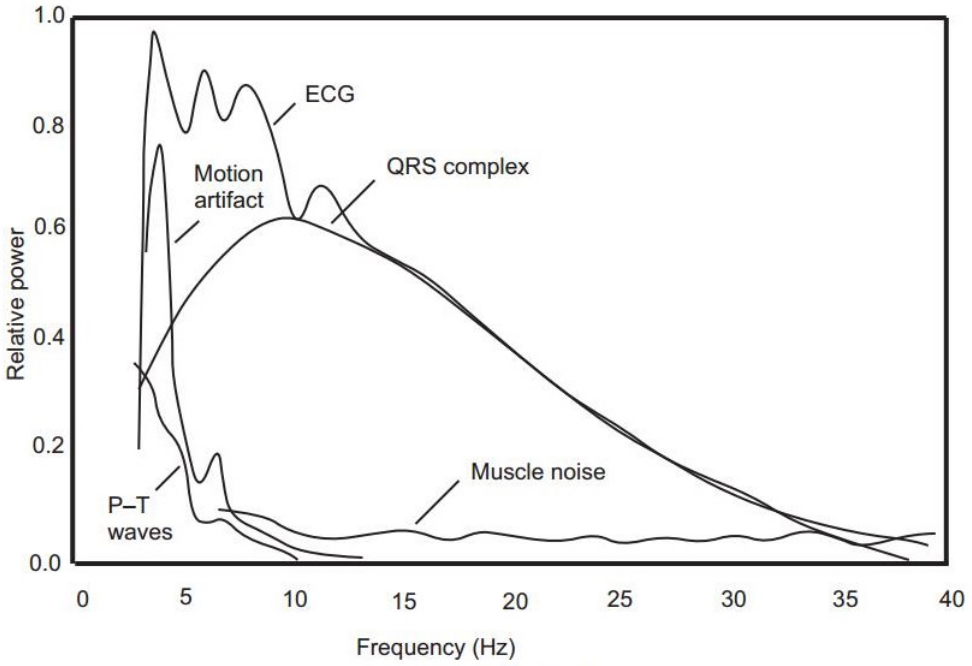

FIGURE 2.2: Relative power spectra of the ECG components as well as noise artefacts [1]

2.6 Heart Rate Variability Analysis

Although similar in name, heart rate and heart rate variability (HRV) are inherently different metrics when discussing or analysing heart health. A person's heart rate (measured in beats per minute or BPM) - or pulse - is a simple metric which measures the number of times their heart beats in a minute. Since it is an average,

[1] http://www.ems12lead.com/2014/03/10/understanding-ecg-filtering/

it ignores the variation in the intervals between successive beats. As an example, a heart rate of 60 BPM could mean one beat per second or could mean one beat every 0.5s, 1.5s etc.

Heart rate variability is an analysis of the beat-to-beat variation, and is the most common form of cardiac analysis. The interval between beats - known as the *RR* interval or inter-beat interval (IBI) - is measured in milliseconds and these intervals can be analysed to give a good overall indication of a person's heart health[7].

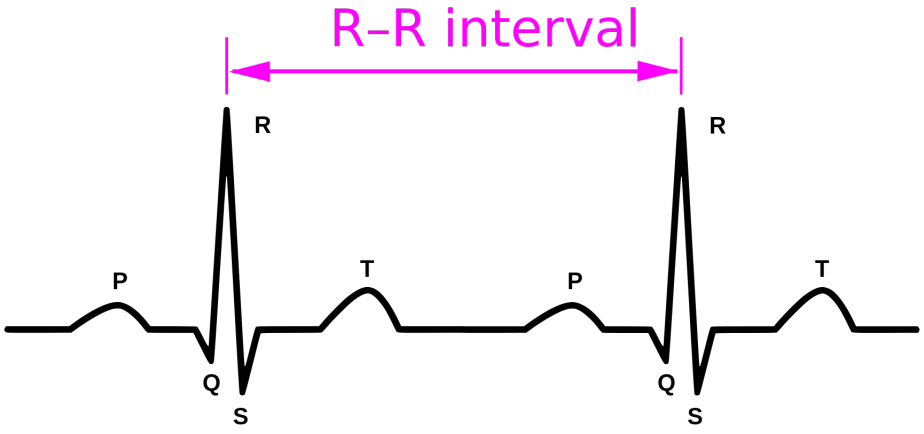

FIGURE 2.3: The RR or inter-beat interval

The autonomic nervous system contains parasympathetic and sympathetic nerves which affect a human's heart rate. Without the autonomic regulation, a resting heart rate would be around 100 beats per minute (BPM), known as the intrinsic heart rate. However, the parasympathetic nervous system (PSNS) lowers the

resting heart rate to around 70BPM. During intensive exercise, the sympathetic nervous system (SNS) increases the rate rate. Simply, the SNS controls the stimulation of the body's "fight or flight" stress response, and the PSNS controls stimulation of the body's "rest and digest" activities for recovery.

HRV is the analysis of the balance between the PSNS and SNS. A high HRV is an indication of a healthy autonomic nervous system and cardiovascular response, that the heart can effectively change rate depending on the activity level and indicates a greater cardiovascular fitness and more resilience to stress or disease. On the contrary, a low HRV indicates the PSNS and SNS are not properly balanced and that the patient is more susceptible to heart attacks, stress, strokes etc.

Various methods of heart rate variability analysis exist, and can mostly be separated into two main categories: time-domain analysis and frequency-domain analysis. Time-domain analysis methods are more commonly used in the medical field and are based on the beat-to-beat intervals of the ECG signal. In this project, analysis is done purely in the time-domain.

2.7 QRS Detection

The QRS complex is the most recognisable feature of a typical ECG signal. The steep slopes of the complex are a result of the increased amplitude compared to other sections of the cardiac cycle. Numerous algorithms and methods have been developed for the detection of QRS complexes.

A comparison of three QRS detection algorithms was done by Alvarez et al. (2002)[8], which included two algorithms based on digital filters - Pan Tompkins and Hamilton Tompkins, and another algorithm based on the phasor transform. The results of their comparison concluded that the Pan Tompkins algorithm had the overall best performance based on sensitivity - correctly identifying the most QRS complexes. The QRS detection implemented in this project is the Pan Tompkins algorithm.

Chapter 3

Methodology

3.1 Hardware

3.1.1 Arduino & Raspberry Pi

One of the first considerations for this project was deciding an appropriate hardware method of collecting and analysing ECG data. The three main features required were low-cost, low-power and small form-factor. The two most viable solutions found were the Arduino microcontroller and the Raspberry Pi single-board computer.

Raspberry Pis (RPI) are general all-purpose computers that run a Linux-based operating system and can run multiple different programs simultaneously, performing various tasks. Arduinos are much simpler, providing no graphical user interface since it just runs a single program over and over when it is powered on.

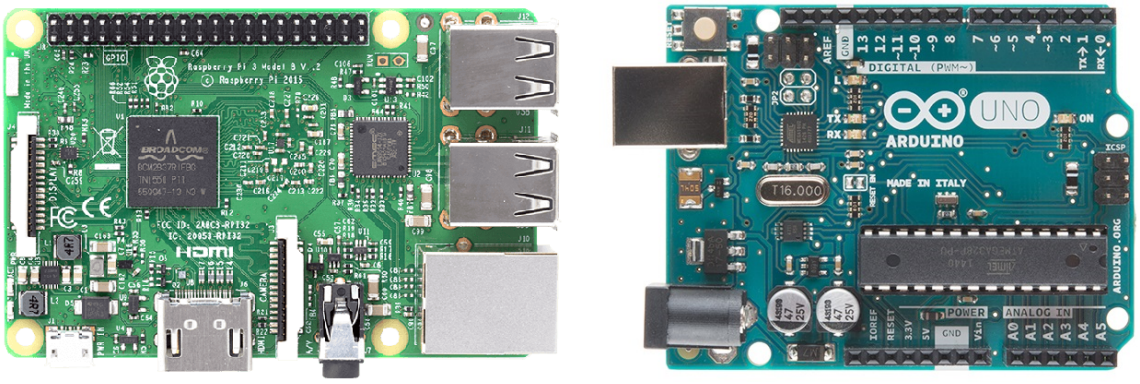

FIGURE 3.1: Raspberry Pi 3 (left) and Arduino Uno R3 (right)

The device would have to perform several tasks:

- Interface with the separate 3-lead ECG sensor.
- Sample the ECG signal at an appropriate rate.
- Analogue-to-digital conversion of the ECG signal.
- QRS peak detection on the signal.
- Heart Rate Variability (HRV) analysis.
- Upload the analysis results to the cloud.

Initially, an Arduino Uno was used to collect and sample ECG data. The Arduino features an on-board analogue-to-digital converter (ADC) and the use of hardware timers make sampling very accurate.

After consideration, the Arduino was removed in favour of the RPI. There are several reasons for this decision:

1. The Arduino runs one specific program when powered on, however this project requires several distinct tasks to be performed.

2. Arduinos can only use a set of C/C++ functions, so its functionality is somewhat limited. The RPI can run a much larger number of programming languages including Java, C/C++ and Python.

3. The Arduino Uno has just 512 bytes of storage, so a separate computer would be needed to store patient ECG data. RPIs have expandable storage using micro-SD cards so patient data can be stored and analysed locally.

3.1.2 Analogue-to-digital Converter

Raspberry Pis do not feature an on-board analogue-to-digital converter, so an external ADC must be used to read the ECG analogue values.

The MCP3008 ADC was chosen for this project, due to the low price and its easy compatibility with the Raspberry Pi. The ADC is capable of sampling at a rate of 200k samples per second, from 8 single-ended input channels. A 10-bit resolution means the digitised output ECG signal will be represented by values in the range 0 to 1023.

The ADC uses successive approximation for analogue-to-digital conversion. Successive approximation works by performing a binary search one bit at a time, beginning with the most significant (MSB). The output of the ADC is a ratiometric value, representing the ratio between the input and the reference voltage.

$$ADC_{out} = \frac{2^N \times V_{in}}{V_{ref}} \tag{3.1}$$

$$StepSize = \frac{V_{ref}}{2^N} \tag{3.2}$$

The reference voltage, V_{ref} is 5V and 2^N is the resolution for the ADC, 1024 in this case, so the step size is 4.88mV.

Communication between the ADC and the Raspberry Pi is achieved using the SPI protocol.

3.1.2.1 Serial Peripheral Interface

The serial peripheral interface (SPI) bus is a synchronous serial communication interface commonly used for embedded systems, microcontrollers and integrated circuits.

The hardware SPI bus uses four GPIO pins on the RPI for synchronous communication, *MOSI, MISO, SCLK* and *CE*. MISO and MOSI are used for transferring data to and from the master and slave (RPI and ADC respectively in this case). SCLK is the serial clock, to keep the master and slave in sync. CE is the chip enable signal, sometimes known as chip or slave select, and is used by the master to select which slave to interface with at any one time.

The RPI allows any of the GPIO pins to be redefined as the SPI pins. This software SPI configuration is useful for multiple SPI slaves, but can lead to a high CPU load. For this reason, the dedicated SPI GPIO pins are used for a hardware SPI configuration.

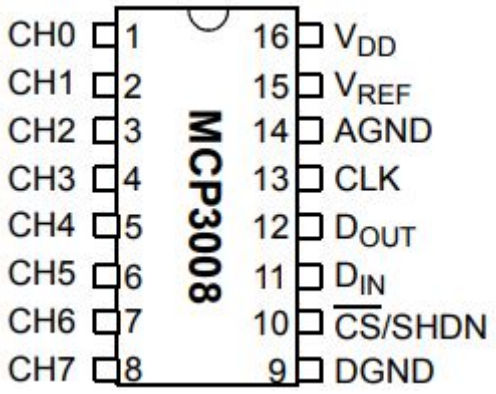

FIGURE 3.2: MCP3008 ADC

3.1.3 ECG Sensor

A third-party ECG sensor was used to measure the heart's electrical activity. The sensor used is the 'Gravity: Analog Heart Rate Monitor Sensor (ECG)' by DFRobot. The low-cost sensor uses 3-leads to acquire the ECG signal from the patient.

The sensor module uses an AD8232 chip. The AD8232 is an integrated signal conditioning block specifically designed by Analog Devices for biopotential measurement applications. The conditioning block is used to filter and amplify ECG signals in the presence of noisy conditions, removing unwanted artefacts that are a result of muscle noise (EMG) or baseline wander. Most low-cost ECG sensors use the AD8232 conditioning block for acquiring a clean ECG signal.

Figure 3.3 shows the placement of the electrodes for a 3-lead ECG sensor. The three limb electrodes - denoted I, II and III - form an Einthoven's triangle at the right arm (RA), left arm (LA) and left leg (LL).

The leads are bipolar, meaning each has a positive and negative pole. Each lead gives the potential difference between two limbs:

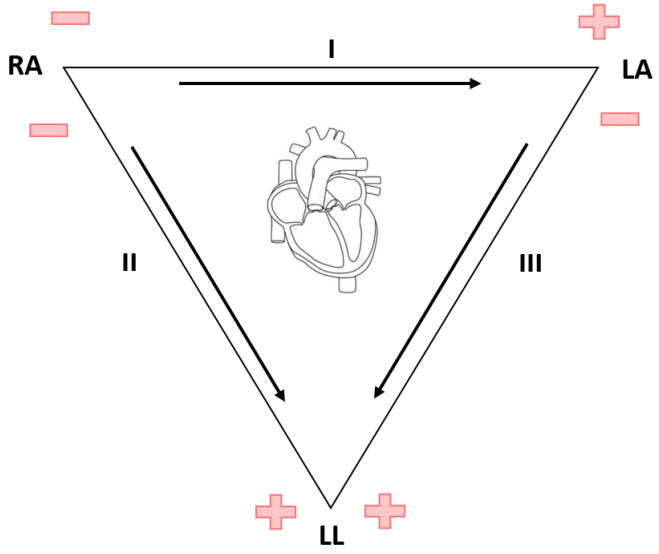

FIGURE 3.3: Placement of electrodes for a 3-lead ECG sensor relative to the heart

- **Lead I:** between the right and left shoulders.

$$I = LA - RA \tag{3.3}$$

- **Lead II:** between the right arm and left leg.

$$II = LL - RA \tag{3.4}$$

- **Lead III:** between the left shoulder to the left leg.

$$III = LL - LA \tag{3.5}$$

3.2 Software

This section describes the basic program flow and structure. A more detailed description outlining the implemented algorithms is given later.

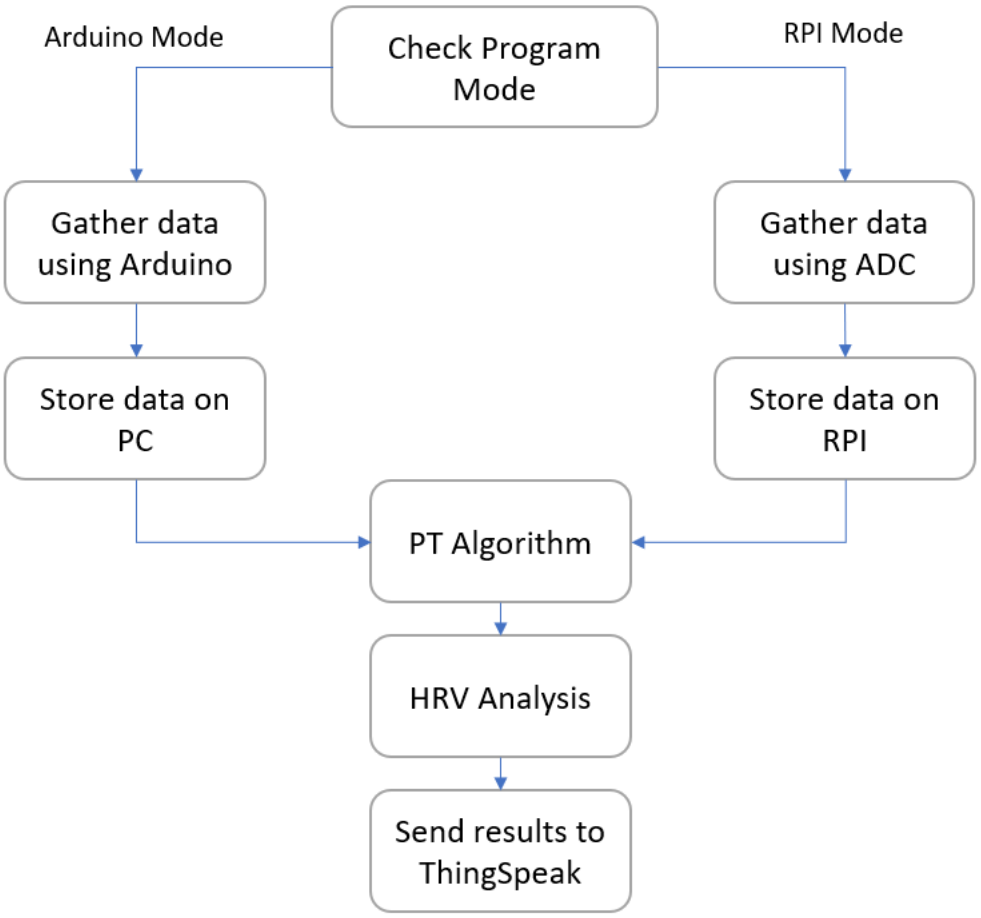

FIGURE 3.4: Program Flow

Initialising the program is done by running the *main.py* file. This allows the user to choose a sampling rate, the period to record for, and the mode in which the program will run. Although this project is focused mainly around the Raspberry Pi, an additional mode allows the user to use an Arduino via the serial port. A final debugging mode was included which allows the user to run the program on a

set of pre-recorded ECG data. This debugging mode was especially helpful during the implementation of the HRV analysis to ensure the results obtained matched the expected results, although this option increases the potential applications for this project as well. Henceforth in this section I will refer only to the main mode of operation, using the Raspberry Pi.

A *csv* (comma separated values) file is created by the program to store the collected ECG data. The filename is automatically generated using the selected mode, the duration of the recording, as well as the date and time at which the program is run. Analysis results will decide whether the collected data is important and needs to be stored permanently.

The program then calls the C program that is used to collect sample values from the ADC. This is achieved by running the executable using a shell command in Python, using the *os.system* function which includes the filename as a command line argument. The decision to use C for this task is outlined in the results section of this report.

Once the recording time has elapsed, the program indicates the number of samples obtained, the recording time and the sampling rate achieved to ensure accuracy. The ECG data is then stored in the *csv* file.

The data file is opened and each line (sample) is appended to a list. Some pre-processing is performed before any analysis is done. The data is scaled, altering the range to be between 0-1023. This data normalisation allows different data sets or different ECG devices to be used. A Python pandas dataframe is created,

which allows the ECG data at each stage of processing to be stored together for convenience.

The program then performs the Pan Tompkins algorithm. The *run_pan_tomp* function is called inside the *pan_tomp.py* file, which runs each of the individual processing steps involved. The results of the algorithm - the locations of the detected *QRS* peaks and the measured RR intervals - are stored in a Python dictionary.

Next, HRV analysis is performed by calling the *run_hrv_analysis* function inside the *heart_rate_variability_analysis.py* file. The inter-beat intervals stored in the results dictionary are used here for analysis, and the results obtained are added to the dictionary.

Finally, the program checks the results by calling the *keep_or_delete_data* function within the *analyse_results.py* file, and makes a decision regarding deleting the original ECG data. The results of the HRV analysis are uploaded to a ThingSpeak channel using the *update_channel* function in the *to_thingspeak.py* file.

3.3 Pan Tompkins

Identification of the QRS complex of the ECG signal was required to perform heart rate variability analysis.

The algorithm used in this project for ECG feature classification is the Pan Tompkins[9] algorithm. Pan Tompkins is a real-time, derivative-based algorithm

developed in 1985 for the detection of QRS complexes in ECG signals. Pan Tompkins algorithm was chosen as it very accurate, computationally efficient and well-documented.

Pan Tompkins algorithm was implemented in Python for this project. The algorithm recognises QRS complexes based on analysis of the slope, amplitude and width. It was originally implemented in assembly language for the NSC800 and Zilog Z80 8-bit microprocessors using integer arithmetic. Processing power has obviously increased dramatically in the 33 years since it was originally developed, but the processing steps are still used as the basis of many ECG feature detection algorithms and implementations today. The signal is first processed in order to make a more accurate decision of the location of each QRS complex - removing any noise artefacts, smoothing the signal and amplifying the QRS slope.

The processing steps in the Pan Tompkins algorithm are as follows:

1. A bandpass filter is used to reduce the interference of noise in the ECG signal, described in the background section of this report. The filter is implemented in the form of a cascaded low-pass (LFP) and high-pass (HPF) filter. The transfer function for the second-order LFP and the first-order HFP are given as:

$$H(z) = \frac{(1 - z^{-6})^2}{(1 - z^{-1})^2} \tag{3.6}$$

$$H(z) = \frac{(1 + 32z^{-16} + z^{-32})}{(1 + z^{-1})} \tag{3.7}$$

Difference equations are used for digital filtering operations. In discrete-time systems, filter transfer functions can be converted to a linear constant-coefficient difference equation (LCCD), which computes the value of an output sample based on present and past input samples, and past output samples. The difference equations for the LFP and HPF are given as:

$$y(nT) = 2y(nT-T)-y(nT-2T)+x(nT)-2x(nT-6T)+x(nT-12T) \quad (3.8)$$

$$y(nT) = 32x(nT-16T) - [y(nT-T) + x(nT) - x(nT-32T)] \quad (3.9)$$

2. The signal is then differentiated to identify the large slopes associated with the QRS complexes of a normal ECG signal. This process suppresses the low-frequency components of the ECG signal, while amplifying the high-frequency components. The result is an ECG signal with attenuated P and T waves, with amplified QRS complexes. The five-point derivative has a transfer function and difference equation:

$$H(z) = (\frac{1}{8T})(-z^{-2} - 2z^{-1} + 2z^{1} + z^{2}) \quad (3.10)$$

$$y(nT) = (\frac{1}{8T})[-x(nT-2T)-2x(nT-T)+2x(nT+T)+x(nT+2T)] \quad (3.11)$$

3. The squaring operation provides non-linear amplification of the signal. The ECG signal is squared point-by-point, resulting in a signal that is all positive

and emphasises the high frequencies found in the QRS complex.

$$y(nT) = [x(nT)]^2 \tag{3.12}$$

4. A moving-window integrator is used extract features in addition to the slope of the R wave. The window size - ie the number of samples in the window - should be approximately equal to the widest possible QRS complex. By choosing an appropriate window size, the aim is to produce a single peak which represents a QRS complex. A larger than necessary window will merge the QRS and T complex, and a small window will produce more than one peak per QRS. The moving-window integrator is calculated using the following formula, where N is the number of samples in the window:

$$y(nT) = \frac{1}{N}[x(nT - (N-1)T + x(nT - (N-2)T) + ... + x(nT)] \tag{3.13}$$

5. Finally, some thresholding is performed and a decision is made as to the location of a QRS peak. Two thresholds are used and a detected peak is classified as a noise peak or a signal peak. A searchback technique is used when the algorithm decides no QRS complex has been detected for longer than a specific interval.

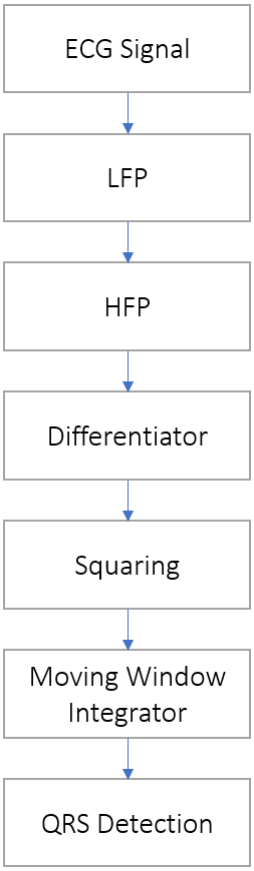

FIGURE 3.5: Block diagram of Pan Tompkins algorithm

The implementation of Pan Tompkins algorithm for this project was done in Python, and follows the same general processing steps outlined above. A band-pass filter, in the form of a low-pass cascaded with a high-pass, is used to filter the noisy input data. Some preemptive filtering is done by the AD8232 signal conditioning block to remove some unwanted noise, but the bandpass filter is used anyway to reject frequencies outside the desired range. A 2_{nd} order low-pass and 2_{nd} order high-pass digital Butterworth filters attenuate frequencies outside the range 5-15Hz when sampling at 200Hz, as outlined by Pan & Tompkins.

The signal is then derivated. This procedure involved convolving the data signal with a filter impulse response, that has the following transfer function:

$$H(z) = \frac{1}{8}(-z^{-2} - 2z^{-1} + 2z^{1} + z^{2}) \tag{3.14}$$

The signal is squared point by point for non-linear amplification. A moving average, or rolling mean of the signal is calculated. The size of the moving window is the number of samples used in each window, and was chosen to approximately 15% of chosen sampling rate. This ratio is the same as that outlined by Pan and Tompkins, who chose a window size of 30 samples for a sampling rate of 200Hz.

The moving average signal is used as a threshold for detecting QRS complexes. The algorithm goes though each value in the derivated signal and checks if the amplitude is greater than the moving average signal at that time, which indicates a peak is detected. The position of all detected R-peaks are stored in the results dictionary.

Finally, the RR intervals are calculated using the positions of the detected peaks. The interval is measured by getting the number of samples between each pair of successive identified peaks and converting the interval, measured in samples, into a time interval measured in milliseconds.

$$\text{Interval in samples} = \text{peak location}_{n+1} - \text{peak location}_{n} \tag{3.15}$$

$$\text{Interval}_{ms} = \frac{\text{Interval in samples}}{\text{sampling frequency}} \times 1000 \tag{3.16}$$

3.4 Heart Rate Variability Analysis

The result of implementing the Pan Tompkins algorithm is a stream of impulses representing the *QRS* complexes of the ECG signal. The peaks are annotated and the interval between each peak - known as the *RR* intervals - are calculated and stored in a list. Using this data, heart rate variability (HRV) analysis can be performed on the patient data to gain an understanding of the patient's cardiovascular health.

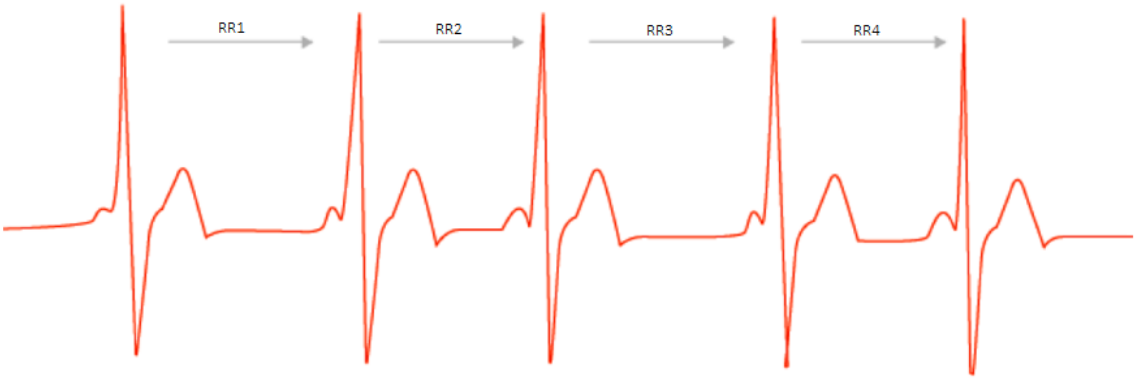

FIGURE 3.6: Intervals between heartbeats - RR intervals or inter-beat intervals

The heart rate is the first metric that is calculated. The mean of the *RR* intervals is calculated and converted from milliseconds to a 60 second rate to get the beats per minute. The formula is given below, where N is the number of RR intervals measured (ie the number of detected peaks - 1).

$$HeartRate(bpm) = \frac{60000ms \times N}{RR_1 + RR_2 + ... + RR_N} \tag{3.17}$$

HRV analysis for this project is performed solely using time-domain methods. Time-domain analysis is concerned with the beat-to-beat intervals of the ECG

signal. It must be noted that in HRV analysis, the notation NN is often used to denote the interval between successive QRS peaks, which I have denoted as RR intervals thus far. The NN notation is used to indicate that the beats or R-peaks are normal.

Several HRV metrics are calculated as part of the analysis section of this project, using the interbeat intervals outlined above.

The root mean square of successive differences (RMSSD) measures the average interval between successive QRS peaks in the ECG signal, and is the most commonly used metric to access a patient's heart health.

$$RMSSD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(RR_{i+1} - RR_i)^2} \tag{3.18}$$

The standard deviation of intervals (SDNN) is a metric which measures standard deviation of the normal NN intervals of an ECG signal. It is often referred to as the "gold standard" of cardiac monitoring. It should be noted that the SDNN notation is usually used when measuring for a period of 24-hours, while SDANN is used for shorter 5-minute periods.

$$SDNN = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(RR_i - \overline{RR})^2} \tag{3.19}$$

NNx is a HRV metric first derived from a study by Ewing et al. (1984) [10], which measures the number of times in which the change in successive NN intervals exceeds a certain predefined number of milliseconds, x. It follows that pNNx is

the proportion - usually represented as a percentage - of all intervals that exceed this threshold. The metric is generally used for intervals of 50ms, however a study by Mietus et al. (2002)[11] found that thresholds of 20ms or lower provided a better separation between normal results and those indicating cardiovascular disease.

$$NNx = \sum_{i=1}^{N} (|RR_{i+1} - RR_i| > x\ ms) \tag{3.20}$$

$$pNNx = \frac{NNx}{N} \times 100 \tag{3.21}$$

The results of the heart rate variability analysis are stored in a Python dictionary. The dictionary uses the HRV metric as the key, with the metric result as the value. The HRV results are stored in a *csv* file which has the date and time-stamp for record keeping purposes.

3.5 Analysing Results & File storage

Once the heart rate variation analysis has been performed, the results are examined. The purpose of this is to uncover any arrhythmias or abnormalities in the patient's data. The results of the HRV analysis are compared with a list of HRV metrics and norms. The list contains standard ranges for each of the evaluated metrics. If the analysis results fall outside any of these standard ranges, it indicates an underlying issue in the patient's health. A table of HRV norms is given below, with the values being the standard for that of an 20-29 year old male - ie

the author of this report. These ranges are an approximation, and are based on a clinical study by Umetani et al. (1998)[12].

HRV Metric	Normal Range
SDNN (ms)	100-250
SDANN (ms)	80-240
RMSSD (ms)	20-90
pNN50 (%)	5-25
Heart Rate (bpm)	60-100

TABLE 3.1: Approximate ranges for normal HRV results of a 20-29 year old male.

If the patient's results are found to be typical for his/her age and activity level, and there is no indication of any arrhythmias or cardiovascular diseases, the original patient ECG data is automatically deleted. This setting is used as a space-saving method on the Raspberry Pi due to limited storage capacity, but can easily be turned off at the physician's discretion.

If the results are found to be abnormal, the original data is stored locally and an indication as to the diagnosis or condition is reported to the user.

3.6 Cloud & ThingSpeak

The results of the analysis are sent automatically to the cloud. The cloud solution used for this project is ThingSpeak, an open-source IoT platform for collecting and analysing sensor data. A private channel is used for displaying the ECG results.

The channel has several fields, one for each of the HRV metrics analysed, and can all be updated simultaneously for visualisation of the patient results.

Sending data to ThingSpeak involves the use of an application programming interface (API) key, used for authentication purposes. A Python dictionary is used to update the channel, where the keys are each field and values are the type of metric to upload. The *urlopen* function in the *urllib.request* module is used to open the *baseurl*, containing the channel address and the API key. This URL is appended with the results for each field, and all fields are updated simultaneously with one request.

```
API_KEY_WRITE = 'QVN6DOKQAI2WK4MR'
channels = {'field1': 'bpm', 'field2': 'rmssd', 'field3': 'sdnn', 'field4': 'pNNx'}
baseurl = 'http://api.thingspeak.com/update?api_key=' + API_KEY_WRITE

def update_channel(results):
    data_to_update = ''
    for channel in channels:
        data_to_update += ('&' + str(channel) + '=' + str(results[channels[channel]]))
    f = urllib.request.urlopen(baseurl+str(data_to_update))
    f.read()
    f.close()
```

Using a cloud service increases the applications for this project. By having the results upload directly to the cloud automatically, remote monitoring can be performed by the cardiologist.

Chapter 4

Results & Discussion

It should be noted that, for the majority of the results obtained, an ECG signal generator was used. The SKX-2000 ECG signal generator generates an analogue ECG signal at an adjustable heart rate and can be used in a 3-lead, 5-lead or 12-lead configuration. The signal generator was used in its 3-lead configuration for this project, and allowed the author to perform analysis and debugging effectively, since the expected result was known.

FIGURE 4.1: SKX-2000 ECG signal generator

4.1 Hardware

An initial configuration was set up to test the ECG sensor reliability. This configuration used an Arduino Uno to perform analogue-to-digital conversion via the on-board ADC, and to display the results. The Arduino was initially used to be able to isolate the testing of the sensor, without using the external MCP3008 ADC.

The Arduino communicated via serial communication and the ECG signal was displayed on the serial plotter on the Arduino IDE (integrated development environment). Figure 4.2 shows the serial plotter window which displays the ECG signal. The signal displays the clear signs of an ECG signal, with an easily distinguishable QRS complex.

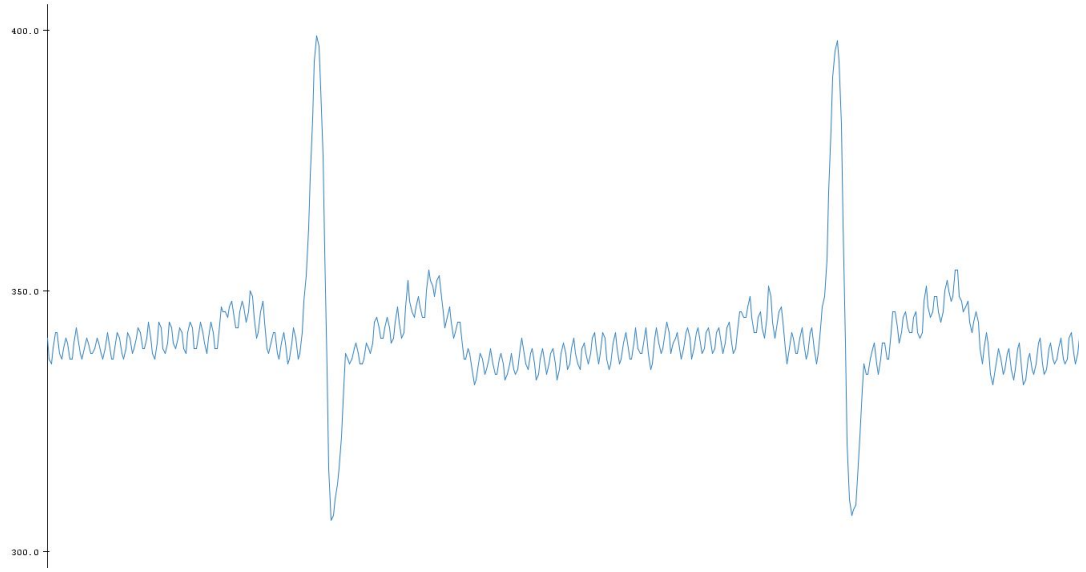

FIGURE 4.2: Arduino serial plotter

The next step was to examine the results obtained when sampling using the Raspberry Pi and MCP3008 ADC. Again, the hardware configuration provided an

accurate ECG signal. The 10-bit ADC samples the analogue signal at the chosen sampling rate, performing successive approximation ADC to convert the measured analogue ECG signal values into discrete integer values between 0-1023. Figure 4.3 shows a section of a recorded ECG signal using the RPI and ADC.

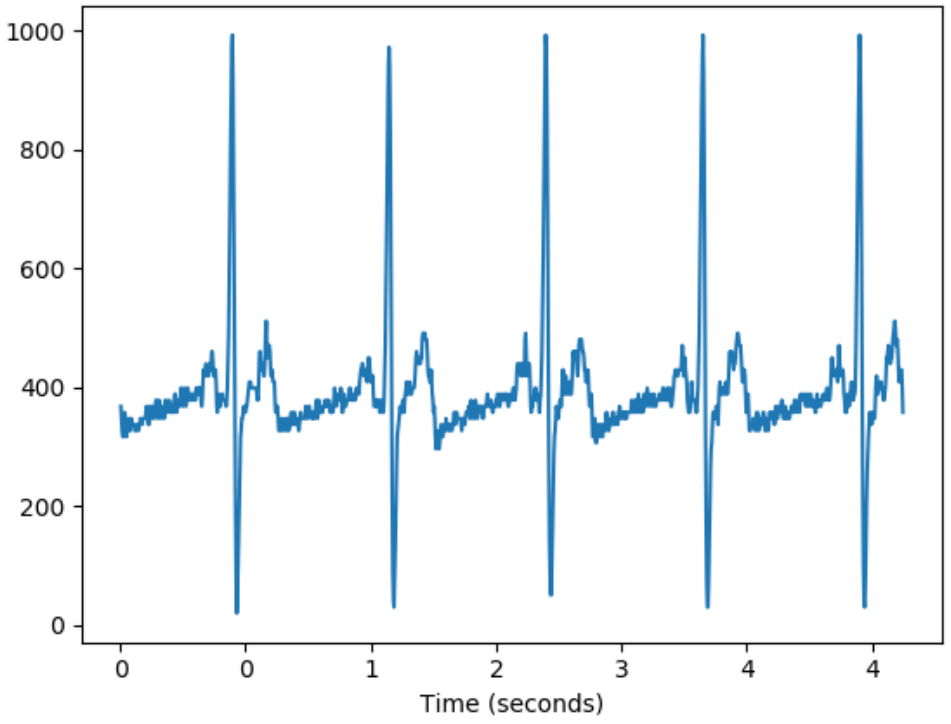

FIGURE 4.3: Segment of an input ECG signal, after normalisation

It was found, however, that during development of the heart rate variability analysis code that this method of sampling was inadequate. Heart rate analysis, which includes calculating the heart rate of the given ECG data, revealed that the results were inconsistent and varied somewhat randomly. For a known fixed rate ECG signal provided by the signal generator, the calculated results would often be slightly higher than expected when using the Raspberry Pi and MCP3008 configuration - see Figure 4.4 - but were as expected when using the Arduino.

The reason for this was found to be the method of sampling used. When using the Raspberry Pi, Python code was originally used for sampling purposes. The Arduino uses hardware timers which results in accurate time intervals between sampling. Additionally, the Arduino is only responsible for running a single piece of code at any one time. However, the Raspberry Pi could be performing several tasks at any one time, with the processor being interrupted continually. Python was determined to be the main cause of this inconsistency. Python is a high-level programming language and can be performing any number of memory management tasks in the background, interrupting the control flow of the program.

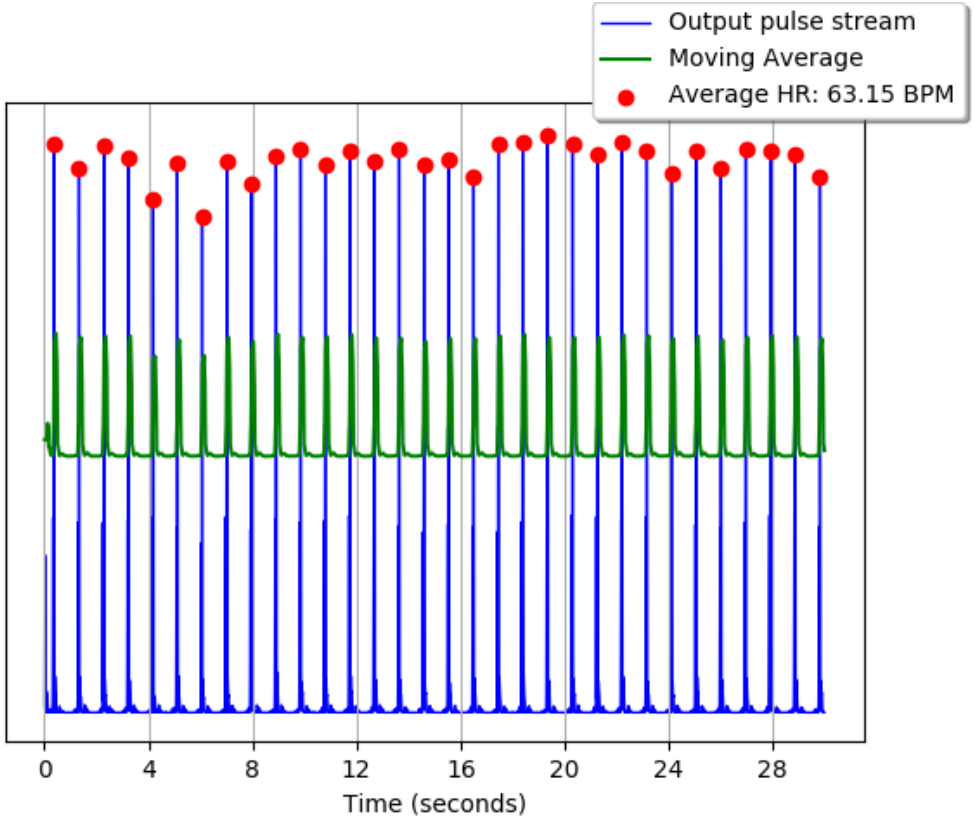

FIGURE 4.4: PT output at fixed input rate of 60BPM when sampling using Python

It was therefore decided to rewrite the sampling process in C, a lower-level language, for a more reliable method of sampling at a fixed rate. The new method

uses the 'pigpio' library to allow control over the General Purpose Input Output (GPIO) pins of the RPI. It uses direct memory access (DMA) to interface with the MCP3008 ADC via SPI, which allows for consistent sampling intervals within approximately a microsecond. The results obtained using this new method are shown below in figure 4.5 and are much more consistent with the expected results.

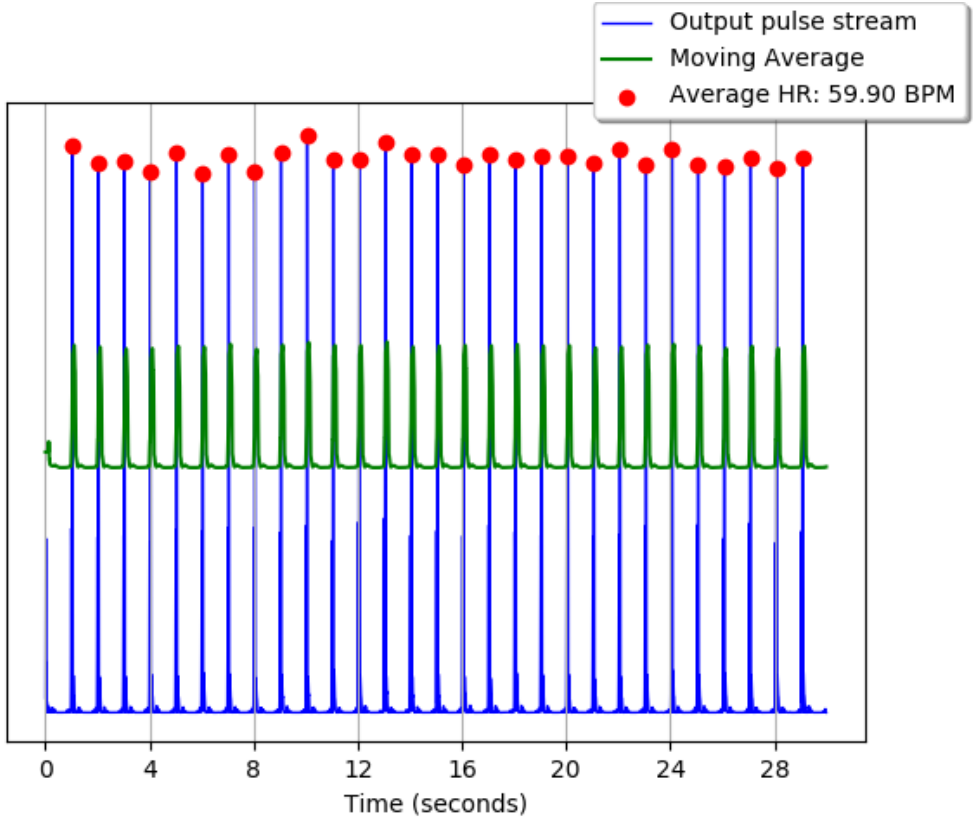

FIGURE 4.5: PT output at fixed input rate of 60BPM when sampling using C

4.1.1 Sampling Rate

A number of sampling rates were tested and analysed to determine if they were suitable. The concern here was the trade-off between accuracy and processing time. Ideally, a very high sampling rate would be used as it would yield the most

accurate ECG signal, with a large number of samples to represent the continuous-time ECG signal as a discrete-time signal, where each discrete sample is an ADC digitised integer. However, since one of the key aspects of this project was the cost-effectiveness, the aim was to reduce the processing power required. Thus, several sampling rates were tested to evaluate both the processing time required and the effect on the results of analysis. Pan Tompkins algorithm took approximately 57 seconds to run on an hour of ECG data sampled at 250Hz.

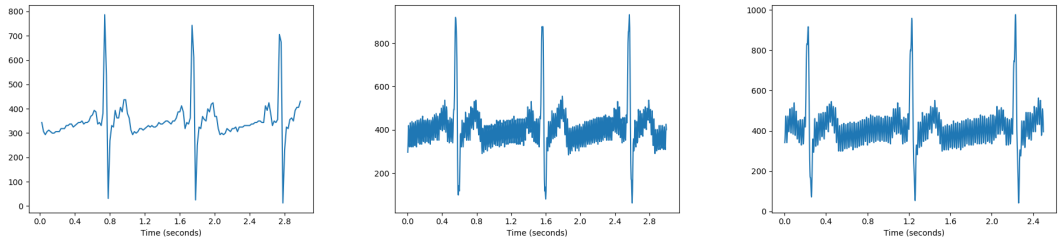

FIGURE 4.6: Comparison of different sampling rates, from left to right: 50Hz, 250Hz, 500Hz.

As seen in Figure 4.6, there is little difference between sampling at 250Hz and 500Hz on the ECG signal. Sampling at 50Hz however provided inconsistent results due to the undefined peak amplitudes, as seen in Figure 4.7.

These conclusions are similar to results obtained by Pizzuti et al (1985)[13], who found no significant difference between sampling at 250Hz and 500Hz, but that recordings obtained at 125Hz or less displayed a significant reduction in peak amplitudes. A study performed by Mahdiani et al. (2015)[14] found that sampling rates as low as 50Hz cause distortion especially in the R-peak waveform of an ECG signal, which could compromise results.

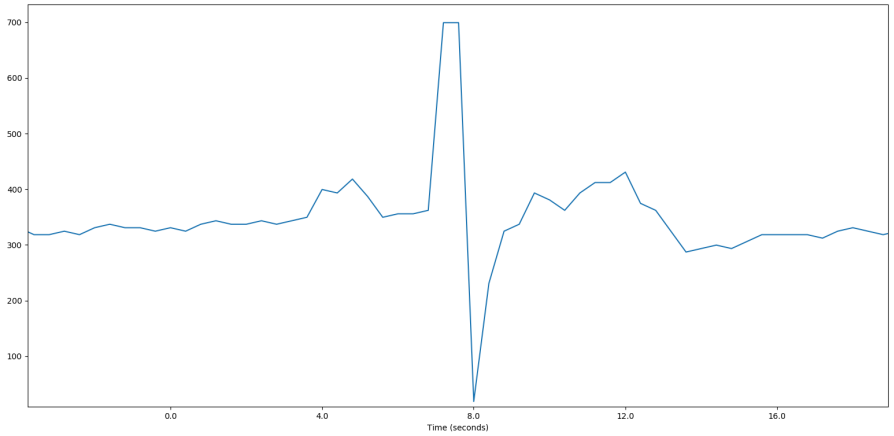

FIGURE 4.7: Input signal sampled at 50Hz

4.1.2 Cost Evaluation

It is important to remember that one of the key focuses of this project is developing a cost-effective solution. It is reasonable therefore, to assume that the accuracy of the results obtained would be improved with a larger spending budget. However given that the hardware used is mostly hobbyist-oriented - the sensor product webpage explicitly states that it is not a medical device and should not be used as such - the results obtained seem quite satisfactory. A price list of the various components used is given in Table 4.1 and is accurate at the time of writing.

Component	**Price (€)**
Raspberry Pi	36.25
ECG Sensor	17.11
MCP3008 ADC	3.55
	€59.61

TABLE 4.1: Breakdown of prices for project components

A picture of the entire hardware setup, including the Raspberry Pi, MCP3008, ECG sensor and ECG signal generator is given in Appendix A in Figure A.1.

4.2 ECG Live View

During the writing of this report, interest was expressed in devising a method of visualising the ECG signal in real time. Although the signal is displayed after processing has been completed, a mode for visualising the ECG signal similar to an ECG monitor in a hospital was deemed to be a necessity for a practical ECG device. Thus, a 'live-view' mode was created which uses the *matplotlib.animate* class to continually update a *pyplot* graph, displaying a live view of the patient's ECG signal. Figure 4.8 shows this new mode in action.

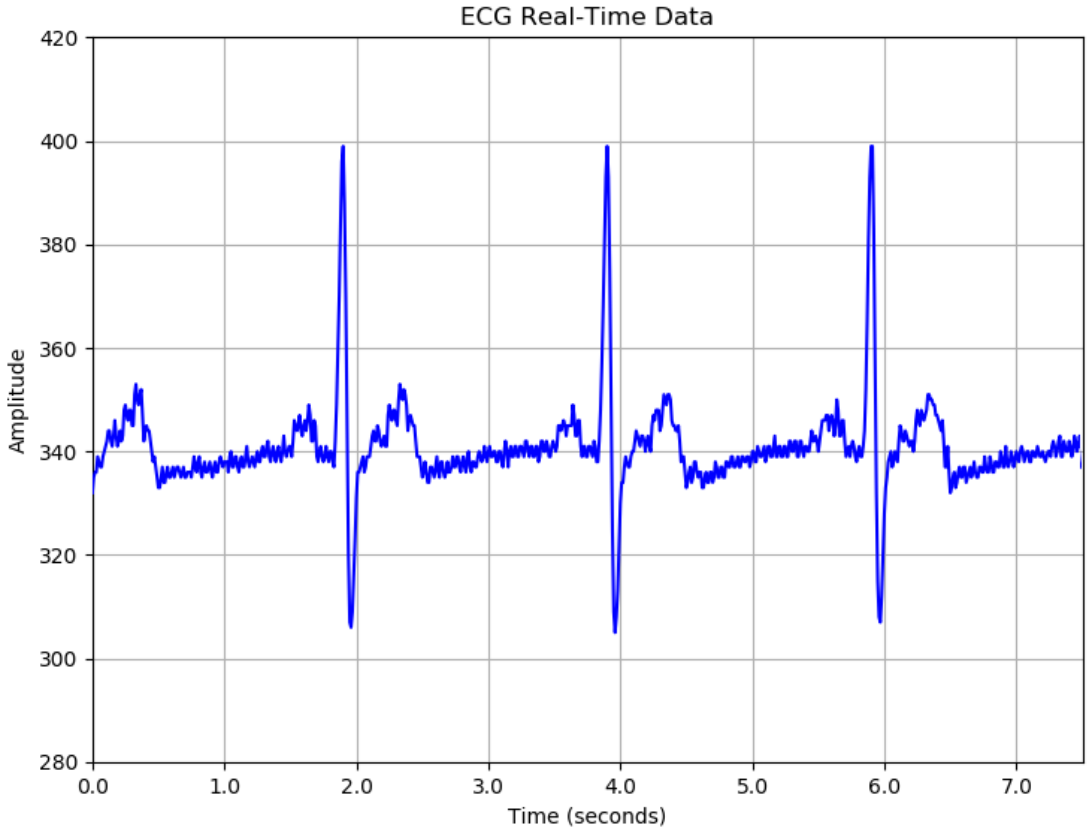

FIGURE 4.8: Live view of ECG signal

4.3 QRS Detection

What follows are the results of the Pan Tompkins algorithm at each intermittent processing stage. The input signal produces by the ECG signal generator was a fixed rate of 60BPM, and was sampled at 200Hz. A segmented section of 800 samples - which shows four consecutive sinus rhythms - was chosen to display the results of each stage of the algorithm. Note the Y-axis for the following figures (4.9, 4.10, 4.11, 4.12, 4.13 and 4.14) is the amplitude of the ECG signal at the various processing stages, which changes scale and units throughout. Since this project uses HRV to analyse the ECG data, and since HRV is based solely

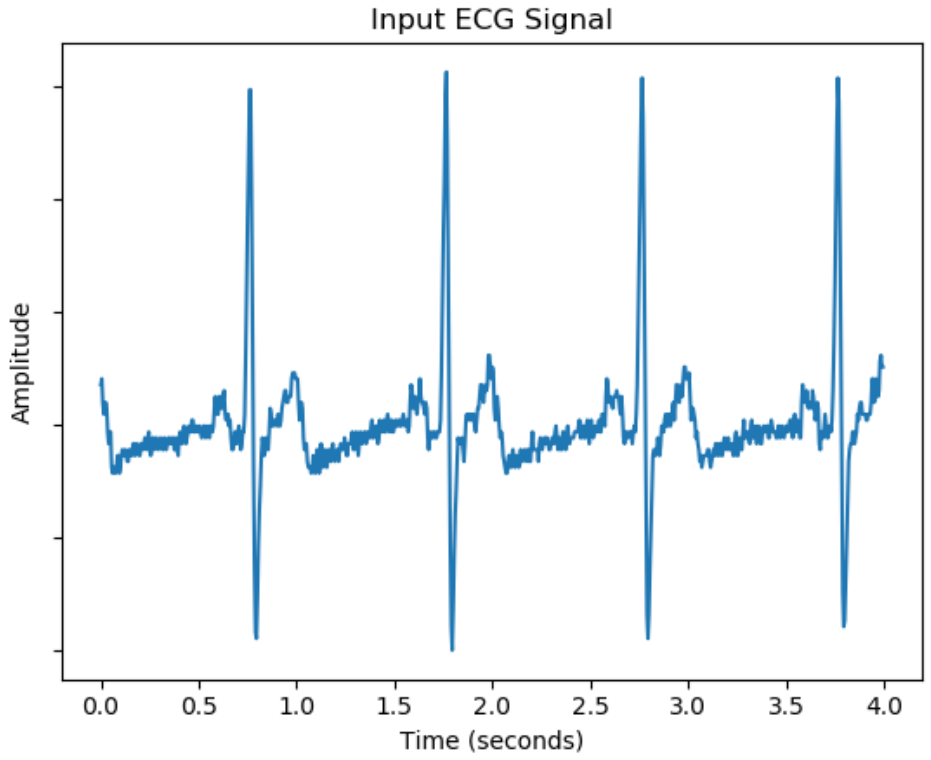

FIGURE 4.9: ECG input signal.

on analysis of the inter-beat intervals and not on the amplitude of the various waveform components, the amplitude scale on the graphs were removed in favour of a more simplistic looking result.

The intervals between each successive identified peak are calculated.

As shown in Figure 4.14, the output of the implementation of Pan Tompkins algorithm is the successful detection of the *QRS* complexes of an ECG signal, where the highest amplitude point, the *R*-peak, is annotated. Doing this allows the user to easily distinguish detected and non-detected peaks.

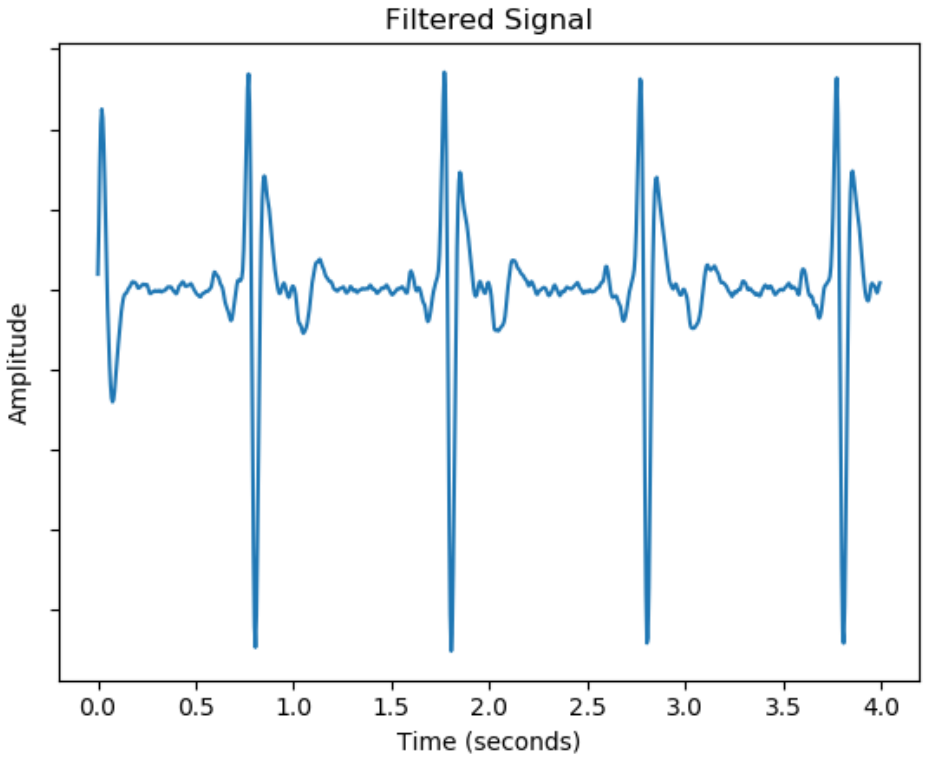

FIGURE 4.10: ECG signal after removing noise.

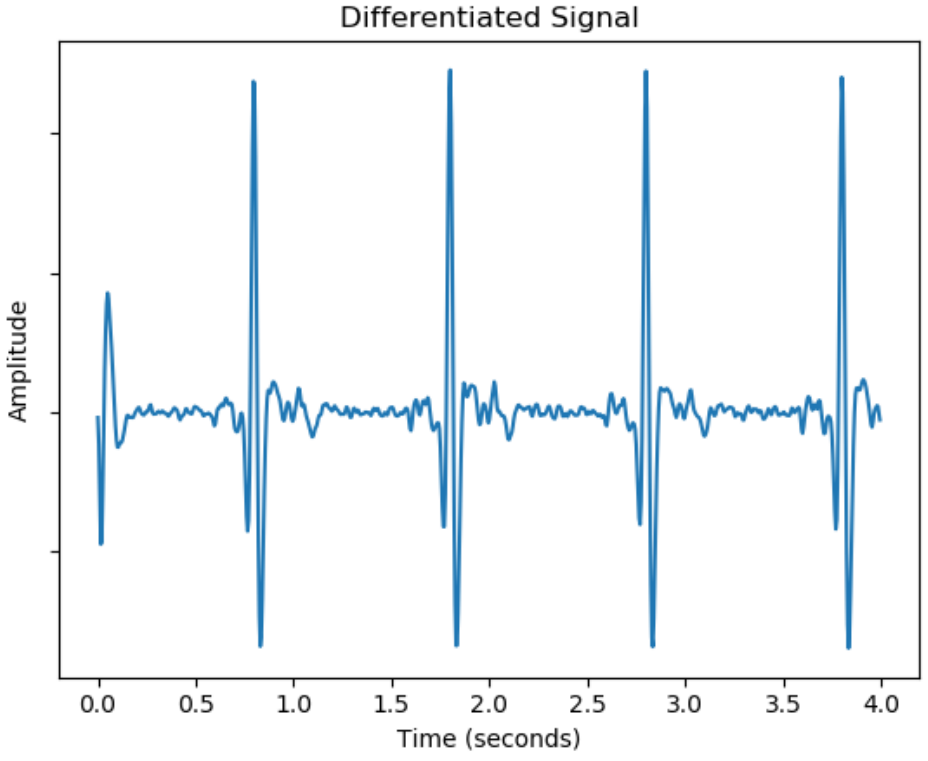

FIGURE 4.11: ECG signal after differentiation

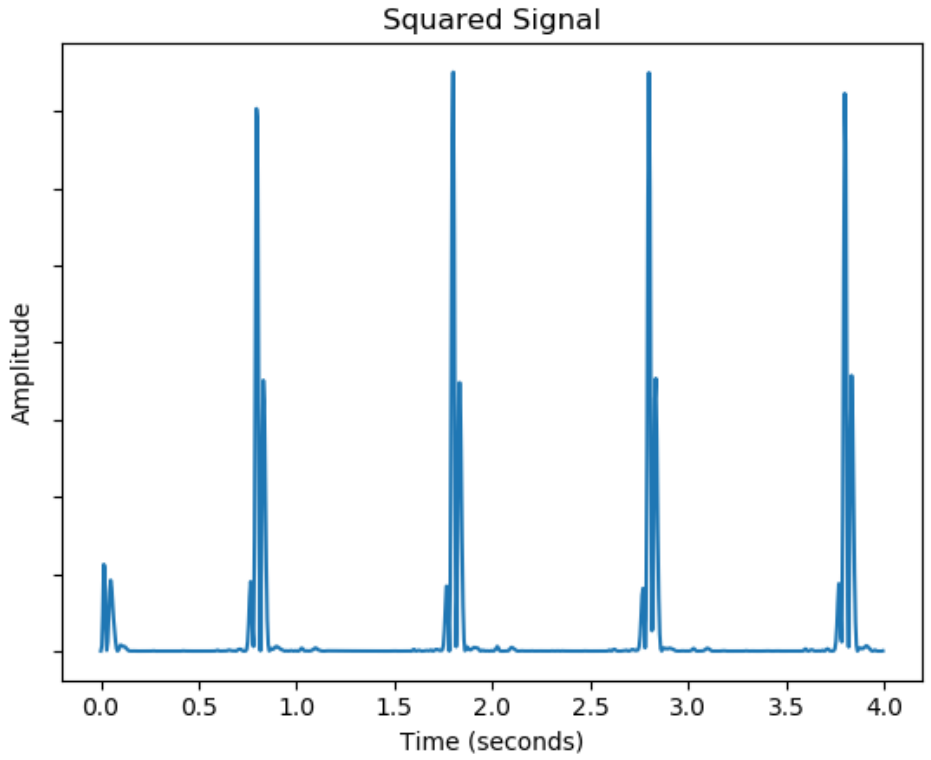

FIGURE 4.12: ECG signal after point by point squaring.

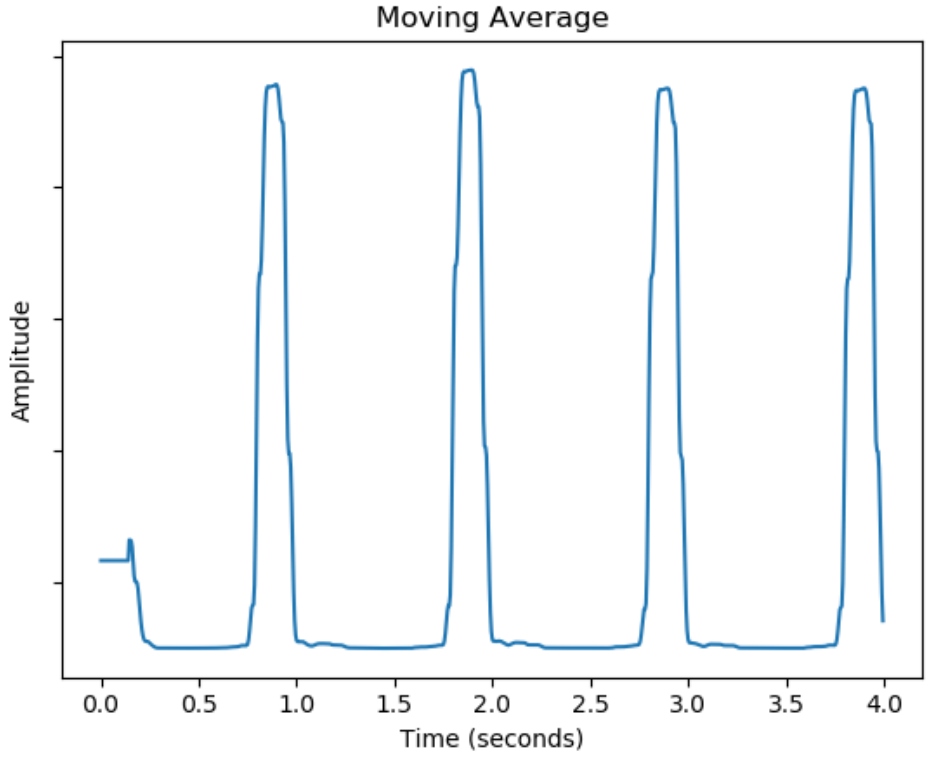

FIGURE 4.13: Moving average of ECG signal.

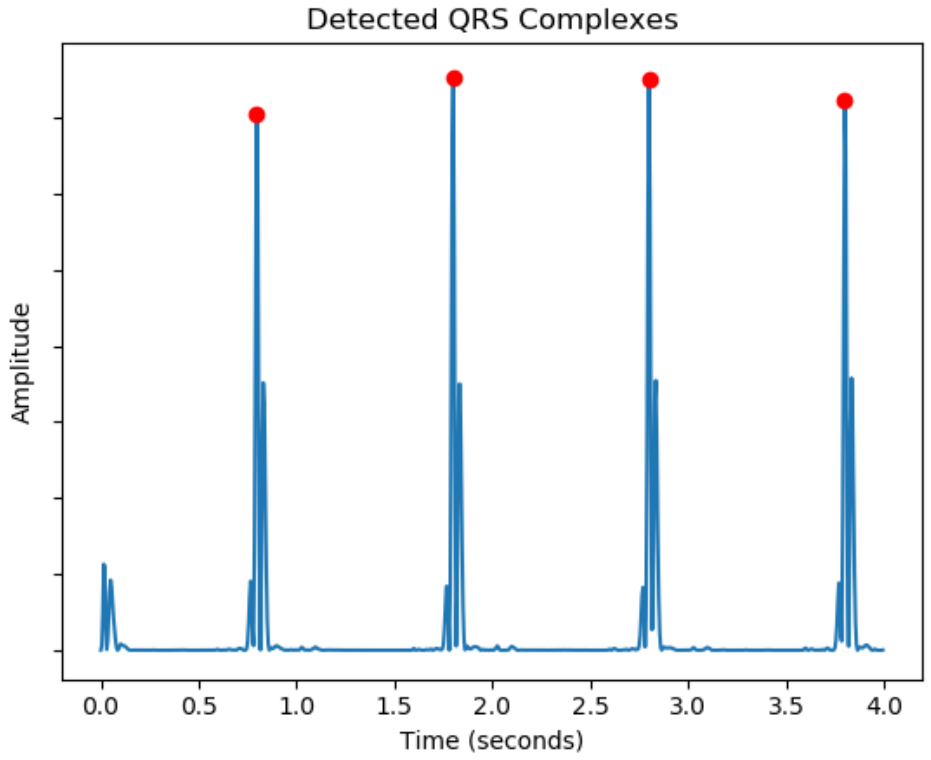

FIGURE 4.14: Detected QRS complexes of ECG signal

4.4 HRV Analysis

The interval between successive identified peaks, the *RR* intervals, are then calculated using the output of the PT algorithm. For the input ECG signal used in this section, which has a fixed heart rate of 60BPM, the expected result is that all measured intervals will be 1000ms.

IBI	Interval (ms)
RR1	1005
RR2	1000
RR3	1000
RR4	1005
RR5	1000
RR6	1000
RR7	1000
RR8	1005
RR9	1000
RR10	1000

TABLE 4.2: RR intervals for a fixed input rate of 60BPM

The results shown in Table 4.2 are a selection of inter-beat intervals calculated from the output of the PT algorithm. For a fixed input rate of 60BPM - which would mean the interval between successive beats is 1 second - the results show the measured intervals to all be approximately 1000 milliseconds as expected.

The experiment as described thus far indicates that the program performs well, correctly identifying and measuring the intervals between successive peaks in an ECG signal. However since HRV analysis is a study of the **variation** in the inter-beat intervals, this method of using an ECG signal generator would not be appropriate for gathering the results of HRV analysis, since there is no variation

in the fixed heart rate. The results of the HRV analysis - namely *RMSSD*, *SDNN* and *pNN50* - using this input data were negligible, as expected.

Therefore, an experiment was set up to gather real ECG data from the author of this report. The signal obtained from this experiment is not perfect, and this is likely due to the absence of electrode gel which is typically used when measuring biopotentials with electrodes placed on the skin. Nevertheless, although there is room for improvement here the results obtained are good.

The RR intervals obtained from the author's ECG data are then used in a number of HRV analysis calculations which give an insight into the cardiovascular health of the user.

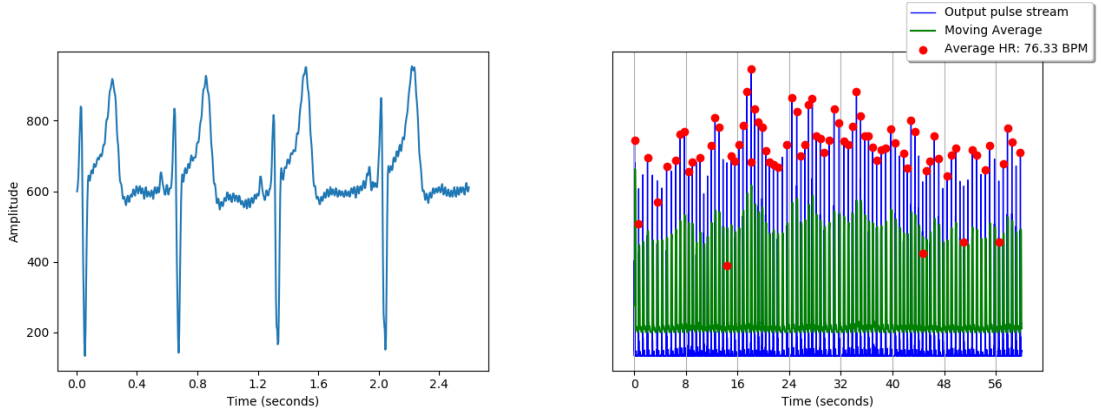

FIGURE 4.15: Author's ECG input signal (left), results of PT on input signal (right)

The HRV results obtained are shown in Table 4.3 show that, although the RMSSD and SDNN obtained are slightly above normal, the author thankfully appears to be in good health.

Metric	Result
Heart Rate (bpm)	76.33
RMSSD (ms)	378
SDNN (ms)	313
pNN50 (%)	23.68%

TABLE 4.3: HRV results obtained from author's ECG signal

4.5 File Storage & ThingSpeak

The results of the HRV analysis are examined to see if there are any indications of arrhythmias or cardiovascular diseases, by comparing the results to a range of normal values for each measured statistic. Using an input ECG signal with a fixed heart rate of 80BPM - which falls within the normal range of 60-100BPM - the patient data is removed automatically. An input at 40BPM or 120BPM - an indication of conditions called bradycardia and tachycardia respectively - means that the data is stored locally, with the filename annotated with the date and time the test was performed.

The results of the HRV analysis are uploaded automatically to ThingSpeak after processing has been completed. The ThingSpeak channel has four fields for each of the results in question - heart rate, RMSSD, SDNN and pNN50. The fields have parameters for adjusting the timescale, to average multiple results from a single period and thresholds to filter out unwanted results.

Chapter 5

Future Work

This project has primarily been a proof of concept and therefore there is considerable scope for future work. There are various aspects for future work that were not considered for the duration of this project either due to time constraints or because they were out of the project's scope. What follows are a collection of suggestions identified as areas of future work.

5.1 MIT-BIH Testing

This project's category falls under medical devices, and as such there a need for meticulous testing for the device. Medical devices, due to the need for accuracy and reliability, require a longer and more rigorous quality assurance phase compared to non-medical devices.

The MIT-BIH Arrhythmia Database is an ideal starting point for future testing to evaluate the accuracy of the implementation of Pan Tompkins algorithm. The database contains dozens of half-hour records which contain both normal, healthy ECG recordings as well as atypical recordings containing arrhythmias. The records are annotated where a physician has decided a *QRS* complex exists, so by comparing the results of the implemented algorithm we can determine its accuracy in detecting beats in the ECG signal.

5.2 Regulations

The purpose of this project was to implement a cost-effective design to analyse ECG patient data. While the main goals of the project have indeed been achieved, it is somewhat naive to believe the device could be used in the medical field in its current state. Due to their nature, medical devices have a large number of regulations surrounding them to ensure they are up to the industries standards. The Health Products Regulatory Authority - or HPRA - is responsible for the regulations regarding medical devices. One future aspect of the project could be to examine the regulations associated with Class II classification devices such as this, and to ensure the project meets these specifications.

5.3 Refine Hardware

The hardware configuration used in this project is a prototype, and as such is more bulky than necessary. There is considerable room for improvement in this area of the project since the current implementation uses a breadboard with male-to-male jumpers and a breakout board to connect to the RPI. A smaller form-factor and more permanent solution would be to use a circuit board, or an RPI prototype board - an example of which is shown in Figure 5.1

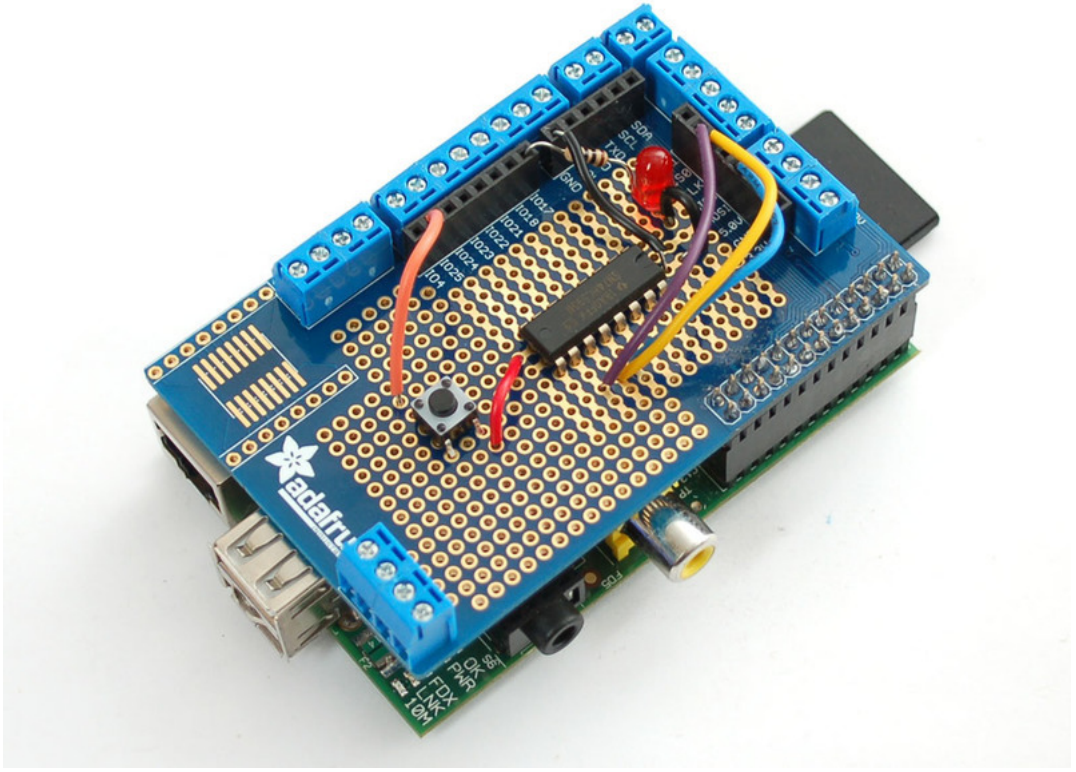

FIGURE 5.1: The Adafruit Prototyping Pi Plate - an example of an RPI prototype board

5.4 Software

5.4.1 Pan Tompkins Implementation Improvements

Although the results obtained with the implementation of Pan Tompkins algorithm are good, the implemented decision stage is quite simplistic compared to the decision stage outlined by Pan & Tompkins originally. The decision stage implemented, which detects *QRS* complexes by comparing the amplitude to the threshold, could be reworked to achieve better results. An potential future aspect could be to improve the algorithm by implementing two sets of thresholds on both the derivative signal and moving window integration signal, and using a search-back technique if the algorithm determines any peaks have been missed.

5.4.2 Cython

In the results section of this report, it was outlined that Python was dropped in favour of C to acquire data from the ADC at a stable rate. However it is generally considered bad programming to mix languages within a project. While writing this report, it became clear than an alternative solution would be to be to use Cython. Cython is a superset of Python which supports writing functions that the compiler converts into C code. A future task therefore could be to rewrite the code for this task using Cython.

5.4.3 Suggest possible conditions from HRV results

Heart rate variability results are calculated and sent to the cloud but are not used in any particularly meaningful way. These results could be used to give a preliminary diagnosis to the physician, which could be done in Python during processing or on the cloud using MATLAB, as discussed in the following section 'Cloud Analysis'. An abnormal heart rate can be an indication of conditions such as bradycardia and tachycardia and depressed HRV can be associated with liver cirrhosis and sepsis.

5.5 Cloud Analysis

ThingSpeak is used in this project for the storage of HRV analysis. This allows a physician to perform remote monitoring on the patient. However, the cloud is not being utilised to its full potential. ThingSpeak supports running automated analysis on data using MATLAB, so one aspect of future work could be to write an algorithm that determines if a patients condition has changed or deteriorated based on the HRV results. ThingSpeak could then be used to trigger a reaction - like an MSM message or email - to alert the physician.

Chapter 6

Conclusion

This thesis outlined a cost-effective method by which a patient's heart activity could be measured, recorded and analysed. It uses a 3-lead ECG sensor for electrocardiography, a 10-bit ADC for digitisation of the continuous ECG signal, and a Raspberry Pi to store and analyse the data using an implementation of Pan Tompkins algorithm in the Python programming language. Heart rate variability analysis is performed on the data and the analysis results are automatically uploaded and made available on the cloud for remote monitoring.

Over the course of this project, a number of potential applications for this project have been identified. The two main viable applications are as follows:

Low Income & Remote Areas: This project could be very beneficial to lower-income or remote areas. Remote areas located far from major hospitals which may not have access to ECG machine, or low-income countries like parts of Africa, Latin America and South-East Asia could benefit greatly from a low-cost approach such

as this. The project could provide an initial diagnosis that could uncover health issues so the patient could be referred to a cardiologist.

Remote Monitoring: By utilising the Internet of Things and the cloud, the possible applications for this project are greatly increased. A physician or cardiologist can use this feature advantageously by performing remote monitoring. The patient could use the device in his/her own home for an extended period of time and by continuously uploading the analysis results to the cloud, the physician can remotely monitor the patient's cardiovascular health.

Although there is room for improvement, the key methodology outlines a novel approach to the problem of heart monitoring to tackle the increasing issue of cardiovascular diseases.

Appendix A

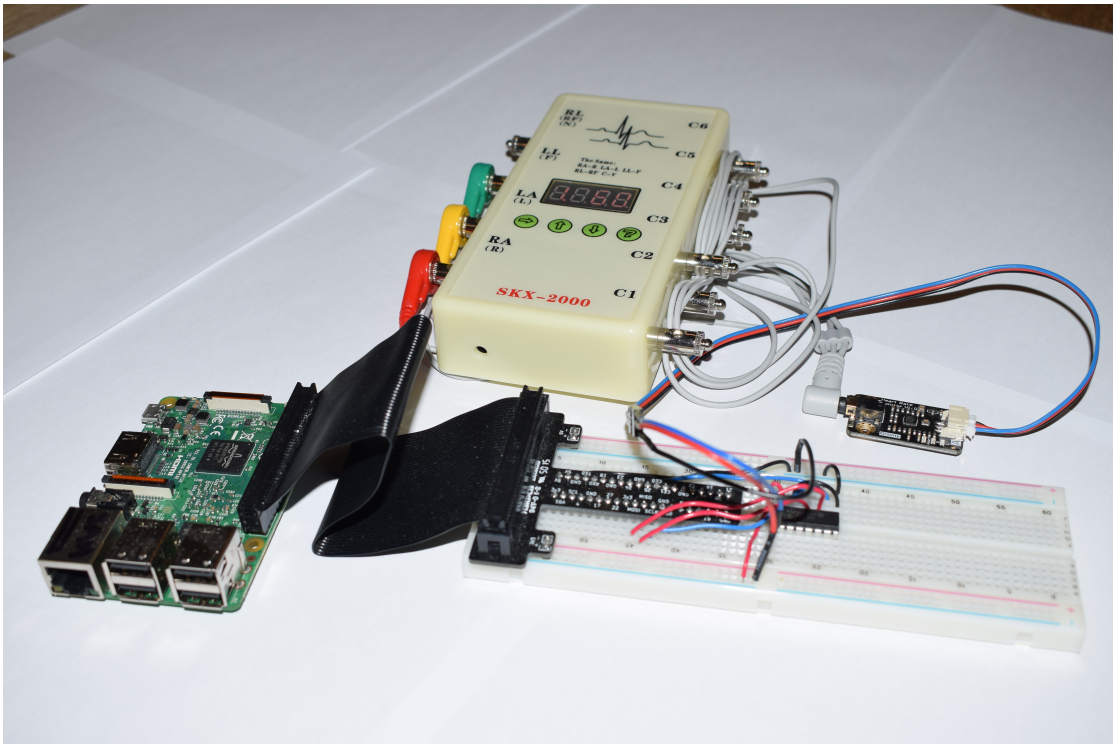

FIGURE A.1: Hardware setup

The hardware setup shown in the figure above consists of: Raspberry Pi 3 (left), ECG signal generator (top), ECG sensor (right of signal generator), MCP3008 ADC (connected to breadboard). Note that a ribbon cable and breakout-board are not necessary, but are used for convenience.

Bibliography

[1] H. Wang *et al.*, "Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: a systematic analysis for the global burden of disease study 2015," *The Lancet*, vol. 388, no. 10053, pp. 1459–1544, 2016.

[2] A. D. Bowry, J. Lewey, S. B. Dugani, and N. K. Choudhry, "The burden of cardiovascular disease in low-and middle-income countries: epidemiology and management," *Canadian Journal of Cardiology*, vol. 31, no. 9, pp. 1151–1159, 2015.

[3] E.-M. Fong and W.-Y. Chung, "Mobile cloud-computing-based healthcare service by noncontact ecg monitoring," *Sensors*, vol. 13, no. 12, pp. 16 451–16 473, 2013.

[4] J. Mohammed, C.-H. Lung, A. Ocneanu, A. Thakral, C. Jones, and A. Adler, "Internet of things: Remote patient monitoring using web services and cloud computing," in *Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE*

and Cyber, Physical and Social Computing (CPSCom), IEEE. IEEE, 2014, pp. 256–263.

[5] K. Ashton *et al.*, "That internet of things thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[6] S. S. Barold, "Willem einthoven and the birth of clinical electrocardiography a hundred years ago," *Cardiac Electrophysiology Review*, vol. 7, no. 1, pp. 99–104, Jan 2003.

[7] A. J. Camm, M. Malik, J. Bigger, G. Breithardt, S. Cerutti, R. J. Cohen, P. Coumel, E. L. Fallen, H. L. Kennedy, R. Kleiger *et al.*, "Heart rate variability. standards of measurement, physiological interpretation, and clinical use," *European heart journal*, vol. 17, no. 3, pp. 354–381, 1996.

[8] R. A. Álvarez, A. J. M. Penín, and X. A. V. Sobrino, "A comparison of three qrs detection algorithms over a public database," *Procedia Technology*, vol. 9, pp. 1159–1165, 2013.

[9] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *IEEE transactions on biomedical engineering*, no. 3, pp. 230–236, 1985.

[10] D. J. Ewing, J. M. Neilson, and P. Travis, "New method for assessing cardiac parasympathetic activity using 24 hour electrocardiograms." *Heart*, vol. 52, no. 4, pp. 396–402, 1984.

[11] J. Mietus, C. Peng, I. Henry, R. Goldsmith, and A. Goldberger, "The pnnx files: re-examining a widely used heart rate variability measure," *Heart*, vol. 88, no. 4, pp. 378–380, 2002.

[12] K. Umetani, D. H. Singer, R. McCraty, and M. Atkinson, "Twenty-four hour time domain heart rate variability and heart rate: Relations to age and gender over nine decades," *Journal of the American College of Cardiology*, vol. 31, no. 3, pp. 593 – 601, 1998.

[13] G. P Pizzuti, S. Cifaldi, and G. Nolfe, "Digital sampling rate and ecg analysis," vol. 7, pp. 247–50, 08 1985.

[14] S. Mahdiani, V. Jeyhani, M. Peltokangas, and A. Vehkaoja, "Is 50 hz high enough ecg sampling frequency for accurate hrv analysis?" in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*. IEEE, 2015, pp. 5948–5951.