THE UNIVERSITY OF MELBOURNE

Integrated Wireless Vital Signs Monitoring System for Specific Anaesthesia Parameters

Submitted in partial fulfilment of the requirements of the Degree of

Master of Engineering (Electrical)

by

Jia Hui Ong (571319)

Jiayi Chen

William Kit Mun Ngeow (596301)

Department of Electrical and Electronic Engineering

The University of Melbourne

Victoria 3010

AUSTRALIA

October 2016

Acknowledgements

We are very grateful to our supervisor Dr. Palaniswami for his enthusiastic and encouraging supervision of our research. Dr. Palani has been a great delight to work with, and a great support in our pursuits in Electrical Engineering.

I would also like to thank Dr. Yu Fan Zheng and Dr. Dragan Nesic for their helpful comments on improving the presentation of the thesis.

The Department of Electrical and Electronic Engineering, especially the Head of Department Assoc. Professor Doreen Thomas, has been very supportive of my part-time candidature.

Lastly I would like to thank my wife Annie for all the usual things, and my son Stephan, for patiently delaying his arrival until all the research for this thesis was completed.

Symbols and Acronyms

In this paper, we will use the following acronyms:

WVSMS = Wireless Vital Signs Monitoring System

ECG = Electrocardiography

EEG = Electroencephalography

PPG = Photoplethysmography

LED = Light Emitting Diode

OS = Operating System

VNC = Virtual Network Computing

CPU = Central Processing Unit

LAN = Local Area Network

HDMI = High-Definition Multimedia Interface

SD = Secure Digital (Non-volatile memory card format)

USB = Universal Serial Bus

GPIO = General-Purpose Input/Output

ARM = Advanced RISC Machines (Processor)

WiFi = Wireless LAN (WLAN)

IoT = Internet of Things

IC = Integrated Circuit

PCB = Printed Circuit Board

NTC = Negative Temperature Coefficient (Thermistor)

PSO = Phase Shift Oscillator

=

NEXT = THIS IS A SAMPLE

Abstract

This thesis considers the problem of the rejection of exogenous disturbance signals applied to a linear time invariant plant. An analysis of the disturbance rejection performance of periodically time varying feedback controllers is conducted. The performance used is the induced system norm of the closed loop disturbance response system.

The analysis considers both discrete systems, in which both plant and controller operate in discrete-time, and also sampled-data systems, in which the plant operates in continuous-time, while the controller operates in discrete-time. The disturbance input signals under consideration are assumed to belong to the l_p signal spaces (for discrete systems) and the L_p signal spaces (for sampled-data systems).

In the analysis, time invariant controllers are distinguished from strictly periodically time varying controllers. This allows a comparison to be made of the relative disturbance rejection performance of these two classes of controllers. For a given nonlinear time invariant periodic controller, a nonlinear time invariant controller is constructed which stabilizes the closed loop system when applied to the linear time invariant plant. Necessary and sufficient conditions are presented under which the nonlinear time invariant controller gives strictly better disturbance rejection performance than the nonlinear time invariant periodic controller.

Earlier results on l_2 (and L_2) performance of linear periodic controllers are extended to present a unified treatment of l_p performance for all $p \in [1, \infty]$ (respectively, L_p performance for all $p \in [1, \infty]$). Results are obtained for both linear and nonlinear controllers. Thus results in the literature on the inferior disturbance rejection performance of linear periodic controllers are shown to remain valid when the class of controllers is extended to include nonlinear controllers. Thus the principal claims to originality of this thesis are that it obtains results for a wider class of disturbance signals, and that it provides a performance comparison of nonlinear time

varying controllers with nonlinear time invariant controllers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 The Spaghetti Syndrome and its Implications - the need for a Wireless Vital Signs Monitoring System

Modern surgeries are challenging working environments involving a combination of complex devices, computers and humans working under stressful and time critical conditions. However, to date there are far too many wired devices communication between machines, monitoring patient physiological conditions and still requiring considerable attention by experts for monitoring and decision making.

In an era where surgery is ubiquitous, an ever increasing number of devices are attached to critically ill patients, multiplying the number of wires and tubes connected to the patient which results in a net resembling a plate of spaghetti; hence, the coinage of the term the Spaghetti Syndrome [38]. This conundrum has been present in the context of critical care, anaesthesia, and operating theatres for almost three decades with the advent of new technological advances in medicine [15]. This issue clearly illustrates the need for a system which is not physically connected to avoid possible harm and injury to both the surgeons and patients in the operating theatre.

The presence of wires in the surgery room introduces the possibility of increased surgery duration due to tripping hazards. More severe trips could lead to further surgical complications, due to wire disconnections which results in inaccurate readings of vital signs. Consequently, a

possible overdose or underdose of anaesthetic agents could follow, leading to patient injury, and in the worst case scenario, the demise of the patient. Indirectly, wires in the operating theatre increase the rate of mortality during surgeries.

One possible course of action to rectify this problem is to remove cables connected to patients by replacing the medium of transmission from a wire to electromagnetic waves. In recent years, multiple research projects have been carried out with the aim to embed vital sign sensors with wireless technology to separate patients from cables [56].

Nevertheless, most advances in this area of wireless device development is limited to the context of medical centres and not operating theatres.

1.2 Benefits of a Wireless Implementation

Having a wireless version of current vital signs monitoring systems will bring additional benefits to

Ease of use - convenience Less risk of injury Faster surgery - smaller delay due to obstruction from wires Safer Better working environment for doctors

1.3 Overview of the Project

This paper will consider the process and challenges of developing a fully integrated wireless vital signs monitoring system for vital signs specifically for anaesthetic parameters such as electrocardiography (ECG), photoplethysmography (PPG), electroencephalography (EEG), blood pressure, and blood temperature. The front end of the project encompasses the collection and processing of raw data from different sensors including the visualisation and presentation of vital signs information, according to some specified performance criterion. The back end of the project involves the wireless transmission and reception of the processed information to a fixed output display and possibly, a secure database.

The aim of this project is to make as many patient monitoring, measuring devices as fully wireless so enabling more freedom of movement for both patients, nurses, and medical staff. What makes this first task challenging is that the wireless network (WA) system must be safe,

secure and as reliable as current wired systems. Once completed, we envisage a new program in the development of pervasive wireless network resources for anesthetists and surgical practice in general. In these cases being hands-free, being able to access information, direct activities by simple movement, gestures adds to more efficient and clean surgical practice.

The project involves two parts:

1. programming hardware nodes to collect wireless sensor data using Android, and
2. software application development to visualize the data in real-time. The project also involves estimating missing data and inferring suitable information to medical professionals. The project requires developing a mobile app and also a data analysis platform.

1.4 Anaesthetic Parameters for Patient Monitoring

Countless operations are conducted on a daily basis which requires patients to be under general anaesthesia. To ensure the patient's optimal safety when under anaesthesia, it is necessary to consistently monitor certain parameters to ensure that they remain within a specified range which is considered to be safe or normal. Based on Atlee's Complications in Anesthesia [8], monitoring of anaesthetic admisnistration reduces the probability of anaesthetic overdose or underdose.

Complications associated with an improper dosage of anaesthesia could arise such as coma, brain damage, nerve damage, or possibly death, if real-time observations of vital signs are not conducted. The risk of such issues occuring during surgery could be reduced significantly if a feedback system were implemented for the amount of anaesthetics administered to the patient. Such parameters have been stipulated by ANZCA.

1.4.1 Theory of Electrocardiography

The electrocardiogram (ECG) refers to the recording of the "differences in electrical potential generated by the heart" using electrodes which are placed on the surface of the skin [50]. Both the action potentials generated by individual cells and sequence of activation affect the signal

registered during electrocardiography [50]. Other factors which alter the final signal include "the position of the heart within the body, the nature of the intervening tissue, and the distance to the recording electrode" [50]. Despite the many factors which can possibly contribute a change to the electrocardiogram, it is still possible to deduce with high accuracy the state of the heart from the surface ECG due to "the careful correlation of electrocardiographic patterns with observed anatomic, pathologic, and physiologic data" [50].

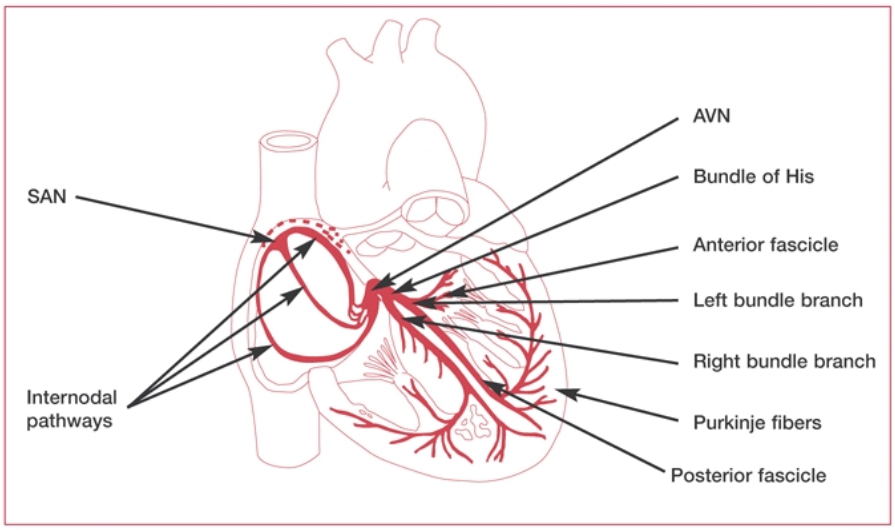

Figure 1.1: The Cardiac Depolarization Route. AVN: Atrioventricular Node; SAN: Sinoatrial Node. [36]

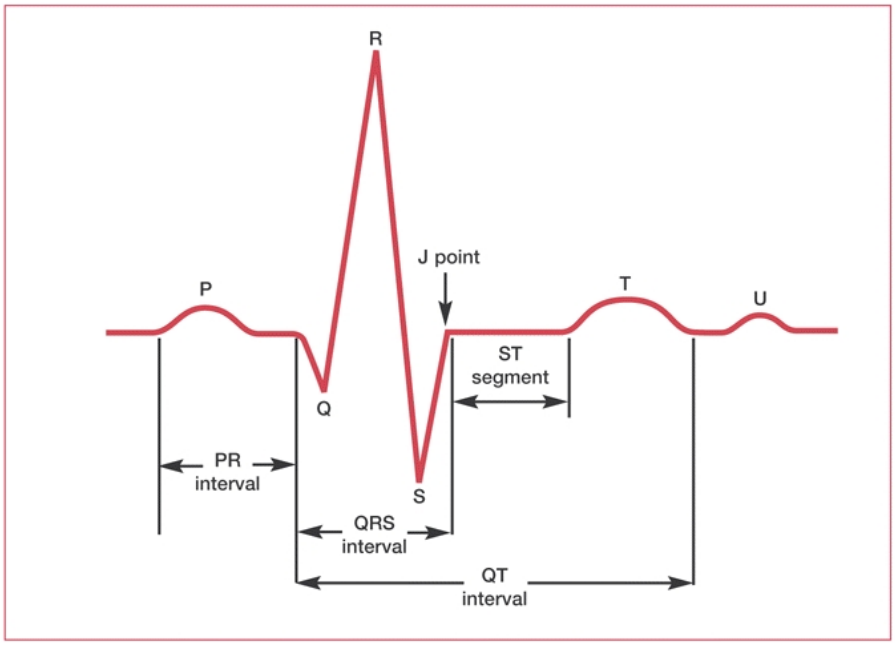

Figure 1.2: The Basic Pattern of Electrical Activity across the Heart [6]

Figure 1.2 shows a graphical representation of a typical electrocardiograph trace of the electrical signals from the heart.

The basic ECG pattern is correlated as follows:

- "Electrical activity towards a lead causes an upward deflection" [6]
- "Electrical activity away from a lead causes a downward deflection" [6]
- "Depolarization and repolarization deflections occur in opposite directions" [6]

Ashley and Niebauer (2004) (p.g. 19) [6] succinctly explains the types of waves and intervals of the ECG trace, (which mainly comprises of three different waves, namely P, QRS complex, and T):

> "The **P wave** is a small deflection wave that represents atrial depolarization.
>
> The **PR interval** is the time between the first deflection of the P wave and the first deflection of the QRS complex.
>
> The three waves of the **QRS complex** represent ventricular depolarization. For the inexperienced, one of the most confusing aspects of ECG reading is the labeling

of these waves. The rule is: if the wave immediately after the P wave is an upward deflection, it is an R wave; if it is a downward deflection, it is a Q wave:

- small **Q waves** correspond to depolarization of the interventricular septum. Q waves can also relate to breathing and are generally small and thin. They can also signal an old myocardial infarction (in which case they are big and wide)
- the **R wave** reflects depolarization of the main mass of the ventricles hence it is the largest wave
- the **S wave** signifies the final depolarization of the ventricles, at the base of the heart

The **ST segment**, which is also known as the ST interval, is the time between the end of the QRS complex and the start of the T wave. It reflects the period of zero potential between ventricular depolarization and repolarization.

T waves represent ventricular repolarization (atrial repolarization is obscured by the large QRS complex)."

1.4.2 Theory of Electroencephalography

An electroencephalogram (EEG) is a non-invasive test that detects electrical activity in your brain using small, flat metal discs (electrodes) attached to your scalp. The brain cells communicate using electrical impulses, and such activity will be showing up as wavy lines on an EEG recording. EEG measures voltage fluctuations resulting from ionic current within the neurons of the brain. It is most often used to diagnose epilepsy, which will cause the patient to have an abnormal EEG reading but also used to detect coma, sleep disorders, brain death etc [47].

Generally, EEG is used to evaluate several types of brain disorders. In surgery context, it is commonly used to determine the overall electrical activity of the brain as well as monitoring blood flow. EEG is also used in quantifying and characterizing effects of anaesthetic agents on the central nervous system, providing valuable information to the anaesthesiologist during the surgery [46].

Through an EEG diagram, depth of anaesthesia as well as the effect of different sedative at different period of surgery will be able to be monitored, especially during medically induced

coma.

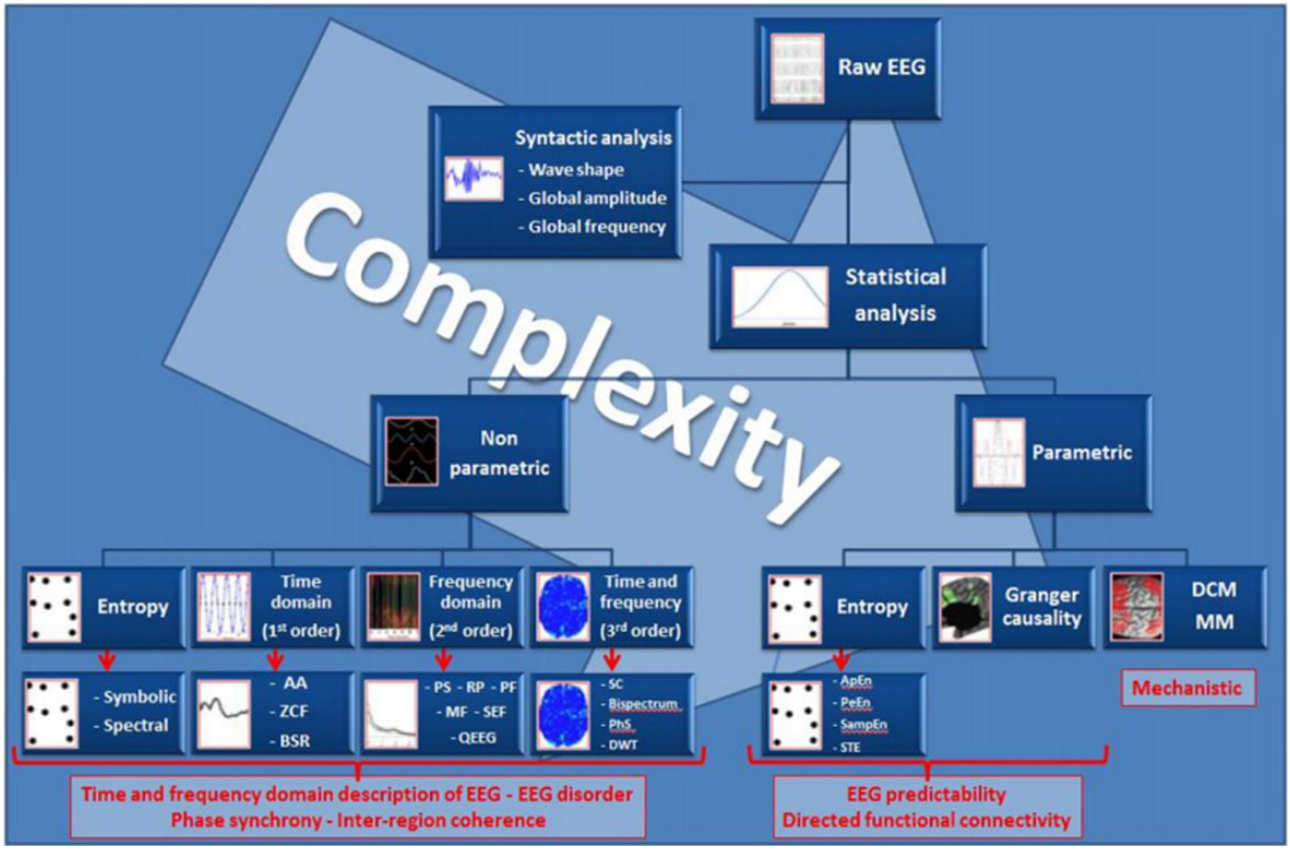

Figure 1.3: The Basic Pattern of Electrical Activity across the Heart [46]

A raw EEG signal will require digitization and computerization using modern technology to be able to generate specific EEG parameters which can be used to correlate to the effectiveness of anaesthetic agents in alteration of consciousness and inducing coma. Useful information can be extracted in both the time and frequency domain, where the frequency domain approach produces a better precise description.

During surgical procedure, anaesthetic agents are administered to ensure that patients can tolerate unpleasant and painful interventions to avoid complications. This is achieved through a combination of different effects of several agents, and the alteration of consciousness (hypnotic effect), immobility (through muscle relaxation) and limitations to reactions to surgical stimulation (anti-nociception). EEG recording is primarily used as a source to measure hypnosis and anti-nociception. In short, EEG provides a huge amount of information to the anaesthesiologist in understanding of mechanism of anaesthesia.

Engineering understanding: (I think I should summarize in a table what do you think)

A typical adult human EEG signal is about 10uV to 100uV in amplitude when measured from the scalp and is about 1-2mV when measured from the subdural electrodes.

EEG typically contains different rhythmic activity as well as transients and can be categorize into a few different bands as illustrated below

1. Delta wave at below 4Hz
2. Theta wave at 4 - 7Hz
3. Alpha wave at 8 15 Hz
4. Beta wave at 16 31 Hz
5. Gamma wave at 32++ Hz
6. Mu wave at 8 12 Hz

Each of the bands correlates to different functionality and regions of brains, and will be used pathologically to monitor different areas of interest. However, it is vital to know that EEG is affected by age, and EEG in childhood has lower frequency oscillations than adult EEG and have to be taken into consideration during analysis [41].

Under the use of anaesthetics, different general anaesthesia will result in different effects on the EEG observed. For example, the use of propofol will result in a rapid alpha and nonreactive EEG pattern seen over most of the scalp [32].

Due to the small amplitude of EEG signal, it is very prone to artefact and noise in some electrical activities arising from other sources other than the brain that are able to modify, distort or cancel the brain activity. This can be categorized into physiological artifacts (e.g. cardiac activity, muscle activity, eye movement) and non-physiological artifacts (e.g. electrode artifacts, external devices etc.) [49].

1.4.3 Theory of Photoplethysmography (PPG)

As a crucial part of patient monitoring system, PPG sensor, known as pulse oximeter is used to measure the SpO2 level. SpO2 stands for peripheral capillary oxygen saturation, an estimate

of arterial oxygen saturation, or SaO2, which refers to the amount of oxygenated haemoglobin in the blood" [37].

The blood oxygen saturation reading of a normal person should be between 95% and 100% [37]. A respiratory or cardiovascular problem may be present if the oxygen saturation drops to 90% 95%. The patient will highly likely experience hypoxic if the oxygen saturation falls under 90% [7].

Working Principle of the Infrared PPG Sensor

Photoplethysmogram Sensor (PPG sensor) takes advantage of the different absorption level of Oxyhemoglobin (HbO2) and deoxyhemoglobin (Hb) for the light beams with different wavelength. By placing a pair of infrared and red LED on one side of the finger and a receiver on the other side, the variation of light intensity can be measured. The light transmitting through fingertip will be absorbed by pulsatile arterial blood, non pulsatile arterial blood, venous blood and tissue [51]. As we know the pulsatile arterial blood is varying according to the heart pulse, hence the light absorption is also varying according to the heart pulse. Such variation is used as the waveform of PPG signal.

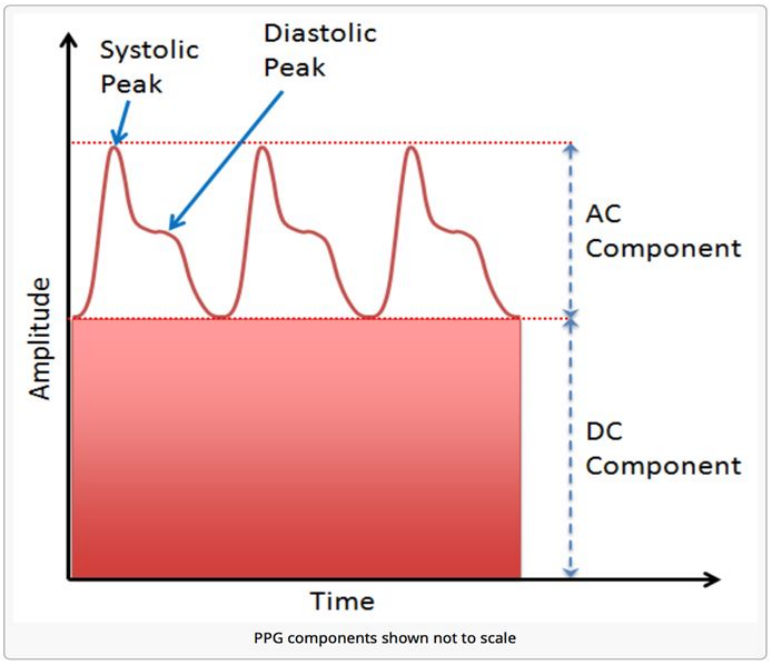

Figure 1.4: Waveform of PPG Signal [44]

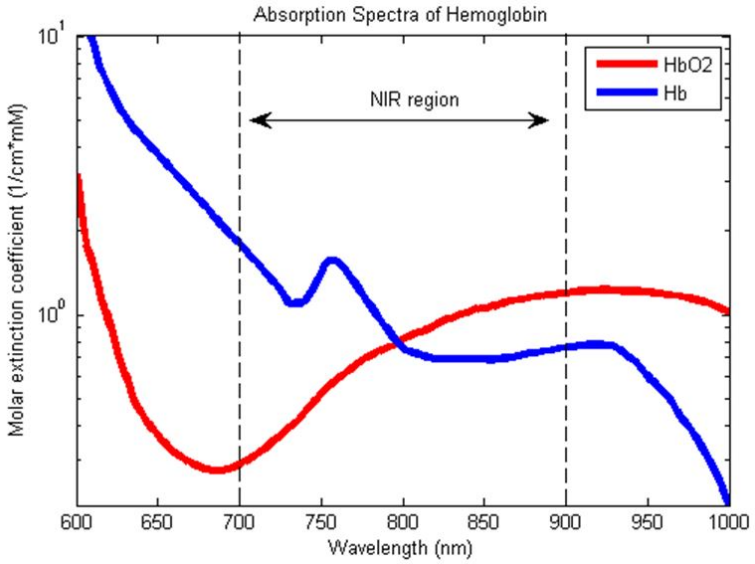

Figure 1.5: Absorption Spectra of Hemoglobin [17]

The Oxygen saturation level can be calculated by the following equation:

$$SpO_2 = \frac{HbO_2}{HbO_2 + Hb} \tag{1.1}$$

Where:

- HbO_2: Oxyhemoglobin
- Hb: Deoxyhemoglobin

The pulse oximeter uses BeerLambert law [10] [51] to define relationship between the light attenuation (light absorption) and the material it travels through [68].

A normalized ratio of the Red to IR then can be calculated:

$$R = \frac{\frac{AC_{RED}}{DC_{RED}}}{\frac{AC_{IR}}{DC_{IR}}} \tag{1.2}$$

Where:

- AC_{RED}: AC Component of Red light waveform

- DC_{RED}: DC Component of Red light waveform
- AC_{IR}: AC Component of Infrared light waveform
- DC_{IR}: DC Component of Infrared light waveform

The accurate SpO2 is proportional to this ratio. Proper calibration can be done to get the accurate SpO2 reading [57].

1.4.4 Theory of Blood Pressure

1.4.5 Theory of Blood Temperature

Chapter 2

Literature Review: Survey of Related Previous Work - Overview of Wireless Sensor Network Technology

2.1 Methodology

A search of literature was conducted using IEEE database to locate relevant literature relating to wireless applications in operating theatre. As the number of studies of this field is relatively new and limited, the search was broadened to include reviews of studies on various existing health monitoring ICT applications and not limited to only in the operating theatre. An online search of grey literature using Google was also conducted to diversify our findings.

The review of literature identified a number of areas where wireless technologies is highlighted in usage in the healthcare industry. Reviews have been conducted with focus on current available options, security and safety, existing wireless sensors as well as data prioritization protocols.

2.2 Introduction

Sensors networks and wireless transmission have become the trend in the electrical and electronics fields as Internet-of-Things is starting to be the norm. One of the major role of sensors now is in the biomedical field. With the help of wireless sensor network, patient monitoring and

care have becoming more advanced and easier. In the operating theatre, a range of wireless technologies have been utilized today, mainly using WiFi or Bluetooth in transmitting data from monitors to slave screens.

Wireless Electrocardiogram(ECG) monitors are also already available, where data is transferred wireless from the electrode to the monitors directly [22]. Similarly, there are also existing wireless sensors for blood pressure by MEMSCAP Wireless Solutions which transmits information from invasive blood pressure monitor to the display wirelessly [48].

Wireless transmissions in hospitals can be achieved through several different bandwidths, which can be seen in Table 2.1 below. Range, power, speed is among factors that are mainly considered when selecting an appropriate wireless technology to be used. However, it is noted that as both WLAN and Bluetooth are operating in the same frequency band at 2.4-2.48Ghz, throughput of data is decreased when both are used in coexistence. WLAN is used in hospital extensively and this may be a significant issue to design a system utilizing Bluetooth [67].

Formats	Power	Max Indoor Range	Max Speed	Frequency Band
802.11a	32mW	115 ft	54 Mpbs	5.15-5.825 GHz
802.11b	32mW	125 ft	11 Mbps	2.4-2.483 GHz
802.11g	32mW	125 ft	54 Mbps	2.4-2.483 GHz
802.11n	32mW	230 ft	300 Mbps	2.4-2.483 GHz
			and	5.15-5.824 GHz
Bluetooth (Class 2)	2.5mW	33 ft	3 Mbps	2.4-2.483 GHz
Ultra Wideband (UWB)	155mW	33 ft	480 Mbps	3.1-10.6 GHz
WHDI	17.8 mW	98 ft	3000 Mbps	4.9-5.9 GHz

Figure 2.1: citeeeeeeeeeeeeeeeeeeeeee [36]

2.3 Wireless in Anaesthesia

In the context of operating room, technical advances over the past decade have contributed to progress in anaesthesia safety. Improved patient monitoring, as well as increased usage of sensor system has introduced a new problem into practice, with great number of cables which makes care and nursing of patient difficult [67].

Conventionally, sensors are wired to ensure uninterrupted monitoring of physiological variables.

The use of wireless sensors will reduce the number of wires attached to the patients, easing transport and ambulation of patient in an intensive care environment [48].

2.4 Benefits

There are many benefits associated with the use of wireless systems to measure vital signs in the operating theatre. First, the clutter of cables can be significantly reduced. The risk of contamination will also decrease as cables are often exposed to contamination and not easily cleaned before repeat exposure to a sterile field. In the long run, there will also be significant cost reductions associated to operating room renovations. Monitors are frequently damaged when operating room staff are required to connect and disconnect cables on a regular basis. The wear and tear of cables will also require replacement from time to time. Using wireless solutions will eliminate the need for cables completely.

Figure 2.2: Benefits of Wireless Monitoring Systems [22]

For example, a wireless footswitch has been designed [22] and this eliminates the need for cords to be draped around or placed on floor under the sterile field. It can also connect to the receiver simply and easily used by the operating room staff. This improves efficiencies, eliminate cables and obstacles reducing operating room hazards and improves the turnover rate of the operation room.

2.5 Challenges

One of the main challenges associated with wireless monitoring systems would be security breach. It is a valid concern of someone snooping on data that is transmitted wirelessly. Adequate security and encryption have to be incorporated before it can be used in OR where a disruption of vital signs will be fatal. In the design of wireless antenna associated with the sensors, it is essential to ensure that consistent performance is achieved to avoid risk of affecting wireless range and quality of the wireless signal.

Furthermore, interference may be a prevalent issue in wireless systems, especially when there are multiple different forms of communications existing at the same time. For example, the use of 2.4Ghz band can be crowded by WiFi and other RF traffic, which may result in noise in the signals. Using wireless technologies will also increase significant costs to the development and maintenance of the medical devices.

2.6 Priority Queueing Models

As in our proposed model, there will be multiple sensors, namely EEG, ECG PPG etc. that are integrated into one transmission, a priority queueing model will need to be devised to ensure important vital signs data are prioritized when bandwidth / throughput is limited at certain stages of transmission. An appropriate queuing models are implemented to improve performance parameters for priority queueing, delay and server utilization time. In a study conducted by The NorthCap University [19] indicate that using M/G/1 Or M/G/N queueing models will have better performance depending on the expected number of incoming packets. An extract of the results is shown below, serving as a reference to our queueing model.

S. no	No. of servers	System utilization (%)	Expected No. of packets in queue (N_Q)	Expected No. of packets in system (N)	Average Waiting Time in Queue (W_Q)	Average Waiting Time in System (W)(includes service)
1	1	100	8.0	10.01	5.236	4.212
2	2	90	7.673	9.473	3.197	3.947
3	3	60	0.532	2.332	0.221	0.971
4	4	45	0.105	1.905	0.043	0.793

Figure 2.3: Result for M/G/1 and M/G/N queueing models for the proposed scenario [19]

2.7 Case Studies

There have been several successful studies that wirelessly monitors independent parameters but not all in the same package. For instance, in a project done in the University of Rhode Island [67], a smart wristband has been made to allow data transmission via bluetooth technology, which is able to send out BPM over standardized GATT Heart Rate Service that will calculate and output several metrics such as control point, heart rate etc.

Wireless system that remotely monitors patients oxygen saturation (%SPO2), pulse oximetry has also been devised [62] In this project, researches have been able to transfer vital signs monitored by pulse oximeter via IEEE 902.15.4 to a wireless sensor network for storage and display. A preliminary evaluation of the proposed system indicates that it is medically satisfactory in terms of accuracy of data, validating the use of WLAN as a primary form of wireless networks.

Case studies on Zigbee has also been done in India [65]. By using Zigbee to monitor body temperature and heart rate, the team has managed to transmit vital parameters from body to all wirelessly connected computer. Zigbee was chosen due to convenience and cost effectiveness. However, the prototype has managed to send signals at 1Hz frequency and its power consumption is minimal. The downside is that the signal transmitted is susceptible to noise. Proof of concept of utilizing Zigbee has been established, but with limited throughput, it will not be sufficient for the arrays of data that will be transmitted in our project.

With the emergence of 3D printed technology, it is possible to integrate it to design a scal-

able wireless monitoring system. A portable wireless systems measuring key metabolites in intensive care monitoring, using a 3d-printed case with sensing devices and integration of PCB with biosensors have been made in 2015, with successful implementation at its trials [11]. The biosensors will then transmit its measurements via Bluetooth module to mobile Android interface. This indicate the possibility of using Android system as a universal display instead of individual displays for each of the separate sensors.

Another solution that have been devised is to connect a dongle with wireless serial capability to convert serial data from some medical sensors to android devices such as smartphones for monitoring [55]. This can be expanded to slave monitor, or even smart glasses in future novel applications. The use of smart devices as a platform to converge information of different sensors can also be used to recognize past trends to predict future trends, presenting a functionality of smart alarm. However, this approach do not solve the fundamental issue of Spaghetti Syndrome, but serves as a good design reference for setup of slave monitors in the Operating Room.

2.8 Analysis of Existing Systems

There are several biomedical wireless systems that have been devised for research purposes, and will be listed for comparison and benchmarking for our project.

2.8.1 Consideration of e-Health Sensor by Cooking Hacks (Libelium)

The first would be the e-Health sensor shield, which is an interface with Arduino and Raspberry Pi to perform up to monitoring 10 different vital signs, including but not limited to SPO2, ECG, temperature, pulse etc [45].

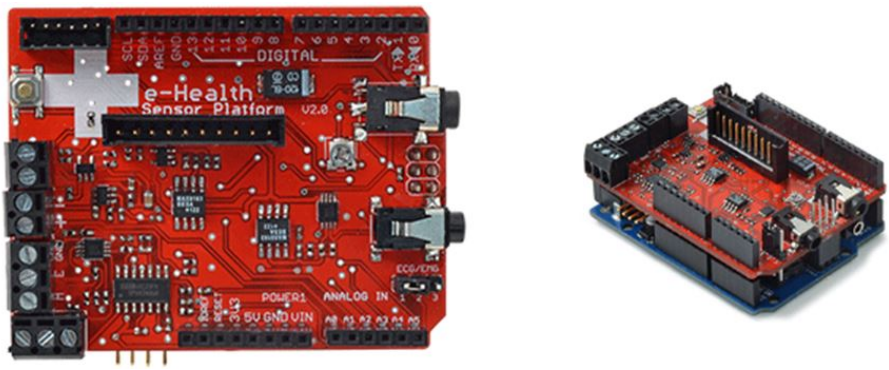

Figure 2.4: e-Health Sensor Shield [45]

This system is able to monitor the real time state of the patient to get sensitive data which can be subsequently used for analysis. Biometric information gather can be used to send wirelessly via multiple channel options: WiFi, 3G, GPRS, Bluetooth, 802.15.4 and Zigbee depending on application. However, this platform is designed to allow user to develop new products, i.e. an open source project to test different sensors and is not to be used in a professional medical context.

A key takeaway from this product is the use of AES 128 for Zigbee transmission and use of WPA 2 protocol for Wifi transmission. This is important as medical data will require to be completely secure especially in the Operating Room, as a mishap in monitoring may result in fatalities.

2.8.2 Consideration of BioNomadix Wireless Wearable Physiology by Biopac Systems

Another existing system would be Bionomadix wireless wearable physiology from Biopac Systems. This is a system they are currently researching on and has developed their own platform where each BioNomadix modules can interface to [13].

Each of the vital signs monitor(e.g. EEG, ECG) has separate transmitter and receiver, which makes the overall system to be bulky and harder to setup in a operating room. Contary to that, our design will allow integration of all sensors to one data transmission point, which reduces the total number of devices required for the same functionality.

Figure 2.5: Sample of BioNomadix Separate Vital Sign Monitor [13]

2.9 Wireless Sensor Networks

In line with the expansion of Internet of Things, this project is able to incorporate the idea of smart and ubiquitous connections between devices [34].

2.10 Body Area Networks

2.11 Methods of Wireless Transmission

2.11.1 Comparison between Virtual Nerwork Computing (VNC) and Raw Transmission

One of the biggest considerations of the project is whether to transmit the raw data wirelessly or to allow access to the data-processing terminal via virtual network computing (VNC). A quick comparison between the strengths and weaknesses is laid out below, with reference to Figure 3.1.

Raw data transmission from one terminal to another requires less processing power as the workload of data visualisation and transmission can be equally delegated between the two

terminals. In terms of the actual transmission, the bandwidth required by raw data transmission is much smaller than the bandwidth needed for setting up and maintaining a VNC server-client connection. This directly implies that there would be a smaller delay for the raw data transmission model.

On the other hand, VNC provides inherent encryption whereas raw data transmission is susceptible to interception if no encryption is implemented. In addition to this, most VNC server-client systems are packaged with user password authentication, providing increased security against unauthorised access to the sensor outputs collected from patients. Again, raw data transmission requires further deliberate work to attain similar security standards.

In wireless systems, disconnection due to the devices going out of range is commonplace and can cause disruption in the flow of information between terminals. For a VNC implementation of the WVSMS, data-processing isolation is a great benefit obtained compared to raw data transmission. Referring to Figure 3.1, should a disconnection occur, data processing and visualisation are not interrupted as the wireless transmission component is a completely separate, independent, and disjointed process between the terminals. As such, the Terminal connected to the Sensors are completely isolated. The Display-connected Terminal is not required for the system to function as the purpose of the second Terminal is only to be a VNC client for accessing the first Terminal. However, for raw data transmission be interrupted, both data processing and visualisation (assuming this is done by the Display-connected Terminal), will terminate completely due to the cessation of input. Using a VNC model effectively eliminates the problem of transmission errors affecting data processing.

Another point worth noting is the full remote control of the Sensor-connected Terminal that VNC affords to the user should there be a need to access the Terminal in real time to operate and change the processes. Raw transmission does not allow such control between Terminals.

From the above comparison, it can be seen that using VNC is highly suited for the purposes of this project instead of raw data transmission according to the required specifications as seen in Section 3.2.

2.11.2 Virtual Network Computing

https://en.wikipedia.org/wiki/Virtual_Network_Computing

VNC is platform-independent

VNC encryption isolation - big point security - authentication full remote control over patient's devices

Raw Less processing power required Less delay

2.12 WiFi

2.13 Bluetooth

2.14 Zigbee

2.15 Comparison of Different Protocols

Test1	Test2

Chapter 3

Project Design and Development

3.1 System Overview

The hardware of this vital signs monitoring system has been divided into three major sections as seen in Figure 3.1, namely:

- Sensors - Collects raw measurements of separate vital sign parameters
- Terminals - Processes data and handle transmission/reception of information
- Display - Visualises output of critical vital sign information

A block diagram of the Wireless Vital Signs Monitoring System is shown in Figure 3.1 down below. The system consists of three major portions: the Sensors which collect the raw measurements of separate vital sign parameters, the Terminals which process the data and handle transmission/reception of information, and the Display which visualises the output. The details of the development and options considered for each portion are further described in Sections 4, 5, and 6.

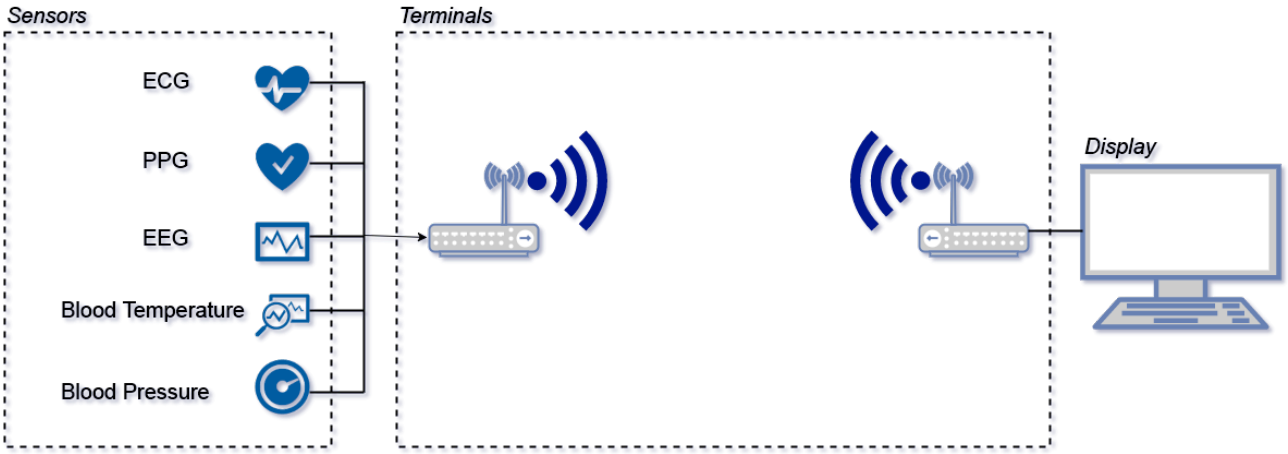

Figure 3.1: Wireless Vital Signs Monitoring System Block Diagram

3.2 Design Specifications

3.3 Hardware Design and Development

3.4 Software Design and Development

3.5 Analysis of Competitors - Similar Papers - Literature Review

3.6 Analysis of Existing Systems

3.6.1 Consideration of Bionomadix

3.6.2 Consideration of Libelium and Waspmote

3.6.3 Consideration of Cooking Hacks

Chapter 4

Terminals for Data Processing

The **Terminal** or **Platform** in this report refers to the machine or computer (hardware) which handles the processing of the raw measurements acquired from the sensors, the visualisation of the data received, in addition to the transmission and reception of the information from one geographical location to another through electromagnetic waves. Multiple types of terminals were considered and the factors which were accounted for were:

1. Processing Power
2. Operating System
3. Price
4. Power Consumption
5. Wireless Capabilities
6. Number of Input/Output Ports

4.1 Operating System

Of all the factors mentioned above, the OS for the Terminal is the most important and deserves a section of its own for further discussion. The OS must be fully capable of running the software required for data processing, data visualisation, virtual network computing (VNC)

server hosting, and wireless transmission. Consequently, the OS determines which Terminal should be chosen as some platforms are designed based on a specific OS.

Based on the above criteria, Linux was chosen for this project due to the open source nature of the OS. Many software programs and applications required for the functionality specified, such as the ones mentioned in Section 3.4, are readily available without cost from the Linux software database. Despite being an open source technology, security is not a concern but on the contrary, a significant advantage [18]. By releasing the source code of the OS to the general public, security experts from various backgrounds and fields are able to identify major security flaws in the system [18]. This allows for the swift resolution of such breaches by developers around the world, making security issues nullified as soon as they are found. Unlike proprietary OS such as Windows, the international community of developers are the ones who maintain the system for Linux, rather than a fixed number of employees working in specific locations around the world.

Nevertheless, one substantial problem with Linux OS is drivers for new hardware components [18]. However, due to the lower level nature of this project, drivers can be developed and so does not pose a major threat. Other drivers required for microcontrollers such as the Arduino Uno/Arduino Pro Mini for analogue to digital conversion are readily available and hence, does not pose a major threat.

One such platform which runs on a Linux kernel is the Rasberry Pi 3. The Linux distribution which was suggested by default for this miniaturised computer is the Raspbian Jessie, which is the Raspberry Pi derivative of the long standing Debian distribution. Section 4.2 has been dedicated to further analyse the Raspberry Pi 3 Model B used for this project.

Having a platform that runs on a Linux kernel provides the significant advantage of the ease of migration. As the project is a pilot study in wireless vital signs monitoring systems, there is a foreseeable need to continuously upgrade the platforms in terms of hardware capabilities to accommodate other important features. As long as the Linux kernel is used, it is not difficult to migrate to another more powerful Linux-based platform as the applications database and drivers available do not greatly differ for specific Linux distributions.

of all these, the OS is the most important - deserves a section of its own for discussion very

important - terminal is chosen based on this

Raspbian Jessie - Linux kernel - open source - more software and support available

- ease of migrating to other platforms - as long as the kernel remains the same

- windows considered -but proprietary - software will be expensive - issues if no support - limited support - need for payment

linux provides [20]

4.2 Raspberry Pi 3 Model B

Based on the criteria set out in Section 4.1, the Raspberry Pi 3 Model B loaded with a Raspbian Jessie OS is chosen as the platform or terminal for the project, henceforth referred to as the **Raspberry Pi**.

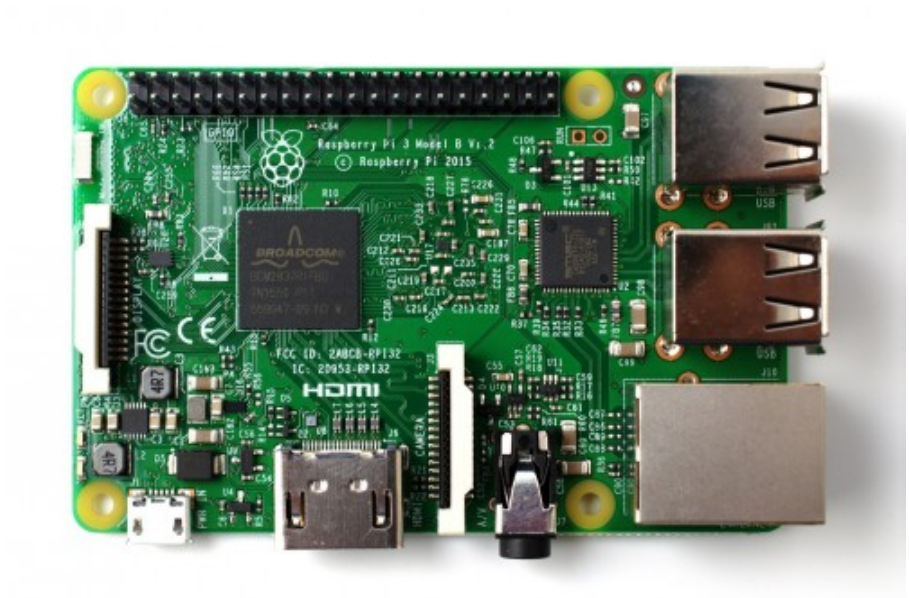

Figure 4.1: Raspberry Pi 3 Model B [29]

[27]

The specifications of the Raspberry Pi 3 Model B are as follows [27]:

- 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN

- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- VideoCore IV 3D graphics core

The Raspberry Pi's CPU is capable of running different types of Linux-based OS, in particular, ARM Linux distributions. Examples of such OS are Raspbian and Ubuntu Mate, whereas an instance of a non-Linux-based OS is the Windows 10 IoT Core. As discussed in Section 4.1, it is preferable to use the Raspbian Jessie as it is designed specifically for the Raspberry Pi.

In terms of connectivity, the Raspberry Pi leaves little to be desired with an integrated 802.11n Wireless LAN. This inbuilt WiFi adapter provides the needed wireless interface for the WVSMS and lays the foundation of the project back end design. An Ethernet port is also available as a contingency for reliable operation.

The 4 USB ports and 40 GPIO pins allow the sensors to be connected to the Terminal. However, it is important to note that all GPIO pins on the Raspberry Pi are only for digital input and output. Due to the absence of the analog-to-digital converters, there is a need to implement an interface for the Sensors, which provide signals in analog form. This is done using the Arduino Uno/Arduino Pro Mini as seen in Section 5.5.2.

The full HDMI port and VideoCore IV 3D graphics core provides the graphic processing power required to render the images when data is visualised and displays it to a monitor.

Based on the criteria laid out in Section 4, the Raspberry Pi provides the necessary processing power, operating system, power consumption, wireless capabilities, and input/output ports at a reasonable price.

The schematic design of the Raspberry Pi 3 Model B can be found in Appendix B.2.

[28]

4.2.1 Raspbian Jessie

Raspbian Jessie is [30] https://www.raspbian.org/

Installation instructions can be found in Appendix A.1

Version:May 2016 Release date:2016-05-27 Kernel version:4.4

4.3 Portable Power Supply for the Raspberry Pi 3 Model B

[25]

The Raspberry Pi is powered using a 5V micro-USB power supply [25]. The current drawn by the Raspberry Pi is variable and depends on the applications running in addition to the current drawn by peripheral devices which are connected. Typically, the Raspberry Pi 3 Model B current requirements range from 1.2A to 2.5A [25].

Buck - 9V

4.4 Intel Nuc

Chapter 5

Sensors

The **Sensors** in this report refer to the separate sensors connected to the body of the patient which collects raw measurements of vital sign paramaters, conditions the data so that it is suitable for analogue to digital conversion, and converts the data into a digital signal that can be interpreted by the platform (**Terminals**).

The sensors used in this project are enumerated as follows with the respective vital sign parameter measured:

1. ECG - AD8232 Heart Rate Monitor
2. PPG -
3. EEG -
4. Blood Pressure -
5. Blood Temperature - Thermistor in Phase Shift Oscillator Configuration

The following sections below further discusses the reason behind the choices made for the individual sensors, the mechanism of operation, and how it interfaces with the entire system.

5.1 ECG

The ECG front end design consists of a number of parts.

The typical range of an ECG signal lies between $0.01 \sim 300Hz$ and $0.05 \sim 3mV$ in terms of frequency and amplitude respectively [39]. The voltages sampled from the surface of the skin through electrodes are too miniscule to be directly plotted and displayed. In addition, there are multiple frequencies which are of interest in ECG signals but the raw measurements will contain unwanted signal artifacts from other sources. Hence, it is necessary to amplify and filter the signal before any analysis or visualisation is performed.

In 1997, Strohmenger [64] did an analysis of ventricular fibrillation ECG signal amplitude and frequency parameters for both successful and unsuccesful cardiac arrest countershocks. In this paper, the frequency range and amplitude range of both cases corroborates with the typical values stated above.

5.1.1 AD8232 IC ECG Sensor

In this project, in line with the aim of producing a system that is a working proof of concept, a simplified version of the ECG is used. The AD8232 IC from Analog Digital was used [1].

Front End

The Analog Devices AD8232 was used to implement the ECG part of this project. The AD8232 is "an integrated signal conditioning block for ECG", "designed to extract, amplify, and filter small biopotential signals" [1]. The functional block diagram is included below in Figure 5.1.

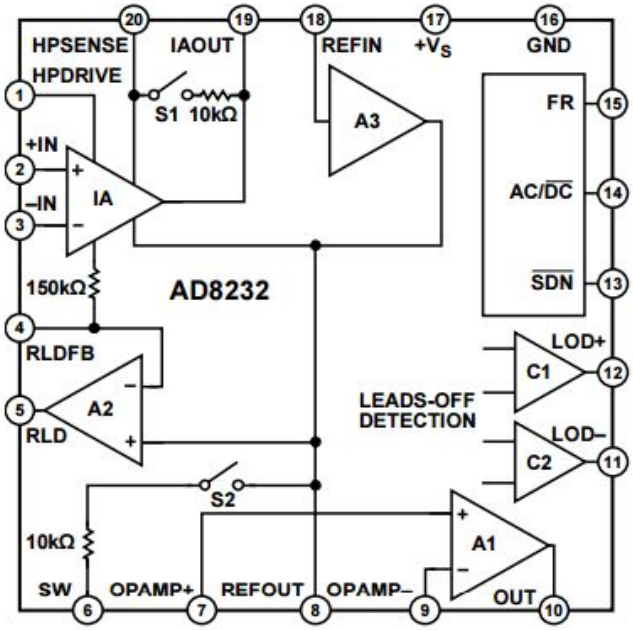

Figure 5.1: AD8232 Functional Block Diagram [1]

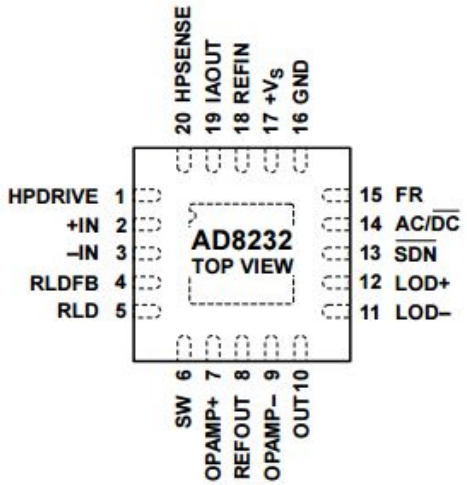

Figure 5.2: AD8232 Pin Configuration [1]

Theory of Operation of the AD8232

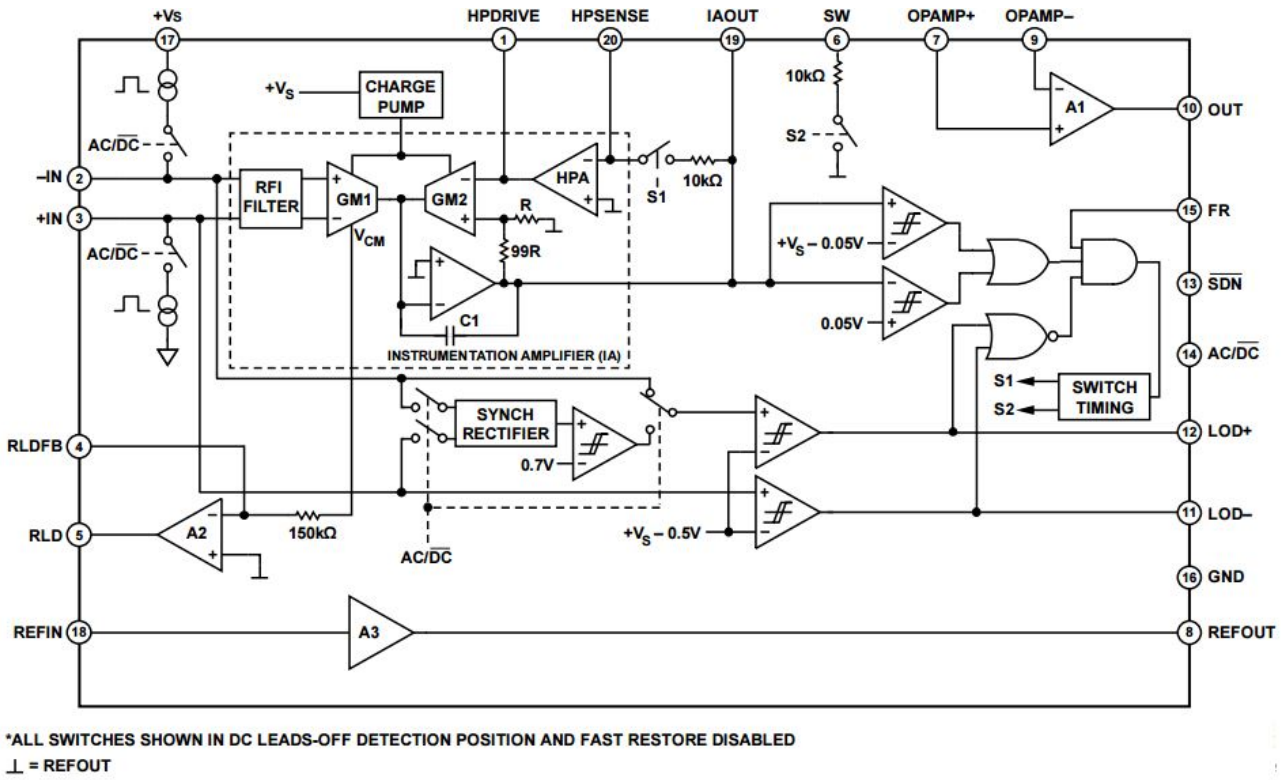

Figure 5.3: AD8232 Simplified Schematic Diagram [1]

Figure 5.3 above shows the architectural overview of the AD8232 chip from Analog Devices used for the "front end signal conditioning of cardiac biopotentials" [1] and is comprised of the following components:

- Specialized instrumentation amplifier (IA) - Amplifies ECG signals
- Operational amplifier (A1) - For low pass filtering and supplying extra gain
- Right leg drive amplifier (A2) - Improves common-mode signal rejection by inverting the signal
- Midsupply reference buffer (A3) - Provides a reference signal for the IA by creating a virtual ground between the supply voltage and system ground
- Leads off detection circuitry - Senses and indicates if the electrodes have been disconnected

- Automatic fast restore circuit - Decrease settling time of the ECG signal for a quicker response

[3]

[2]

[1]

AD8232 IC Cardiac Monitor Circuit Configuration

The datasheet for the AD8232 provided by Analog Devices supplies a typical circuit setup and configuration for a cardiac monitoring as seen in the schematic diagram in Figure 5.4 below [1]. A printed circuit board with this configuration could have been assembled and constructed using Altium Designer but due to time and cost constraints, an exact implementation of this circuit (with additional headers, an LED indicator, and a 3.5mm jack for biomedical pad connection) from SparkFun Electronics was used instead [14].

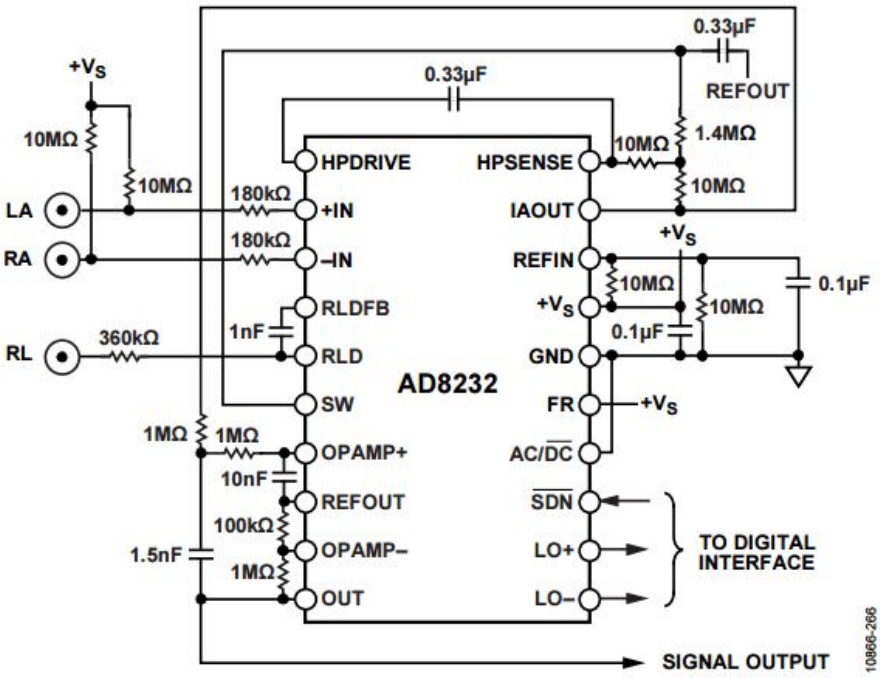

Figure 5.4: Circuit Configuration for ECG Waveform Monitoring using the AD8232 [1]

The schematic of the actual schematic capture of the AD8232 by SparkFun can be found in Appendix B.1. The AD8232 Heart Monitor in Figure 5.5 is essentially a breakout board for

the AD8232 integrated circuit provided by Analog Devices.

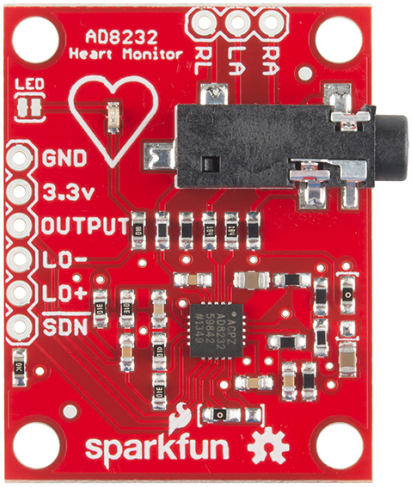

Figure 5.5: AD8232 Heart Rate Monitor from SparkFun Electronics [14]

CONSIDER PUTTING ALTIUM DESIGNER DESIGN

Complete AD8232 ECG Sensor Setup

Due to the absence of an analog-to-digital converter, it is necessary to use a microcontroller which is capable of this conversion and serial communication through a USB port for prototyping. As such, an Arduino Pro Mini 328 (3.3V/8MHz) was used in the development of the interface. The design of the system was sourced from SparkFun Electronics [14].

The assembly of the complete AD8232 ECG sensor requires the following components [14]:

1. Arduino Pro Mini 328 - 3.3V/8MHz (DEV-11114)
2. SparkFun USB Mini-B Cable - 6 foot (CAB-11301)
3. SparkFun FTDI Basic Breakout - 3.3V (DEV-09873)
4. Break Away Headers - Straight (PRT-00116)
5. Sensor Cable - Electrode Pads (3 connector) (CAB-12970)
6. Biomedical Sensor Pad (10 pack) (SEN-12969)

7. SparkFun Single Lead Heart Rate Monitor - AD8232 (SEN-12650)

These components are assembled in the pattern found in Figure 5.6.

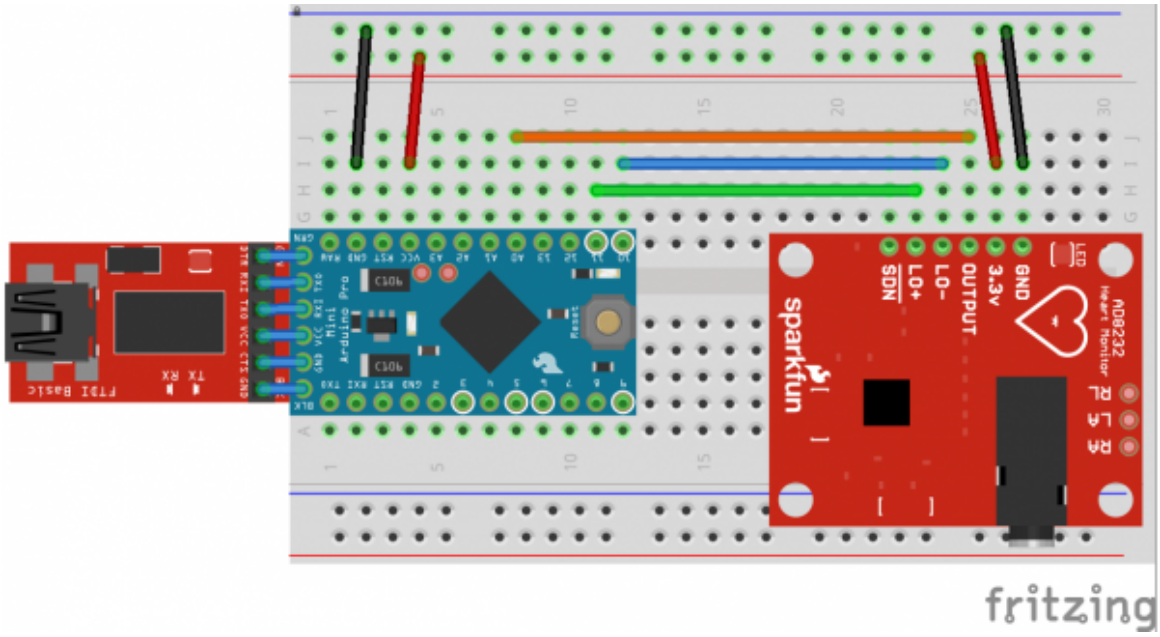

Figure 5.6: AD8232 Connection Diagram using Fritzing [14]

In terms of the pin connections between the AD8232 and the Arduino Pro Mini, the connections are found in Table 5.1.

Table 5.1: Pin Configuration for the AD8232 and the Arduino Pro Mini [14]

Board Label	**Pin Function**	**Arduino Connection**
GND	Ground	GND
3.3v	3.3v Power Supply	3.3v
OUTPUT	Output Signal	A0
LO-	Leads-off Detect -	11
LO+	Leads-off Detect +	10
SDN	Shutdown	Not used

The complete circuit can be found in the Figure below

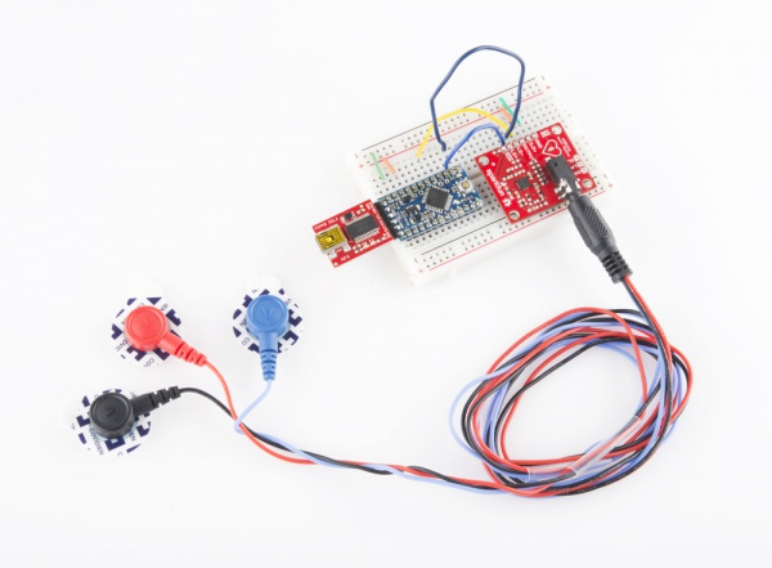

Figure 5.7: Sensors Connected to Heart Monitor [14]

To circumvent the need of using a breadboard, a PCB shield for the AD8232 and the Arduino Pro Mini was designed to accomodate both breakout boards on a single board without the need for wires to connect the internal components. The PCB design can be found in Appendix B.1.

5.2 EEG

5.2.1 Neurosky TGAM module

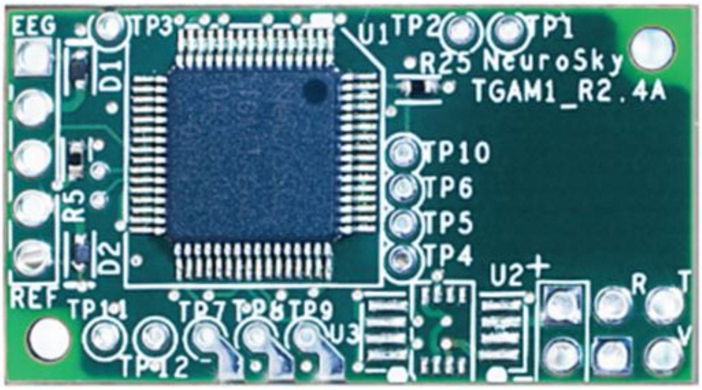

Figure 5.8: Neurosky TGAM Module

PLEASE CITEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE DATASHEET

Summary:

- The TGAM module is used as a primary brainwave sensor ASIC module.
- Able to process and output EEG frequency spectrum, EEG single quality, raw EEG
- Can be used together with simple dry electrodes
- Low power consumption ideal for portable battery driven application

Features:

- One EEG channel using three contact points, i.e. EEG electrode, EEG reference electrode and ground electrode
- Inbuilt filter system that resets when Poor signal quality detected for consecutive 4 seconds, or small noisy signals for 7 seconds

- Max power consumption at 15mA @ 3.3V
- Raw EEG data output at 512 bits per second
- Able to measure raw brainwave signal, different EEG spectrum such as Alpha, Beta, Gamma etc.

The low power consumption as well as different EEG spectrum that provides waveform we required is the basis that this chip is selected for our proof of concept

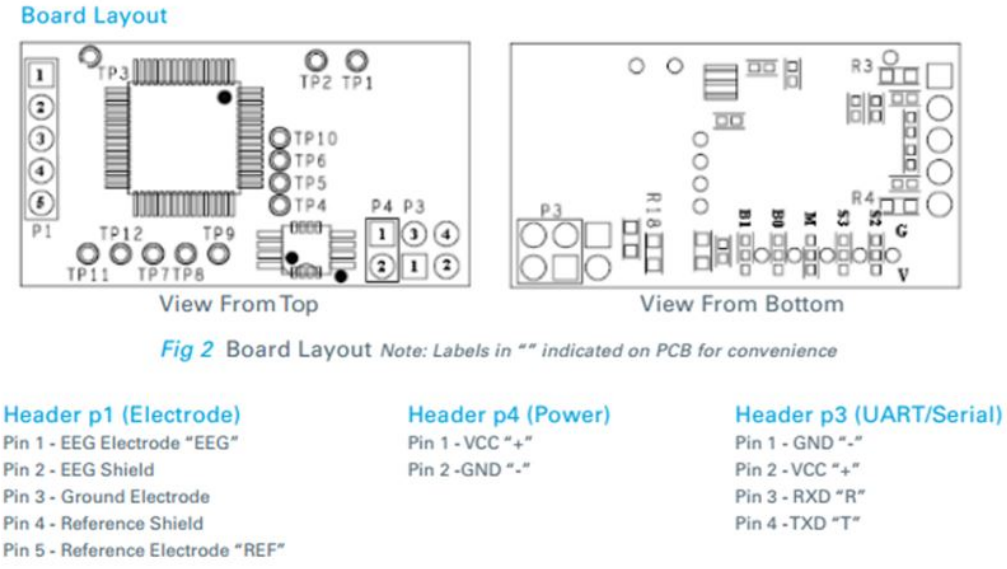

Figure 5.9: Board Layout

The chip is setup as above, with several important points below: - The EEG electrode is place on top of the skull / at forehead with application of electrolytes gel - EEG shield and Reference shield are open ended wires that is twisted around the corresponding electrodes - Reference electrode is placed at the base of the ear-lobe as it has least neurons and acts as a reference potential to the EEG electrode - The board is powered up by a separate voltage regulator providing 3.3V which will be discussed later

Preferable for this part to be in paragraphs. Overusage of bullet points not recommended by Robert Schmidt. :(

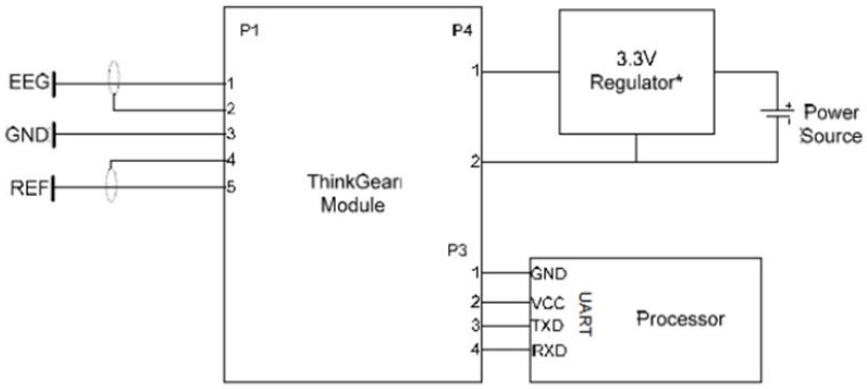

Figure 5.10: ThinkGean Module

A 9V battery is used as the power source, and Arduino is used as an interface to capture the output data to be visualized at the console.

5.3 PPG

When we tried to obtain the PPG waveform by directly connecting a pair of infrared transmitter and receiver with the Arduino, we were not able to visualize any valid signals. This could be due to two main reasons:

1. The photoplethysmogram signal is very small in amplitude compare with the DC offset component in the carrying signal.
2. The pair of sensors is not working properly.

To eliminate the second possibility, we did some testing with the sensors along. The intensity of the infrared beams received at the receiver is proportional to the separation between the transmitter and the receiver. By moving the transmitter towards and away from the receiver, we can observe instant change of the readings.

Hence, we come to the conclusion that the signal from our PPG sensor is very small in amplitude compare with the DC offset and the high frequency noise. Therefore, low-pass filter, high-pass filter and op-amp are used to extract the useful information from the raw signal from PPG sensor.

For the Arduino we are using, it has 10-bit analog to digital convertor which maps the analog input between 0 to 5V into 1024 discrete levels between 0 and 1023. This gives the resolution around 5mV per unit. Hence, in order to obtain a clear visualization of the PPG signal, the AC component needs to be amplified to few volts in amplitude (less than supply voltage 5V).

5.3.1 Bandwidth Selection

The frequency of the normal resting heart rate is between 1Hz to 2Hz [16]. Hence, for our Pulse Oximeter, we allow the signal between 0.5Hz to 4Hz to pass through. Extra margin is designed for special cases.

5.3.2 Circuit design

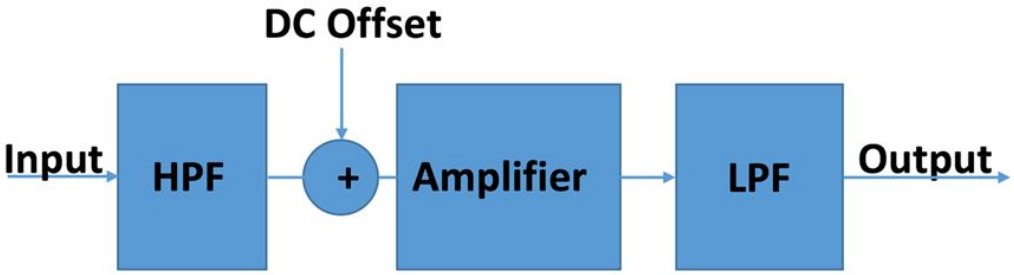

Figure 5.11: Circuit Design

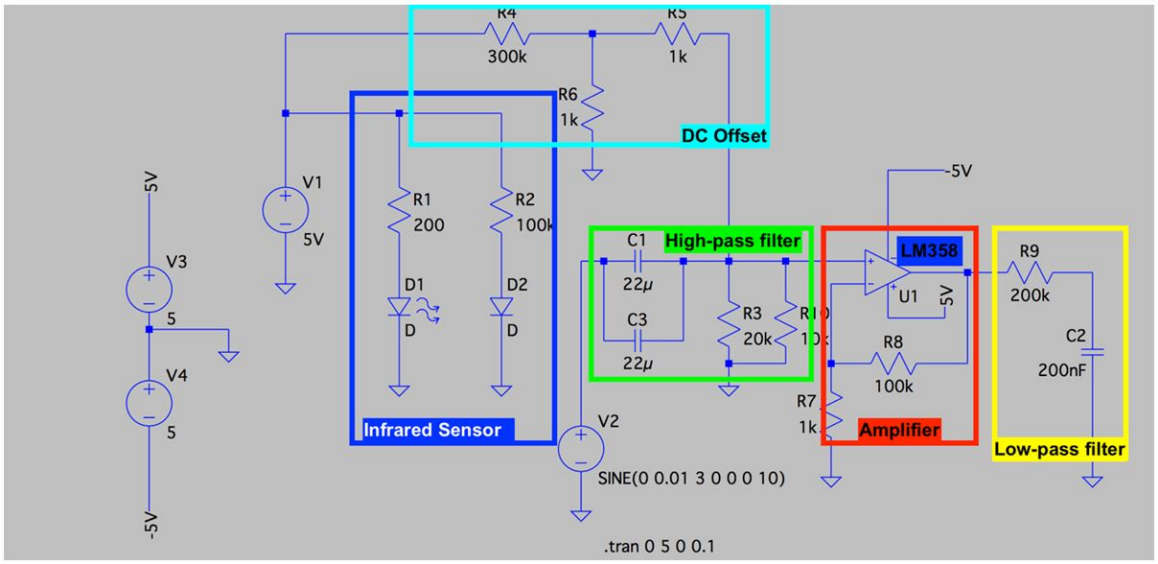

Figure 5.12: Circuit Design

High Pass Filter

A RC high pass filter is designed to meet the lower bound of the bandwidth requirement.

$$f_c = \frac{1}{2\pi RC} \tag{5.1}$$

In our design, we choose $R = 6666\Omega$ (20k and 10k in parallel) and $C = 44\mu F$ which gives us a cut-off frequency of $0.543Hz$.

Low Pass Filter

A RC low pass filter is designed to meet the upper bound of the bandwidth requirement.

$$f_c = \frac{1}{2\pi RC} \tag{5.2}$$

In our design, we choose $R = 200k\Omega$ and $C = 200nF$ which gives us a cut-off frequency of $3.98Hz$ which is close to $4Hz$.

Amplifier

Since we are not able to observe any waveform from the original signal, the amplitude of the original AC component must have magnitude of few mV because the resolution of the Arduino is 5mV. If the oscillation of the original signal has few hundred mV in amplitude, the waveform would be clearly observed without any signal processing. In order to amplify the signal to the magnitude of volts, we design an amplifier with gain of 100 which gives the output in Volts.

Another limitation is the power supply from the Arduino UNO. A normal operational amplifier requires positive and negative voltage for its supply rails. But Arduino UNO only provides positive 5V and GND. LM358 op-amp, which is used in our prototype, requires only positive voltage supply which is perfectly compatible with Arduino.

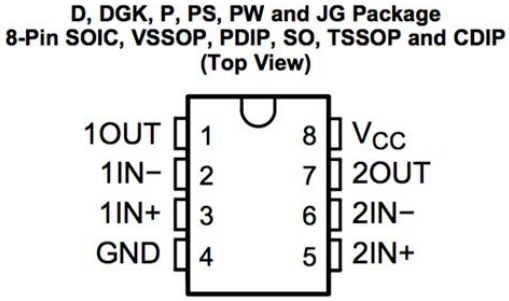

Figure 5.13: LM 358 [40]

Offset Voltage

To avoid signal being saturated at 0, an offset voltage is supplied to the input signal before the amplifier. Since the Arduino UNO can detect values between 0 5V, it is reasonable to have an offset voltage of 1.5V.

In order to achieve 1.5V offset at the output, the pre-amplified offset is set to be 1.5V/100 = 15mV, where 100 is the gain of the amplifier. Voltage divider is used to get 15 mV from a 5V source.

LTSpice Simulation

A sinusoidal input with frequency 3 Hz with peak-to-peak voltage 20 mV is used for simulation:

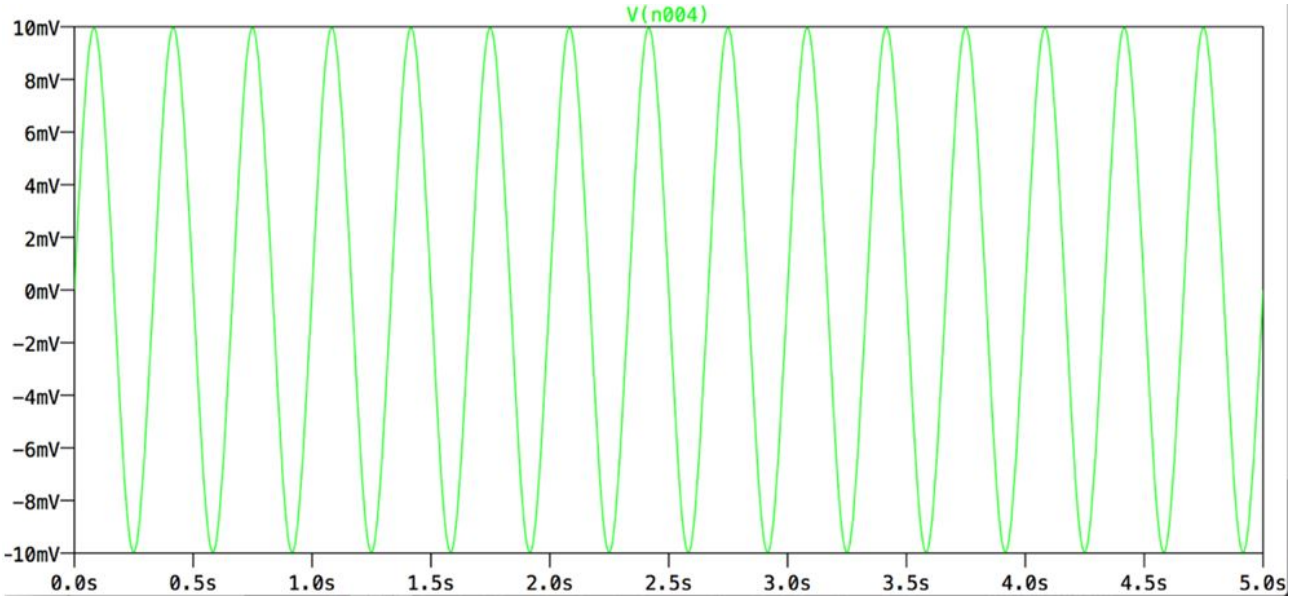

Figure 5.14: Sinusoidal Input

The output is:

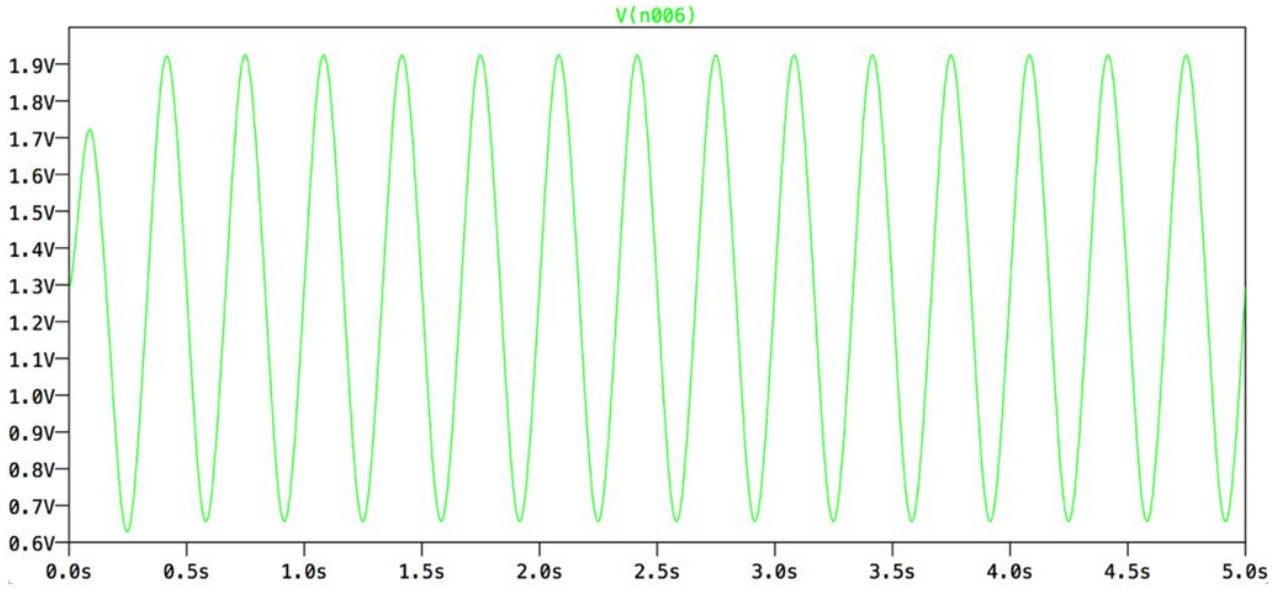

Figure 5.15: Output of LTSpice Simulation

From the output plot, we are able to see the input sinusoidal signal has been amplified to roughly 1.4V peak-to-peak with 1.3V DC-offset.

5.4 Blood Temperature

5.4.1 Thermistor

One of the methods to measure body temperature is through measuring heat conduction via direct physical contact with the patient's body. As a thermistor is a temperature-dependent resistor, the resistance of a thermistor varies with temperature. The property of thermistors therefore, are desirable in the design of a temperature sensor.

Using an Arduino Uno for prototyping, a simple voltage divider is constructed to study the characteristics of a thermistor. The thermistor used in this project is a material type F thermistor [54]. The circuit in Figure 5.16 was constructed for testing the thermistor.

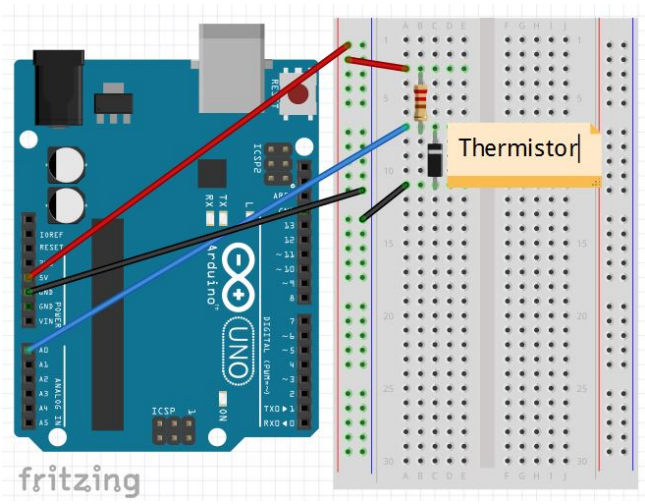

Figure 5.16: Thermistor in Voltage Divider Configuration using Fritzing

The datasheet of the material type F thermistor [54] reveals the following formulae and properties:

To calculate Rt/R25 at temperatures other than those listed in the table, use the following equation:
$Rt/R25 = \exp\{A + B/T + C/T^2 + D/T^3\}$
where T = temperature in K
where K = °C + 273.15

Temp Range (°C)	A	B	C	D
-50 to 0	-1.4122478E+01	4.4136033E+03	-2.9034189E+04	-9.3875035E+06
0 to 50	-1.4141963E+01	4.4307830E+03	-3.4078983E+04	-8.8941929E+06
50 to 100	-1.4202172E+01	4.4975256E+03	-5.8421357E+04	-5.9658796E+06
100 to 150	-1.6154078E+01	6.8483992E+03	-1.0004049E+06	1.1961431E+08

Figure 5.17: Thermistor Formulae and Properties [54]

To calculate the actual thermistor temperature as a function of the thermistor resistance, use the following equation:
$1/T = a + b(\ln Rt/R25) + c(\ln Rt/R25)^2 + d(\ln Rt/R25)^3$

Rt/R25 range	a	b	c	d
68.600 to 3.274	3.3538646E-03	2.5654090E-04	1.9243889E-06	1.0969244E-07
3.274 to 0.36036	3.3540154E-03	2.5627725E-04	2.0829210E-06	7.3003206E-08
0.36036 to 0.06831	3.3539264E-03	2.5609446E-04	1.9621987E-06	4.6045930E-08
0.06831 to 0.01872	3.3368620E-03	2.4057263E-04	-2.6687093E-06	-4.0719355E-07

†The deviation resulting from the tolerance on the material constant, Beta. The deviation must be added to the resistance tolerance of the part as specified at 25°C.

Figure 5.18: Thermistor Formulae and Properties (cont.) [54]

$$\frac{Rt}{R25} = exp(A + \frac{B}{T} + \frac{C}{T^2} + \frac{D}{T^3}) \tag{5.3}$$

$$\frac{1}{T} = a + b(ln\frac{Rt}{R25}) + c(ln\frac{Rt}{R25})^2 + d(ln\frac{Rt}{R25})^3 \tag{5.4}$$

T is the temperature in Kelvin where $K =^\circ C + 273.15$.

Choosing an operating temperature range of $0^\circ C$ to $50^\circ C$, Equation 5.3 and 5.4 has the following parameters:

Table 5.2: Material Type F Thermistor Parameters for $0°C$ to $50°C$

A = -1.4141963E+01	a = 3.3540154E-03
B = 4.4307830E+03	b = 2.5627725E-04
C = -3.40789983E+04	c = 2.0829210E-06
D = -8.8941929E+06	d = 7.3003206E-08

Therefore, we know that the range of $\frac{Rt}{R25}$ is between 0.36036 to 3.274. R_{25} was chosen to be $10k\Omega$. This circuit was tested with the Arduino code found in Appendix A.2.4.

It is also known that this material type F thermistor is a negative temperature coefficient (NTC) thermistor, that is, when the temperature rises, there is a decrease in resistance.

Steps for testing Negative Temperature Coefficient (NTC) behaviour:

1. Set up circuit as in Figure 5.16. Attach RT between A0 and GND. Upload AnalogRead code to Arduino.
2. Begin Serial Monitor.
3. Touch (for a few seconds) and release thermistor with bare fingers.
4. Once the value stabilises, insert data into Microsoft Excel for plotting.

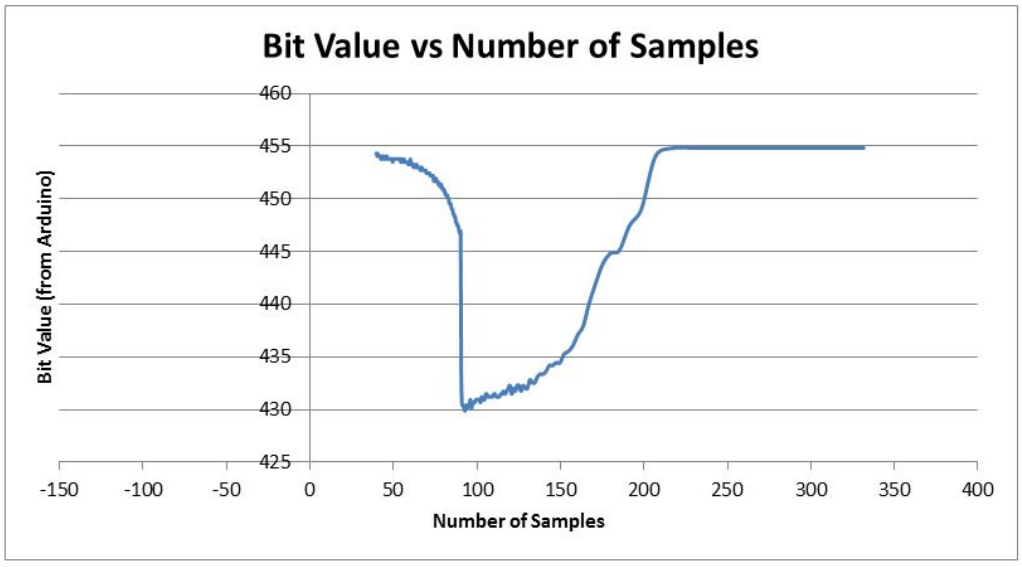

Figure 5.19: Bit Value vs Number of Samples

From Figure 5.19, when the thermistor is touched, the sensorValue decreases. This is corresponds to a decrease in RT when the temperature increases. The thermistor therefore is consistent with a NTC thermistor.

A simple voltage division describes the above behaviour:

$$\frac{RT}{R25} = \frac{sV}{1023 - sV} \tag{5.5}$$

The bit range of an Arduino Uno is 1024 bits, meaning that there are 1024 bits ($0 \sim 1023$) which describe the input into A0 as there are 2^{10} different voltage levels for the analog input of an Arduino Uno. sV represents the detected sensorValue.

Analysis of Nonlinearity between Thermistor Resistance and Bit Value of the Arduino

The following graph is obtained by plotting the relationship between temperature and bit value using Equation 5.3 and 5.5, by iterating through all possible values of the sensorValue within the bit range.

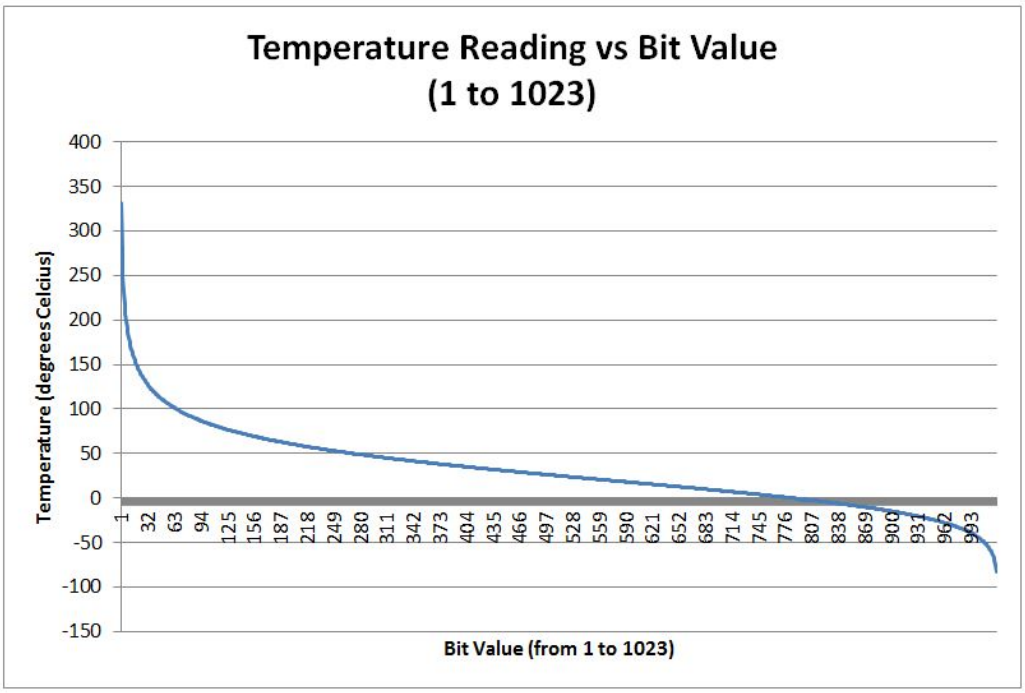

Figure 5.20: Temperature Reading vs Bit Value

The graph shown in Figure 5.20 indicates that the relationship between the thermistor resistance and the bit value registered is nonlinear. However, we are only interested in the values between the range of $5°C$ to $45°C$ (narrower for better accuracy). Plotting the same graph for this specified range yields Figure 5.21

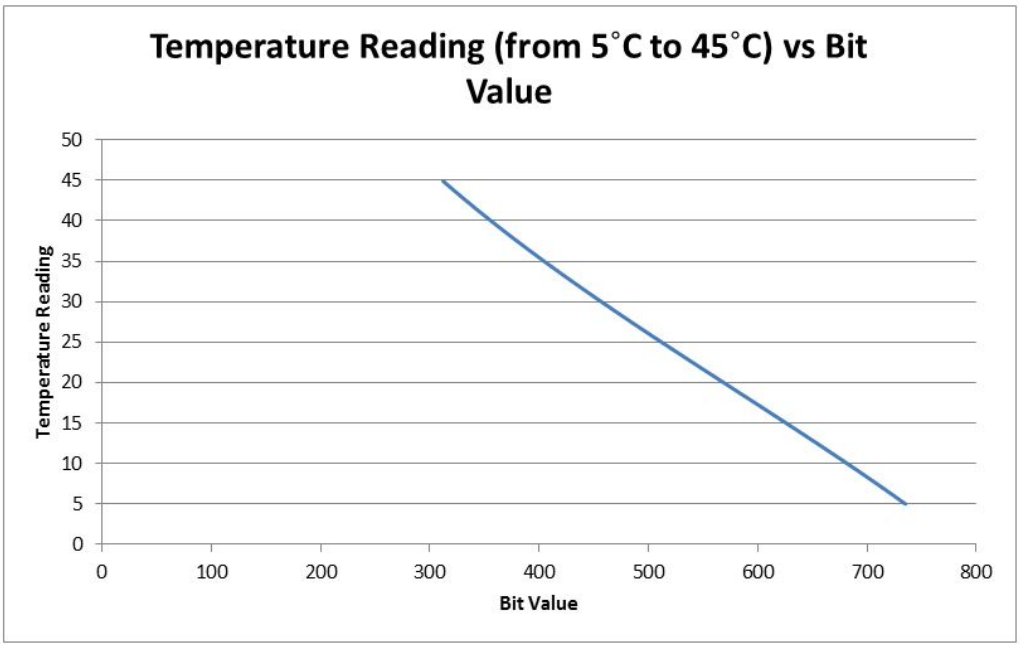

Figure 5.21: Temperature Reading ($5°C$ to $45°C$) vs Bit Value

By keeping within the range of $5°C$ to $45°C$, the relationship between the thermistor resistance (given by the temperature) and the bit value of the Arduino is approximately linear.

Within $5°C$ to $45°C$, the bit values that represent this temperature range are between 312 and 735. Roughly 424 bits are used to represent $5°C$ to $45°C$ in the analog-to-digital converter of the Arduino Uno.

Sensitivity Analysis of Material Type F Thermistor

The derivative of Equation 5.3 is:

$$\frac{d(RT/R25)}{dt} = -\left(\frac{B}{T^2} + \frac{2C}{T^3} + \frac{3D}{T^4}\right) exp(A + \frac{B}{T} + \frac{C}{T^2} + \frac{D}{T^3}) \tag{5.6}$$

By plotting Equation 5.6 (the time derivative of RT/R25) for the temperature range between 275K and 350K, we obtain Figure 5.22.

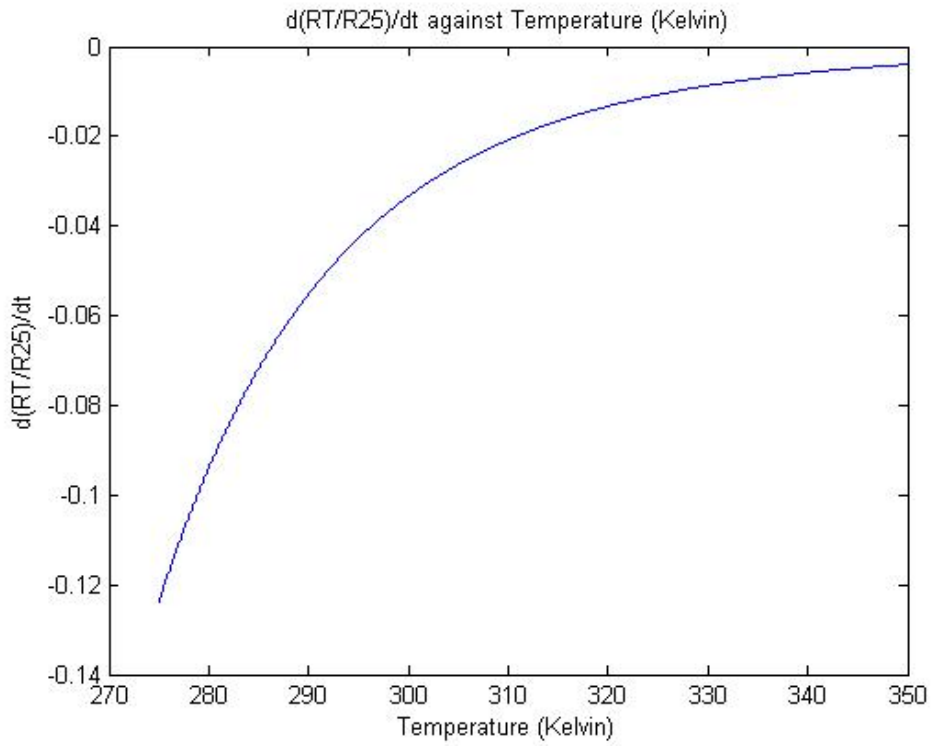

Figure 5.22: $\frac{d(RT/R25)}{dt}$ against Temperature (Kelvin)

Figure 5.22 shows that as temperature increases (270K to 350K), the rate of change of RT/R25 decreases that is, as ΔT increases, ΔR decreases. A change in temperature at lower temperatures gives a bigger change in resistance than a change in temperature at higher temperatures. Hence, the sensitivity of the thermistor decreases as temperature increases.

Self-Heating of Negative Temperature Coefficient (NTC) Thermistor

Due to the nature of negative temperature coefficient (NTC) thermistors, it can be deduced that the higher the temperature, the lower the resistance of the thermistor. By having a lower resistance with the same voltage across the component, the current through the component increases due to the relation in Equation 5.7. An increase in current always translates to more heat being dissipated across a resistor, thereby increasing its temperature. As such, there is a positive feedback mechanism for NTC type thermistors, leading to the problem of self-heating. This affects the accuracy of devices which use thermistors as voltage dividers. Hence, the thermistor configuration used above in Figure 5.16 is only suitable for testing the properties of the material type F thermistor but not for actual temperature measurement.

$$V = IR \tag{5.7}$$

5.4.2 Thermistor in Phase Shift Oscillator (PSO) Configuration

To circumvent the issue of self-heating of NTC thermistors in Section 5.4.1 above, it is necessary to reduce the amount of current flowing through the component.

From the voltage division perspective, temperature is resistance-dependent. However, it is possible to make temperature dependent on frequency by implementing thermistor coupled with a phase shift oscillator.

The phase shift oscillator (PSO) is an oscillator with a resistor-capacitor network, leading to regenerative feedback which produces a sine wave output signal [66]. This regenerative feedback is due to the charge storage capacity of the capacitor [66].

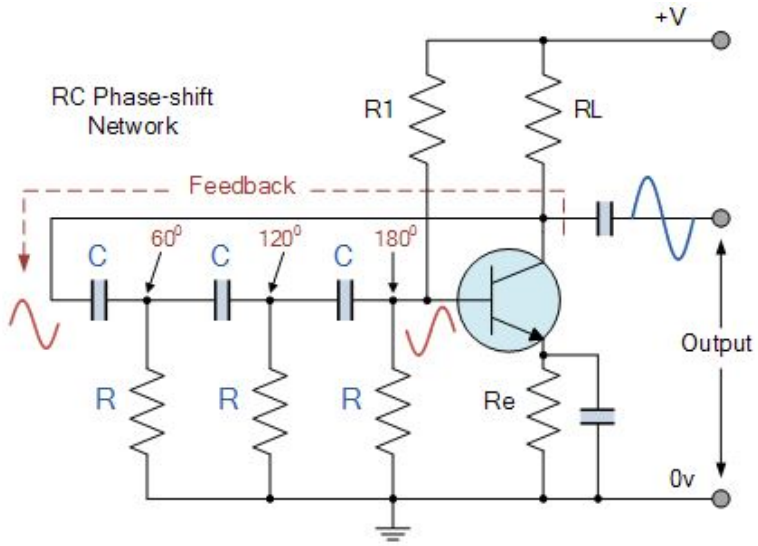

Figure 5.23: Basic RC Oscillator Circuit (Phase Shift Oscillator) [66]

One of the properties of the phase shift oscillator is the change in frequency when any of the values of the resistances in the phase shift network is adjusted. This leads to the dependence of the output frequency on the resistance.

By replacing one of the resistors with a thermistor, it is then possible to keep the current relatively small across the thermistor while still measuring the change in temperature based on

the change in frequency. Hence, using a thermistor coupled with phase shift oscillator helps to mitigate the issue of self-heating in the thermistor.

The frequency of the oscillations of the phase shift oscillator is governed by the following formula [66]:

$$f_r = \frac{1}{2\pi RC\sqrt{2N}} \tag{5.8}$$

Where:

f_r is the Output Frequency in Hz

R is the Resistance in Ω

C is the Capacitance in F

N is the Number of RC stages ($N = 3$)

Based on the phase shift oscillator design by Dr. Peter Farrell from the University of Melbourne, the following circuit was simulated in LTSpice [23]:

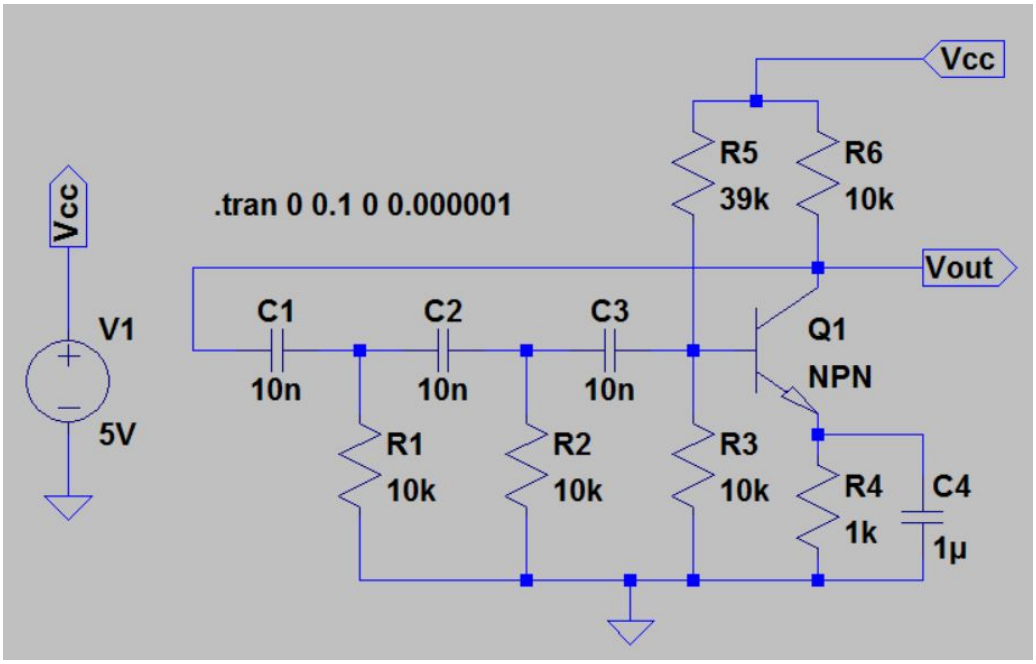

Figure 5.24: Phase Shift Oscillator Circuit in LTSpice [23]

The above circuit was assembled using a BC548 as the NPN transistor. [**?**]

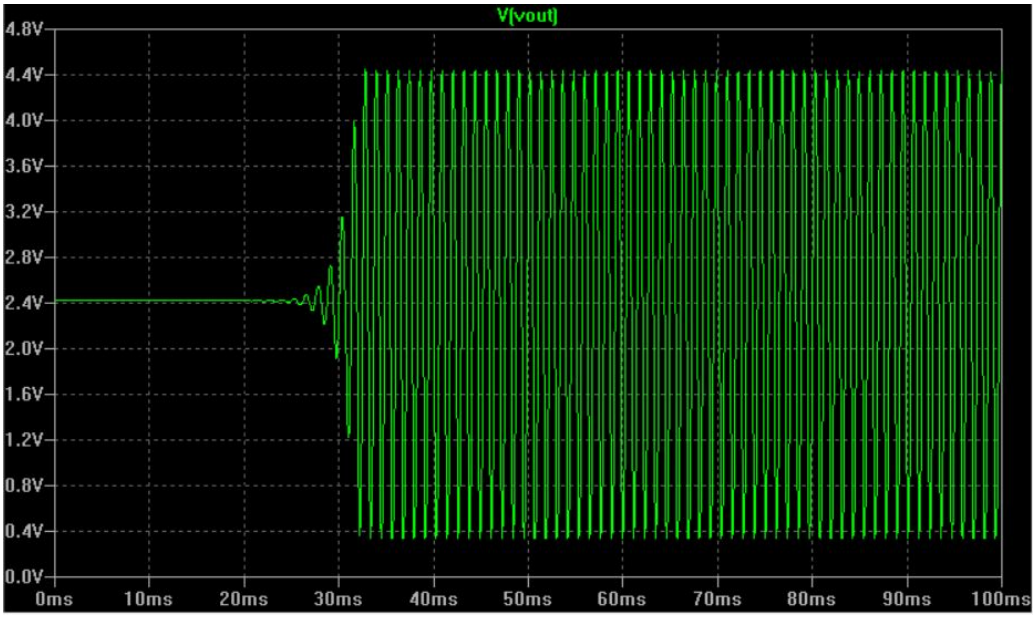

Figure 5.25: Phase Shift Oscillator LTSpice Simulation Voltage Output [23]

Figure 5.26: Phase Shift Oscillator LTSpice Simulation Voltage Output (Zoomed In) [23]

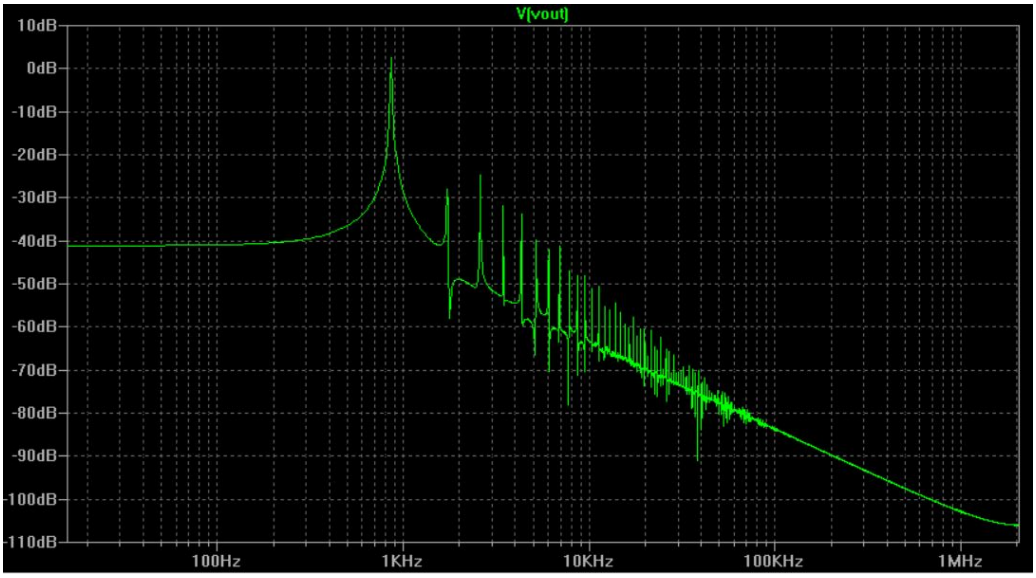

Figure 5.27: Phase Shift Oscillator LTSpice Fast Fourier Transform (FFT) Simulation [23]

The dominant frequency of the phase shift oscillator at equilibrium ($R_1 = 10k\Omega$) is about 860 Hz from the FFT plot in Figure 5.27. By replacing R_1 with a thermistor, the frequency becomes temperature-dependent. Using an Arduino Uno, it is possible to deduce the frequency of the output by measuring the period, which is time between two consecutive transitions from HIGH to LOW.

Calibration of Phase Shift Oscillator

In order to integrate the thermistor into the circuit in Figure 5.24, it is necessary to empirically calibrate the Arduino Uno for the measured frequency to correspond to the temperature. As such, R_1 is replaced with differing values of resistances to simulate the varying resistance of the thermistor and to observe its correlation with the temperature based on the thermistor datasheet. The formula used is Equation 5.4.

Procedure:

1. Choose random values of resistors (in the range 3kohms to 25kohms) based on the range of temperatures desired.

2. Connect the resistor to the point where the thermistor is supposed to be found.

3. Record the average of the frequencies using the Arduino.

4. Record resistance and temperature in a table in Microsoft Excel.

5. Calculate T based on formula on previous page.

6. Plot T against R and find trendline.

7. Insert trendline (best fit line) equation into Arduino (calibration). Now rewrite code for Arduino so it outputs T.

Values of R chosen: $16k\Omega$, $18k\Omega$, $6.2k\Omega$, $9k\Omega$, $7.5k\Omega$, $18k\Omega$, $18k\Omega$, $22k\Omega$, $3.9k\Omega$, $16k\Omega$, $10k\Omega$, $12k\Omega$, $23k\Omega$, $9k\Omega$, $25k\Omega$, $6.7k\Omega$, $22k\Omega$, $13k\Omega$, $3.6k\Omega$

The measurement results can be found in Appendix C.2. The tables in this Appendix give rise to the following summarised results in table 5.3.

Frequency (Average) (Hz)	**Resistor (k ohm)**	**Temperature ($^{\circ}C$)**
713.84	16	14.62522155
793.2	10	25.00009198
759.49	12	20.89759901
668.81	23	7.043807812
810.8	9	27.41774841
657.95	25	5.351508212
868.13	6.7	34.38035887
667.96	22	7.953407074
746.29	13	19.12817814
1008.56	3.6	50.02636632
712.37	16	14.62522155
694.47	18	12.12527069
885.42	6.2	36.2584574
810.08	9	27.41774841
848.1	7.5	31.68532759
695.69	18	12.12527069
694.83	18	12.12527069
668.5	22	7.953407074
984.74	3.9	47.93002289

Table 5.3: Summary of Output Frequency, R_1 Resistance, and Temperature

Using the results from Table 5.3, a trendline is plotted in Microsoft Excel. A logarithmic trendline suits the data better than a quartic trendline.

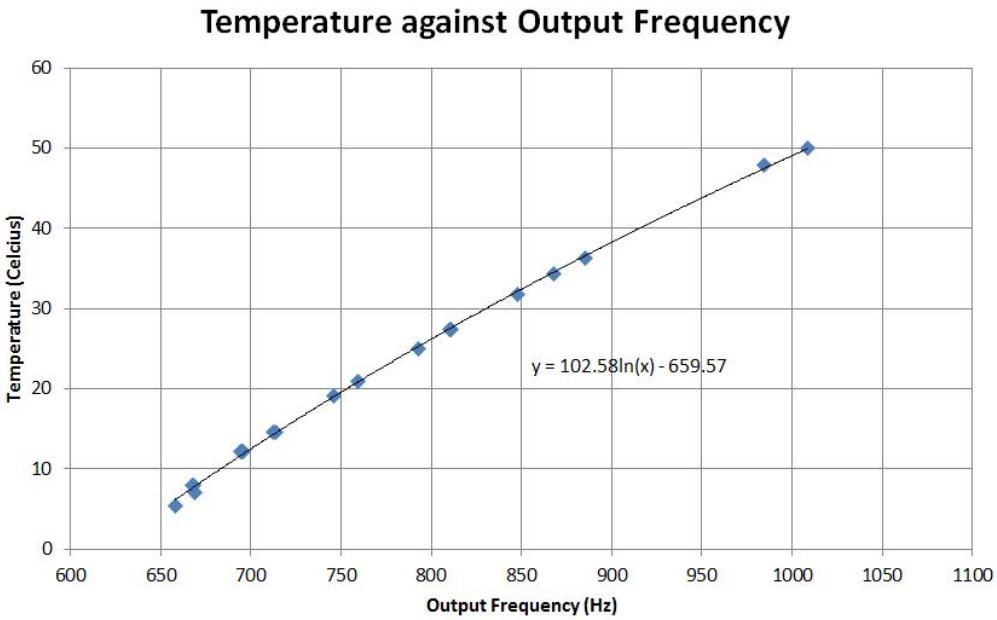

Figure 5.28: Temperature against Output Frequency of Phase Shift Oscillator

The relationship between temperature of the thermistor and the measured output frequency of the phase shift oscillator (PSO) can therefore be written as Equation 5.9.

$$T = 102.58ln(F) - 659.57 \tag{5.9}$$

The updated Arduino code for directly reading the temperature from a thermistor in a phase shift oscillator configuration can be found in Appendix A.2.5.

5.4.3 Astable Operation of 555 Timer

Another method of signal condition for the thermistor involves using the 555 Timer in astable operation.

555 Timer Datasheet.

5.5 Interface

5.5.1 USB

5.5.2 Arduino Uno / Arduino Pro Mini - Analog to Digital Conversion

Arduino Pro Mini 328 - 3.3V/8MHz

Sensitivity of Arduino Uno

1023 bits

Chapter 6

Data Processing, Visualisation, and Transmission

6.1 Installation of Necessary Applications

This setup is based on the Raspbian Jessie. It is possible to replicate the results with similar Linux-based OS but with a few minor adjustments. It assumes the complete installation of the Raspbian Jessie on the Raspberry Pi as seen in Appendix A.1.

The following programs are required for the complete functionality of the WVSMS using the Raspberry Pi Terminals running on Raspbian Jessie:

- **Tight VNC Server** - Set up VNC server required to access the Raspberry Pi with its internal programs and applications [31].
- **X Tight VNC Viewer** - Allows the Display-connected Terminal to access and view the Sensor-connected Terminal via the VNC server [31].
- **Arduino** - Provides analog-to-digital interface between Sensor and Terminal, converts the signal into a serial input for processing, and visualisation [4].
- **Matplotlib** - Software which renders graphs from text files [42].
- **Processing 2.2.1** - Software sketchbook which receives serial inputs and renders real-time graphs of given information [12].

- **Java 7** - Allows Java-based applications to run on the Platform (specifically required for Processing 2.2.1) [52].

The installation of the first four applications are relatively simple and can be done through running the following commands in the LXTerminal (default terminal emulator) of the Raspberry Pi with an Internet connection:

```
sudo apt-get update
sudo apt-get install tightvncserver
sudo apt-get install xtightvncviewer
sudo apt-get install arduino
sudo apt-get install python-matplotlib
```

It is possible to use the following command to install Processing but only for the latest release of the program. At the time of testing this code, the version of Processing installed is version 3.1.1.

```
curl https://processing.org/download/install-arm.sh | sudo sh
```

However, for the use of the ECG and PPG, Processing 2.2.1 is required, not version 3.1.1. The devices will not function with the newer versions of Processing, displaying "size()" errors. It is necessary to uninstall package 'libgles2-mesa' before using Processing to prevent startup errors related to the P2D and P3D renderers.

To install Processing 2.2.1 [58]:

1. Download Processing 2.2.1 from https://processing.org/download/?processing for Linux 32 platforms.
2. Save processing-2.2.1-linux32.tgz (98.4 MB) to the Desktop in the Raspberry Pi.
3. Extract the tar file using the following command:

   ```
   tar xvzf processing-2.2.1-linux32.tgz
   ```

Processing 2.2.1 requires Java to operate but it only performs well on Oracle Java 7, not Oracle Java 8. By default the Raspbian Jessie ships with Oracle Java 8.

To install Oracle Java 7 [58]:

```
sudo apt-get update
sudo apt-get install oracle-java7-jdk
sudo update-alternatives --config java
rm -rf ~/processing-2.2.1/java
ln -s /usr/lib/jvm/jdk-7-oracle-armhf ~/processing-2.2.1/java
```

The last two commands remove the x86 Java runtime and replace with Raspberry Pi armhf version. This is required due to the different processor architecture which the Raspberry Pi is built on, namely the ARM architecture.

Should the folder name be different, the last command will not work. In that case, copy '/usr/lib/jvm/jdk-7-oracle-arm-vfp-hflt' to the 'java' subfolder in the "processing-2.2.1" folder [58].

It is also necessary to install the Java Simple Serial Connector (jSSC) for the serial connection to work between the port and Processing 2.2.1 [58]. The zip file can be obtained from the following link:

```
http://code.google.com/p/java-simple-serial-connector/downloads/detail?name=jSSC
    -2.6.0-Release.zip&can=2&q=
```

To install the Java Simple Serial Connector (jSSC) [58]:

```
unzip jSSC-2.6.0-Release.zip
mv jSSC-2.6.0-Release/jssc.jar ~/processing-2.1/modes/java/libraries/serial/
    library/
```

The latter command overwrites the default jssc.jar file of Processing 2.2.1 with the 2.6.0 Release.

6.2 Wireless Transmission: Operation of VNC Client-Server System

For this project, wireless communications between the Sensor-connected Terminal and the client requires the use of VNC as seen in Section 2.11.1.

The VNC server used for this project is the Tight VNC Server [31]. After the installation of the program in Section 6.1 above, the VNC server has to be set up before the Raspberry Pi

becomes externally accessible as seen in Figure 6.1.

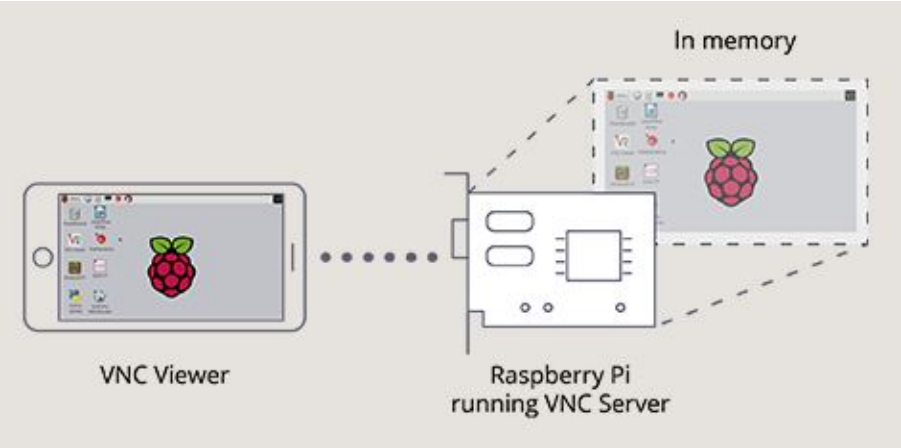

Figure 6.1: Thermistor Formulae and Properties [31]

To do this, run the following commands in LXTerminal:

```
sudo tightvncserver
vncserver :2 -geometry 1920x1080 -depth 24
hostname -I
```

The first command will establish the initial settings for a VNC server and prompt for a password to authenticate and verify the user.

The second command sets up a VNC server on the Raspberry Pi. ":2" tells the Raspberry Pi to establish the server in session 2. The resolution is set to 1920x1080 with the "-geometry" option.

The third command reveals what IPv4 address has been allocated to the Raspberry Pi.

The Raspberry Pi has to be connected to a WiFi network for the external connection and accessibility of its VNC server. There are two main options, namely:

1. An externally established WiFi network (through a router or gateway)
2. An ad hoc network set up by the Raspberry Pi

The latter choice is preferred to isolate the Raspberry Pi from a dependence on external networks. However, as a proof of concept, the former option was used for this project through a hotspot created on a local laptop with a WiFi adapter.

Both the Sensor-connected Terminal and Display-connected Terminal must be connected on the same WiFi network.

6.2.1 Access from Another Raspberry Pi

From the Display-connected Terminal, run Tight VNC Viewer using the following commands:

```
sudo xtightvncviewer
```

A prompt for the IP address and password of the VNC server will appear. After completing the details as found using the 'hostname-I' command, the graphical user interface to access the Sensor-connected Terminal will start.

6.2.2 Access from a Windows Platform

On the Windows platform, download and install TightVNC Viewer [60]. After completion, run TightVNC Viewer and input the IP address of the Raspberry Pi followed by the session number (':1' or ':2'). For instance: '192.168.1.2:2'. Enter the password and click 'Connect'. The graphical user interface for the Raspberry Pi will appear as in Figure 6.2.

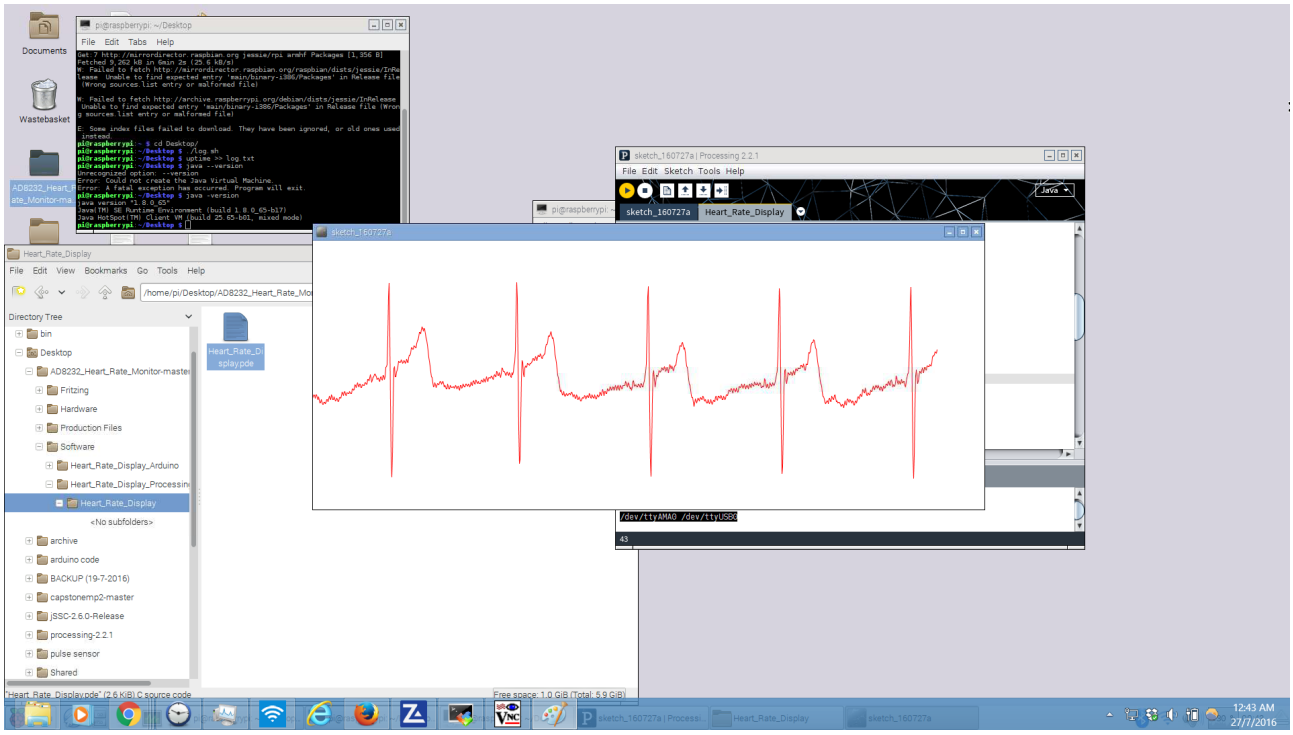

Figure 6.2: Actual Raspberry Pi VNC Server Viewed through a VNC Viewer in a Windows Environment Wirelessly over WiFi

6.3 ECG Software

The Arduino sketch from SparkFun's AD8232 Heart Rate Monitor GitHub repository must first be uploaded to the Arduino Pro Mini [63] which can be found in Appendix A.2.1. It is important to note the board type when uploading the sketch, that is to choose Arduino Pro Mini 328 - 3.3V/8MHz.

Next, run Processing 2.2.1 and open the pde type file from the same repository [63] which can be found in Appendix A.2.2. Change 'myPort = new Serial(this, Serial.list()[1], 9600);' depending on which serial port is used.

6.4 Thermistor with Phase Shift Oscillator Software

The Arduino sketch written for the phase shift oscillator must first be uploaded to the Arduino Uno which can be found in Appendix A.2.5.

Chapter 7

Simulation

In the early phases of development, it was necessary to simulate the ECG signals as inputs into the system to observe both how the signals would be transmitted (through raw data transmission), and the typical form of the output waveform. To achieve this, an ECG signal was produced and plotted in MATLAB using the code from PhysioNet which can be found in Appendix A.2.6 [53].

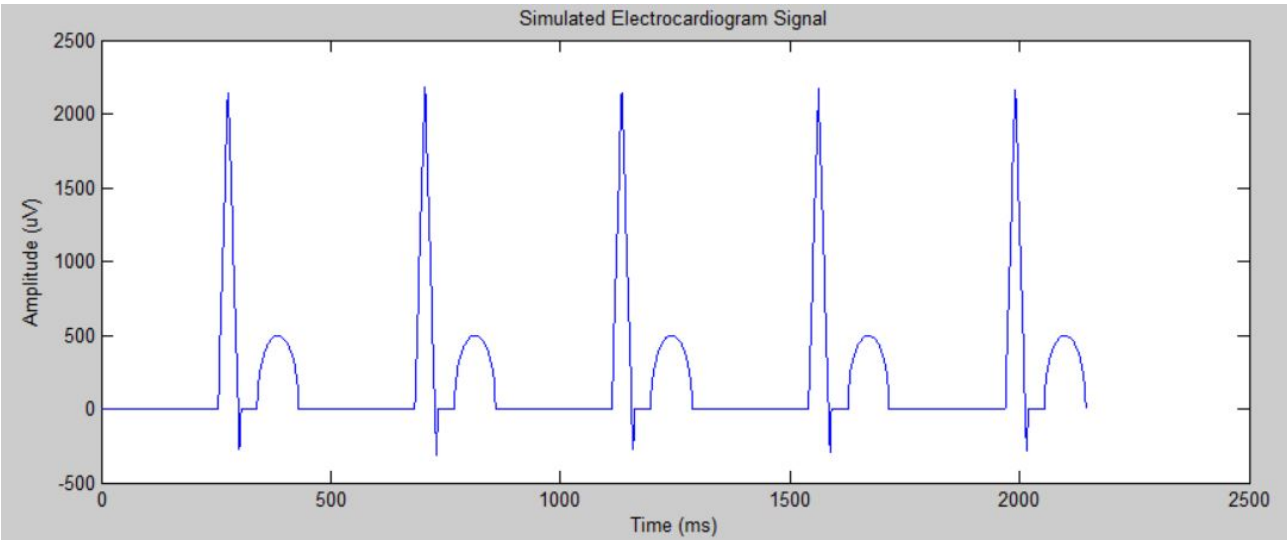

Figure 7.1: Simulated ECG Signal

Chapter 8

Testing

Chapter 9

Results

9.1 Wireless Vital Signs Monitoring System

9.1.1 Setup

9.2 Interface Output

9.3 ECG

9.3.1 AD8232 and Arduino Pro Mini Setup

9.3.2 ECG Electrode Placement

For normal ECG systems, 10 cables are sufficient to acquire and display 12 electrical perspectives of the heart [6]. The typical attachment sites are found in Figure 9.1 below.

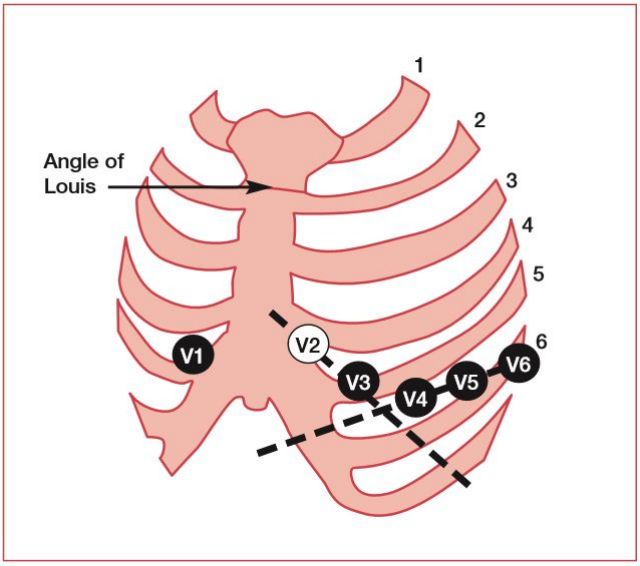

Figure 9.1: Typical Sensor Placements [6]

The AD8232 Heart Rate Monitor from SparkFun Electronics [14] provides three separate sensor pads and should be located in close proximity to the right arm, left arm, and the right leg, as seen in Figure 9.2 below.

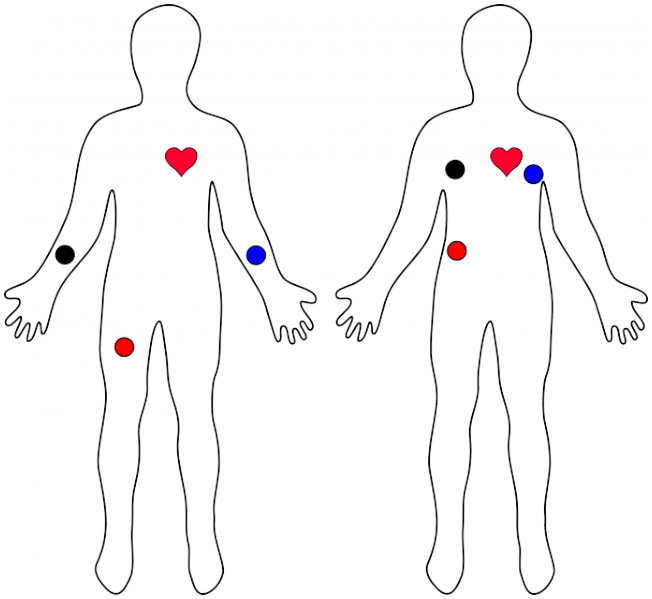

Figure 9.2: Typical Sensor Placements (3 Electrodes) [14]

In compliance with the suggested sites of sensor attachments, the approximate position of the electrodes are:

- 1 inch above the right nipple
- 1 inch above the left nipple
- 2.5 inches right of the navel

9.3.3 ECG Output

When initially operating the AD8232 Heart Rate Monitor, the ECG waveform output was not in the form of a recognisable heartbeat as seen in Figure 9.3. Several adjustments were necessary to reduce noise in the ECG output and to capture the heartbeat waveform.

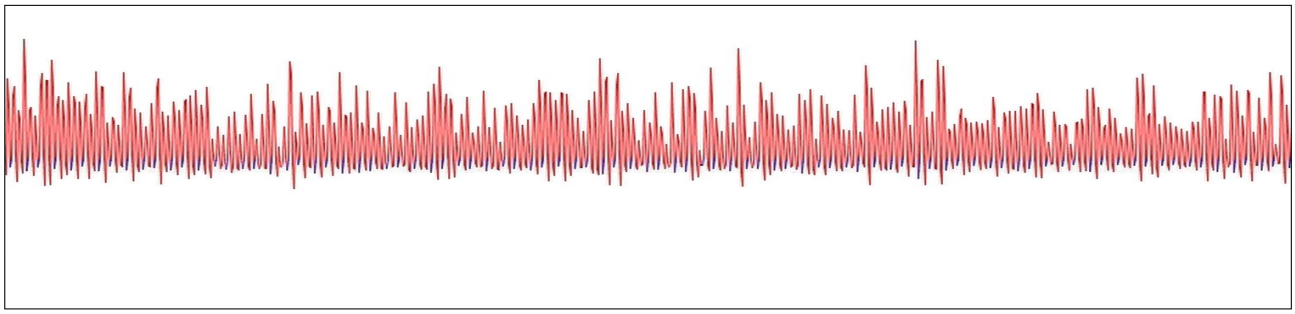

Figure 9.3: ECG Output Test 2

When conducting the experiments, it is noted that the ECG output is significantly affected by posture and position of the body. This is explained more in Section 9.3.4. The best results were obtained from an upright sitting position with hands resting in front on a horizontal platform.

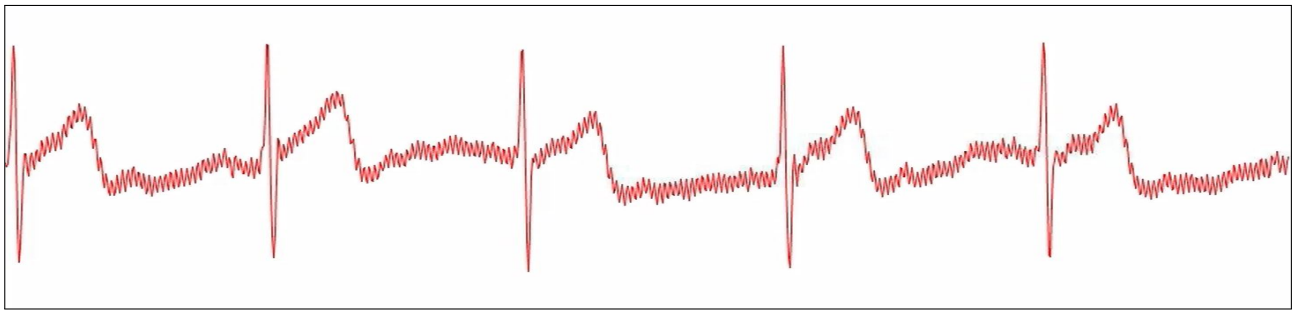

Figure 9.4: ECG Output recorded by the Raspberry Pi over WiFi

9.3.4 Motion Artifacts

At its core, ECG records electrical activity from muscle contractions through electrodes located on the skin [6]. As such, ECG systems are significantly affected by muscle contractions regardless of its origin. In measuring heart electrical activity, it is inevitable that electrical signals from other muscles in the body will be registered by the ECG system. Therefore, the output of ECG systems are highly variable and extremely susceptible to muscle movement, which introduces unwanted noise into the heart ECG output. This includes deep breathing or any movement of limbs while measurements are taken.

The AD8232 cardiac circuit configuration from the datasheet "assumes that the patient remains relatively still during the measurement, and therefore, motion artifacts are less of an issue" [1].

Figure 9.5 illustrates how breathing can affect the vertical axis of the ECG output.

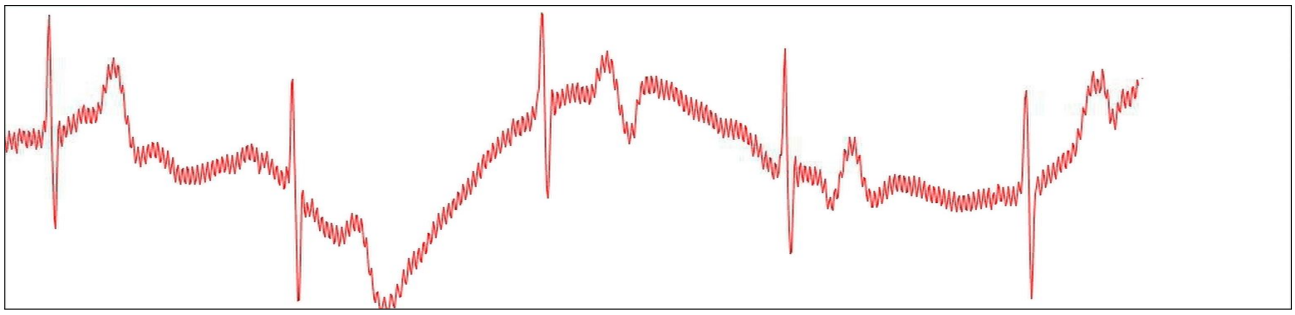

Figure 9.5: ECG Output with Deep Breathing

Figure 9.6 illustrates how moving limbs can affect the output of the ECG system.

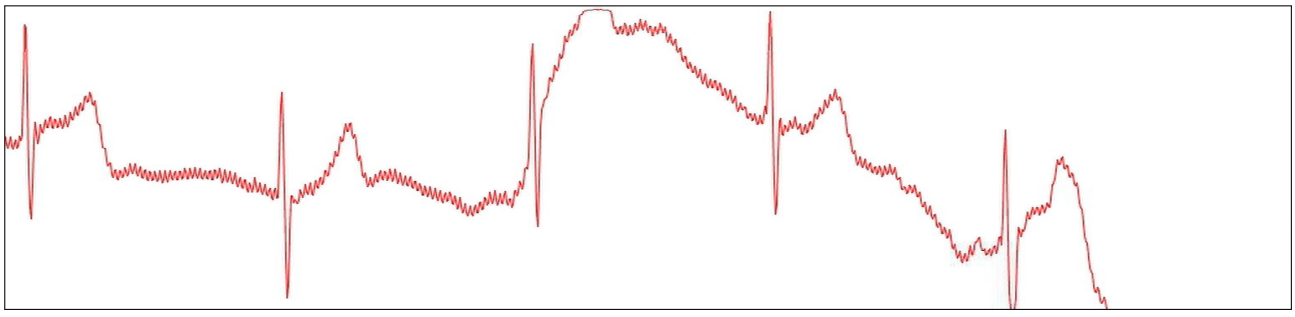

Figure 9.6: ECG Output with Limb Movement

Inconsistent ECG readings are also caused by actual abnormalities or deficiencies in the heart but these cases are not discussed in this report as they are not relevant to the inherent sources

of errors found in the ECG system.

9.4 EEG

9.4.1 Operation

For normal EEG operations, EEG electrodes will be placed evenly over the scalp of the head (roughly 20 electrodes on a mesh circuit). A typical attachment site can be seen in the graph below. However, for the proof of concept of application, only 2 electrodes are used for simplicity to collect data. This is due to the complexity in designing an appropriate amplifying circuit with filtering for 20 electrodes. The chip used in our design, NeuroSky TGAM module has the capability to process information from one electrode with reference to a reference electrode placed at the ear lobe.

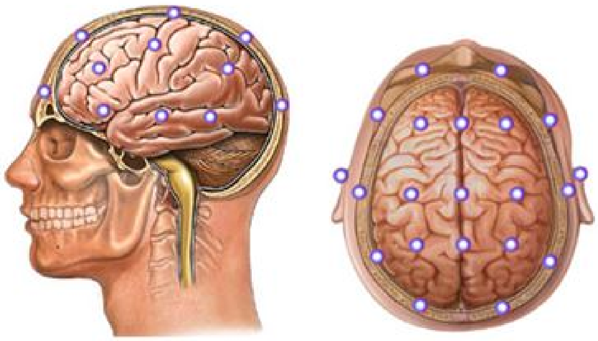

Figure 9.7: Points for Placement of Macroelectrodes in EEG

PLEASE CITEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

The TGAM chip is connected to an electrode place on the forehead/ top of the skull for our measurements. The chip has and inbuilt function to filter off and reset if the incoming signal is too noisy/small to be analysed.

The ideal output will be similar to:

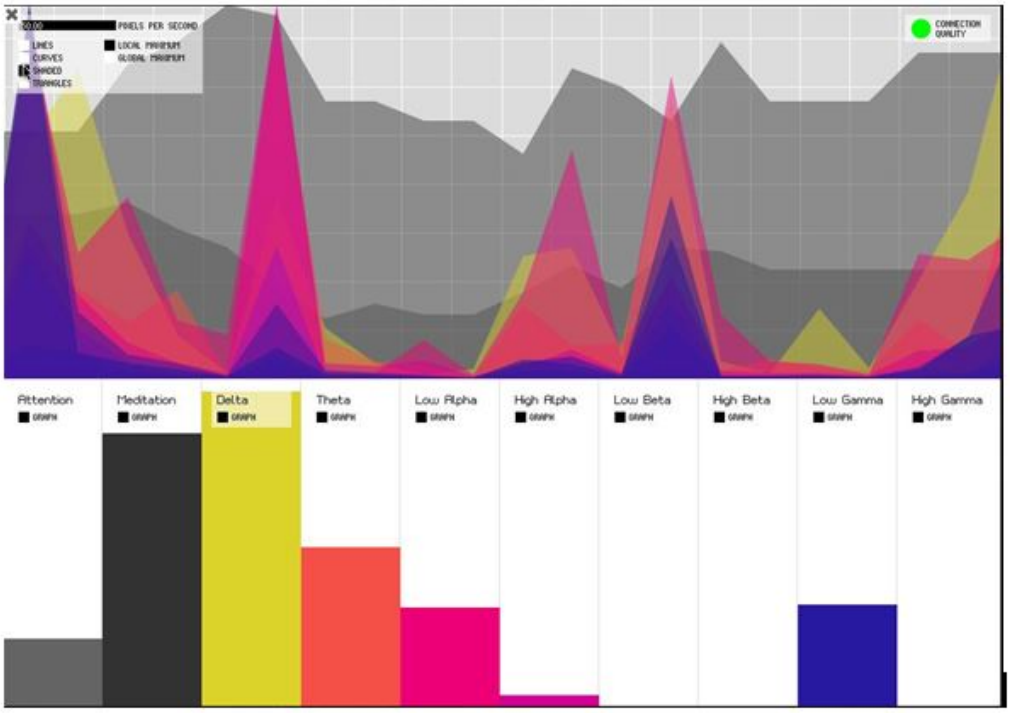

Figure 9.8: Expected EEG Output

However, for the output that is colleceted from the setup:

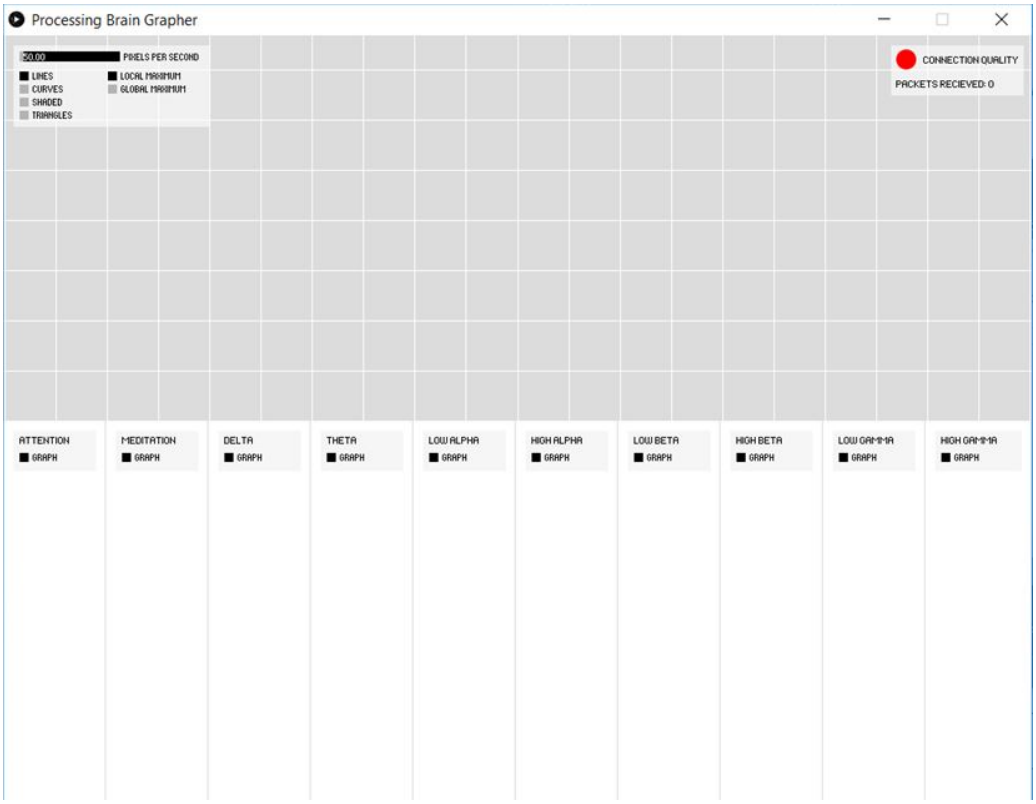

Figure 9.9: Actual EEG Output

From the code at the appendix, we can see that the Neurosky module is pre-loaded with schemes to output no results when the input is too unreliable. This can be due to artifacts discussed earlier, contact resistance and imperfections in soldering as well as losses in the circuitry as the signal is very minute. A second attempt on building an individual EEG circuit from scratch will be conducted to amplify and filter the signal before feeding it into the TGAM chip.

9.5 PPG

9.5.1 Prototype Sensor with Arduino Uno

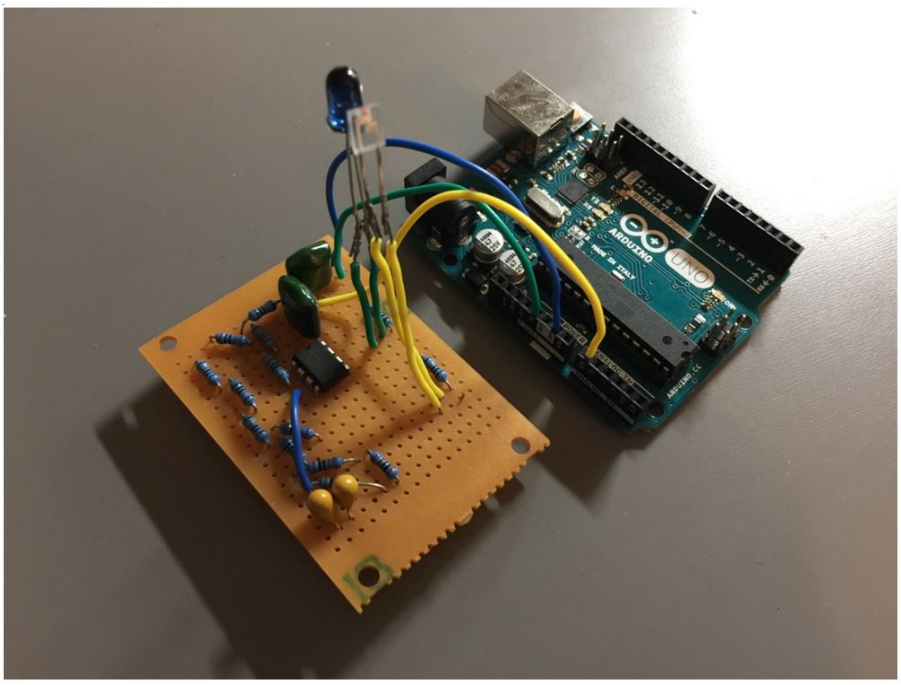

Figure 9.10: Prototype Sensor with Arduino Uno

Processing 2+ is used for waveform visualization:

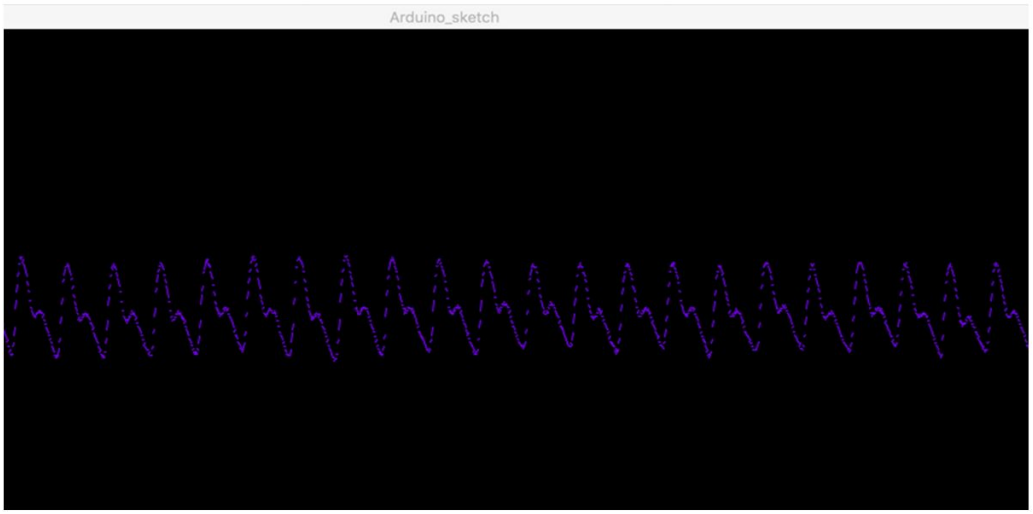

Figure 9.11: Actual PPG Sensor Output

Comparison with the PPG wave from medical standard monitor display:

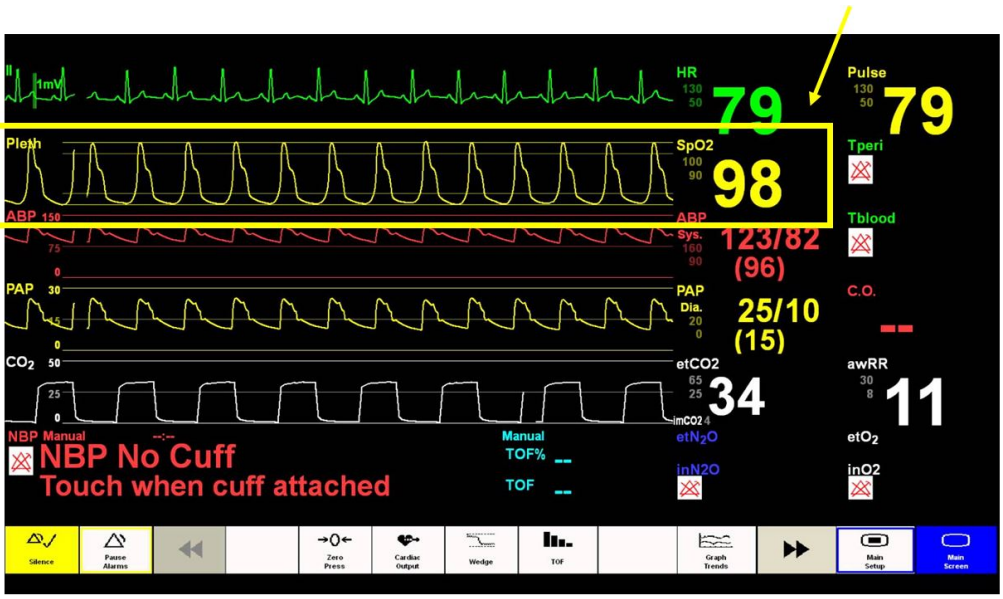

Figure 9.12: Expected PPG Sensor Output

9.5.2 Heart Rate Measurement

The basic idea of calculating heart rate with the data collected by pulse oximeter is measuring the period of each periodic pulse and calculate the frequency. For normal sinusoidal waves, this is very easy to be achieved by finding the DC offset of the waveform and measure the time difference between every time the signal goes above that DC offset. Then the frequency can be calculated from the time difference.

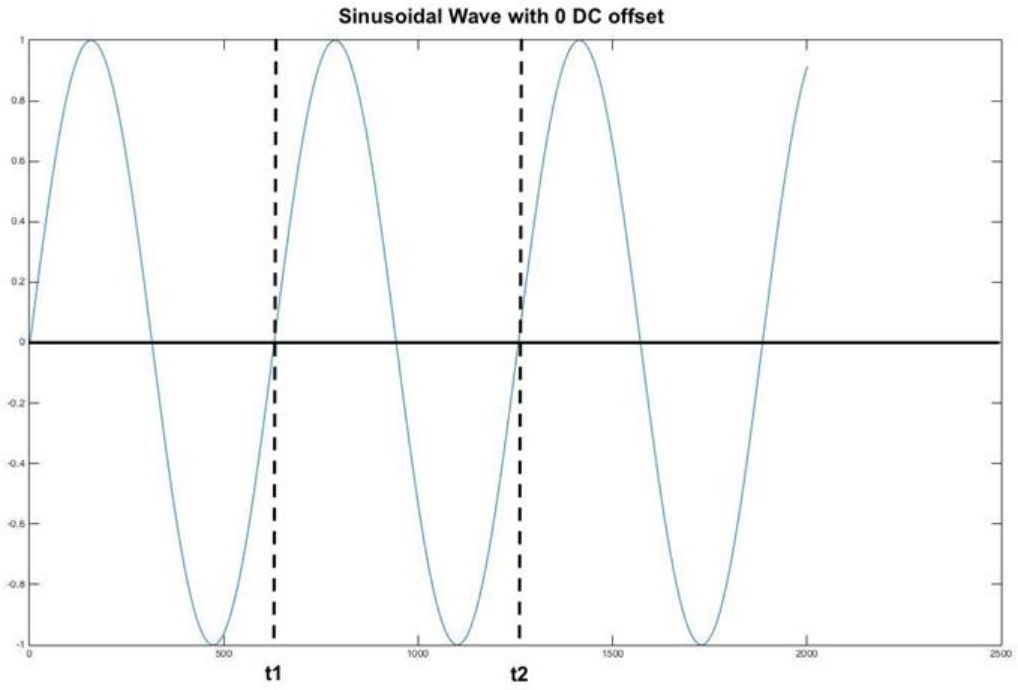

Figure 9.13: Sinusoidal Wave with 0 DC Offset

However, the pulse oximeter is very sensitive to any physical movement. The amplitude of the waveform is relatively stable if you keep your finger at the same position. But the movement of the pulse oximeter itself can cause the change in DC offset of the signal.

In order to mitigate the fluctuation of DC offset, we implement a moving average filter with window size of 400 samples. For normal human resting heart rate, each periodic pulse takes approximately 200 samples. With a window size of two periods, the moving average can be used as the DC offset for calculating the heart rate.

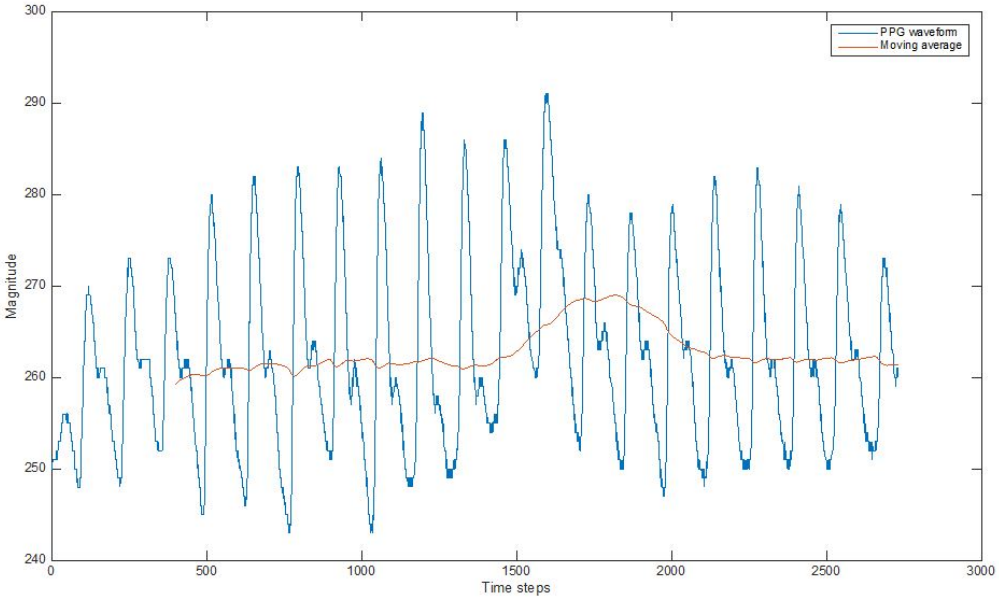

Figure 9.14: Moving average DC offset (Matlab plot)

From the plot above, we can observe the moving average line behaves as a trend line for the PPG waveform.

9.5.3 Abstract AC/DC Values

As explained in the previous, the modulation ratio R is the key parameter for calculating Oxygen Saturation level SpO2.

For DC component value for Red/IR signals, the moving average can be used here as the. The (might need one more page later)

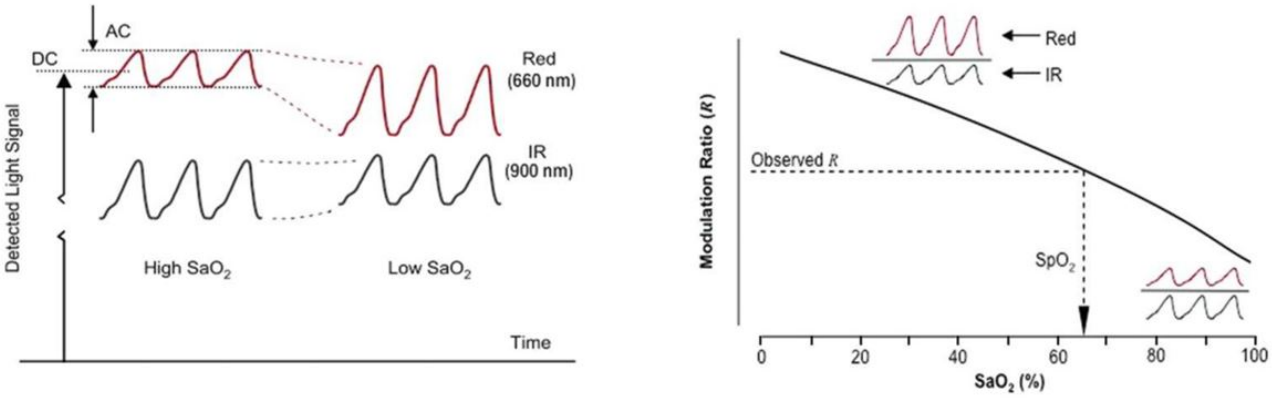

Figure 9.15: Detected Light Signal and Modulation Ratio for $SaO_2\%$

9.6 Statistics

SRW

Chapter 10

Discussion

10.1 Challenges and Limitations

10.2 Power Efficiency

According to Sohraby, Minoli, and Taieb [61], power efficiency can be achieved by having:

1. "Low-duty-cycle operation."
2. "Local/in-network processing to reduce data volume (and hence transmission time)."
3. "Multihop networking reduces the requirement for long-range transmission since signal path loss is an inverse exponent with range or distance. Each node in the sensor network can act as a repeater, thereby reducing the link range coverage required and, in turn, the transmission power."

10.3 Interference

One of the major problems identified with wireless systems in general is the high probability of interference.

10.4 Security Risk

10.5 Data Management

10.6 Regulation and Compliance Standards

10.7 Reliability

Raspberry Pi has "uptime"

10.8 Range

10.9 Raspberry Pi USB

Only a single root USB port is available on the Raspberry Pi 3 Model B and all data traffic from USB devices are directed to this single bus [28]. The maximum speed of this root USB port is 480Mbps [25]. The WVSMT utilises all 4 USB ports available for connections to individual Sensors and this could possibly lead to bottlenecking in the root USB port. However, it is noted that the bandwidth required for each sensor is low and that this scenario is not probable.

Chapter 11

Conclusion

11.1 Summary of Main Results

11.2 Practical Interpretation of our Results

To provide some practical interpretation of our results, we may consider the following hypothetical scenario. A control engineer has designed a particular periodic discrete or sampled-data controller K for his closed loop feedback system. The engineer finds out about our results and is interested to hear that a time invariant controller can in general give strictly better disturbance rejection performance than his periodic controller. The engineer then asks us to tell him how to modify his periodic controller to obtain a superior time invariant controller. To advise him we

Bibliography

[1] Inc. Analog Devices. Ad8232 datasheet: Single-lead, heart rate monitor front end. http://www.analog.com/media/en/technical-documentation/data-sheets/AD8232.pdf. Accessed: 2016-9-13.

[2] Inc. Analog Devices. Ad8232-evalz user guide ug-514. http://www.analog.com/media/en/technical-documentation/user-guides/AD8232-EVALZ_UG-514.pdf. Accessed: 2016-9-13.

[3] Inc. Analog Devices. Ad8232 single-lead heart rate monitor analog front end. http://www.analog.com/en/products/application-specific/medical/ecg/ad8232.html. Accessed: 2016-9-13.

[4] Arduino. Arduino. https://www.arduino.cc/. Accessed: 2016-9-25.

[5] Arduino. A running average class for arduino. http://playground.arduino.cc/Main/RunningAverage. Accessed: 2016-9-30.

[6] Euan A Ashley and Josef Niebauer. *Conquering the ECG*. Remedica, 2004.

[7] Humane Society Veterinary Medical Association. Monitoring the anesthetized patient. http://www.ruralareavet.org/PDF/Anesthesia-Patient_Monitoring.pdf. Accessed: 2016-9-30.

[8] John L Atlee. *Complications in anesthesia*. Elsevier Health Sciences, 2006.

[9] Australian and New Zealand College of Anaesthetists (ANZCA). Guidelines on monitoring during anaesthesia how should i cite thisssssssssssssssssssssssssssssssssssssssssssssssss. *Australian and New Zealand College of Anaesthetists (ANZCA)*, PS18, 2016.

[10] Sangeeta Bagha and Laxmi Shaw. A real time analysis of ppg signal for measurement of spo2 and pulse rate. *International journal of computer applications*, 36(11):0975–8887, 2011.

[11] Flavia Basilotta, Stefano Riario, Francesca Stradolini, Irene Taurino, Danilo Demarchi, Giovanni De Micheli, and Sandro Carrara. Wireless monitoring in intensive care units by a 3d-printed system with embedded electronic. In *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, pages 1–4. IEEE, 2015.

[12] Casey Reas Ben Fry. Processing. `https://processing.org/download/?processing`. Accessed: 2016-9-25.

[13] Inc. Biopac Systems. BIONOMADIX WIRELESS WEARABLE PHYSIOLOGY. `https://www.biopac.com/product-category/research/bionomadix-wireless-physiology/`. Accessed: 2016-10-1.

[14] SparkFun Electronics (R) CaseyTheRobot. Ad8232 heart rate monitor hookup guide. `https://learn.sparkfun.com/tutorials/ad8232-heart-rate-monitor-hookup-guide`, 2014. Accessed: 2016-9-12.

[15] FL Cesarano and AR Piergeorge. The spaghetti syndrome. a new clinical entity. *Critical care medicine*, 7(4):182–183, 1979.

[16] Edward R. Laskowski (Mayo Clinic). What's a normal resting heart rate? `http://www.mayoclinic.org/healthy-lifestyle/fitness/expert-answers/heart-rate/faq-20057979`. Accessed: 2016-9-30.

[17] Wikimedia Commons. File:oxy and deoxy hemoglobin near-infrared absorption spectra.png. `https://commons.wikimedia.org/wiki/File:Oxy_and_Deoxy_Hemoglobin_Near-Infrared_absorption_spectra.png`. Accessed: 2016-9-30.

[18] Jennifer Marsh StorageCraft Technology Corporation. Linux: Advantages and disadvantages of open-source technology. `http://www.storagecraft.com/blog/linux-advantages-disadvantages-open-source-technology/`, 9 2016. Accessed: 2016-9-15.

[19] Kusum Grewal Dangi and Supriya P Panda. Performance analysis of patient monitoring wireless body area networks using queueing models. *International Journal of Applied Engineering Research*, 11(9):6671–6675, 2016.

[20] Ubuntu-Team-Member DeviantArt. 8 advantages of using linux over windows. http://ubuntu-artists.deviantart.com/journal/8-Advantages-of-using-Linux-over-Windows-291681914, 3 2012. Accessed: 2016-9-15.

[21] Casey Kuhns SparkFun Electronics. Ad8232_heart_rate_monitor_v10. https://cdn.sparkfun.com/datasheets/Sensors/Biometric/AD8232_Heart_Rate_Monitor_v10.pdf, 7 2014. Accessed: 2016-9-14.

[22] Pfiedler Enterprises. Wireless technology in the or. http://pfiedler.com/ce/1294/files/assets/common/downloads/Wireless%20Technology%20in%20the%20OR.pdf. Accessed: 2016-10-1.

[23] Peter Farrell. Elen30013 electronic system implementation 2014 lecture (university of melbourne).

[24] Silent H Films. Simulated patient monitor. https://www.youtube.com/watch?v=jcXidSeirMU. Accessed: 2016-9-30.

[25] RASPBERRY PI FOUNDATION. Faqs. https://www.raspberrypi.org/help/faqs. Accessed: 2016-9-26.

[26] RASPBERRY PI FOUNDATION. Noobs. https://www.raspberrypi.org/downloads/noobs/. Accessed: 2016-9-26.

[27] RASPBERRY PI FOUNDATION. Raspberry pi 3 model b. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed: 2016-9-24.

[28] RASPBERRY PI FOUNDATION. Raspberry pi hardware. https://www.raspberrypi.org/documentation/hardware/raspberrypi/. Accessed: 2016-9-26.

[29] RASPBERRY PI FOUNDATION. Raspberry pi weekly. https://www.raspberrypi.org/weekly/connected/. Accessed: 2016-9-26.

[30] RASPBERRY PI FOUNDATION. Raspbian. https://www.raspberrypi.org/documentation/raspbian/. Accessed: 2016-9-26.

[31] RASPBERRY PI FOUNDATION. Vnc (virtual network computing). https://www.raspberrypi.org/documentation/remote-access/vnc/. Accessed: 2016-9-25.

[32] RJ Gajraj, M Doi, H Mantzaridis, and GN Kenny. Comparison of bispectral eeg analysis and auditory evoked potentials for monitoring depth of anaesthesia during propofol anaesthesia. *British Journal of Anaesthesia*, 82(5):672–678, 1999.

[33] Mervin Goldman et al. Principles of clinical electrocardiography. 1976.

[34] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[35] J Guyomard and R Stortelder. Heart rate measurement through ppg: Heartbeat measurement in a wireless headset. 2015.

[36] John E Hall. *Guyton and Hall textbook of medical physiology*. Elsevier Health Sciences, 2015.

[37] Withings Inspire Health. What does spo2 mean? what is a normal spo2 level? https://withings.zendesk.com/hc/en-us/articles/201494667-What-does-SpO2-mean-What-is-a-normal-SpO2-level-. Accessed: 2016-9-30.

[38] Michael Imhoff. The spaghetti syndrome revisited. *Anesthesia & Analgesia*, 98(3):566–568, 2004.

[39] National Instruments. Biomedical engineering education portal. http://www.ni.com/white-paper/5593/en/, 3 2016. Accessed: 2016-9-13.

[40] Texas Instruments. Lm358 (active) dual operational amplifiers. http://www.ti.com/product/LM358. Accessed: 2016-9-30.

[41] Leslie C Jameson and Tod B Sloan. Using eeg to monitor anesthesia drug effects during surgery. *Journal of clinical monitoring and computing*, 20(6):445–472, 2006.

[42] Eric Firing Michael Droettboom John Hunter, Darren Dale and the matplotlib development team. matplotlib. http://matplotlib.org/. Accessed: 2016-9-25.

[43] Erik Kershaw. Wireless networking reshapes the face of patient monitoring systems. *Biomedical instrumentation & technology*, 36(3):201–202, 2002.

[44] R-B (Embedded Lab). Easy pulse sensor (version 1.1) overview (part 1). http://embedded-lab.com/blog/easy-pulse-version-1-1-sensor-overview-part-1/. Accessed: 2016-9-30.

[45] Libelium. e-health sensor platform v2.0 for arduino and raspberry pi [biometric / medical applications]. https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical. Accessed: 2016-10-1.

[46] Nicolas Marchant, Robert Sanders, Jamie Sleigh, Audrey Vanhaudenhuyse, Marie-Aurélie Bruno, Jean François Brichant, Steven Laureys, and Vincent Bonhomme. How electroencephalography serves the anesthesiologist. *Clinical EEG and neuroscience*, page 1550059413509801, 2014.

[47] Johns Hopkins Medicine. Electroencephalogram (eeg). http://www.hopkinsmedicine.org/healthlibrary/test_procedures/neurological/electroencephalogram_eeg_92,p07655/. Accessed: 2016-9-28.

[48] Memscap. Memscap sensor solution. http://www.sintef.no/globalassets/project/tradlospasient/080208-seminar-rikshospitalet/bwsn_memscap.pdf. Accessed: 2016-10-1.

[49] Natus Neurology. Natus eeg webinar: Noise and artifacts sources eeg how to identify, reduce and avoid. https://www.youtube.com/watch?v=CywlDhf2Vno. Accessed: 2016-9-28.

[50] R Joe Noble, J Stanley Hillis, and Donald A Rothbaum. Electrocardiography. 1990.

[51] NXP. Pulse oximeter fundamentals and design. http://www.nxp.com/files/32bit/doc/app_note/AN4327.pdf?tid=AMdlDR. Accessed: 2016-9-30.

[52] Oracle. Java. https://www.java.com/en/. Accessed: 2016-9-25.

[53] PhysioNet. Ecg waveform generator for matlab/octave. https://www.physionet.org/physiotools/matlab/ECGwaveGen/, 6 2016. Accessed: 2016-9-26.

[54] Electrocomponents PLC. Ntc thermistors. http://docs-europe.electrocomponents.com/webdocs/14c2/0900766b814c2f5a.pdf. Accessed: 2016-9-28.

[55] Andy Pybus. Expanding the role of smartphones in the operating theatre. http://www.anzca.edu.au/documents/expanding-the-role-of-smartphones-in-the-operating.pdf. Accessed: 2016-10-1.

[56] Kelli Rosenthal. New frequencies, new monitoring technology. *Nursing management*, 34(3):49–51, 2003.

[57] Praveen Arou (Texas Instruments) Sang-Soo Oak. How to design peripheral oxygen saturation (spo2) and optical heart rate monitoring (ohrm) systems using the afe4403. http://www.ti.com/lit/an/slaa655/slaa655.pdf. Accessed: 2016-9-30.

[58] scruss. Processing 2.1 + oracle java + raspberry pi + serial + arduino = . http://scruss.com/blog/2014/01/07/processing-2-1-oracle-java-raspberry-pi-serial-arduino-%E2%98%BA/. Accessed: 2016-10-1.

[59] Iman Shames. Elen90075 power electronics 2016 workshop 2 (university of melbourne).

[60] TightVNC Software. Tightvnc: Vnc-compatible free remote control / remote desktop software. http://www.tightvnc.com/. Accessed: 2016-9-25.

[61] Kazem Sohraby, Daniel Minoli, and Taieb Znati. *Wireless sensor networks: technology, protocols, and applications*. John Wiley & Sons, 2007.

[62] Radosveta Sokullu, Mustafa Alper Akkas, and Hüseyin Ertürk Çetin. Wireless patient monitoring system. In *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*, pages 179–184. IEEE, 2010.

[63] SparkFun. Ad8232 single lead heart rate monitor (github). `https://github.com/sparkfun/AD8232_Heart_Rate_Monitor`. Accessed: 2016-10-1.

[64] Hans-Ulrich Strohmenger, Karl H Lindner, and Charles G Brown. Analysis of the ventricular fibrillation ecg signal amplitude and frequency parameters as predictors of countershock success in humans. *Chest*, 111(3):584–589, 1997.

[65] Prashant Tripathi, Naman Pal, Paras Kaushal, and Preeti Jaidka. Wireless biomedical parameter monitoring system using zigbee cc2500 rf module. 2015.

[66] Electronics Tutorials. The rc oscillator circuit. `http://www.electronics-tutorials.ws/oscillator/rc_oscillator.html`, 2014. Accessed: 2016-8-27.

[67] Mats KEB Wallin and Samson Wajntraub. Evaluation of bluetooth as a replacement for cables in intensive care and surgery. *Anesthesia & Analgesia*, 98(3):763–767, 2004.

[68] Wikipedia. Beerlambert law. `https://en.wikipedia.org/wiki/Beer%E2%80%93Lambert_law`. Accessed: 2016-9-30.

Appendix A

Program Installation

A.1 Operating System - Raspbian Jessie

The operating system used for the project is Raspbian Jessie. To simplify installation, the SD card of the Raspberry Pi was flashed with an OS installer containing Raspbian Jessie, called NOOBS (New Out Of the Box Software) [26]. When starting up, the loading screen will look typically like Figure A.1. The version of NOOBS used was version 1.9.2, with a release date of 27/5/2016.

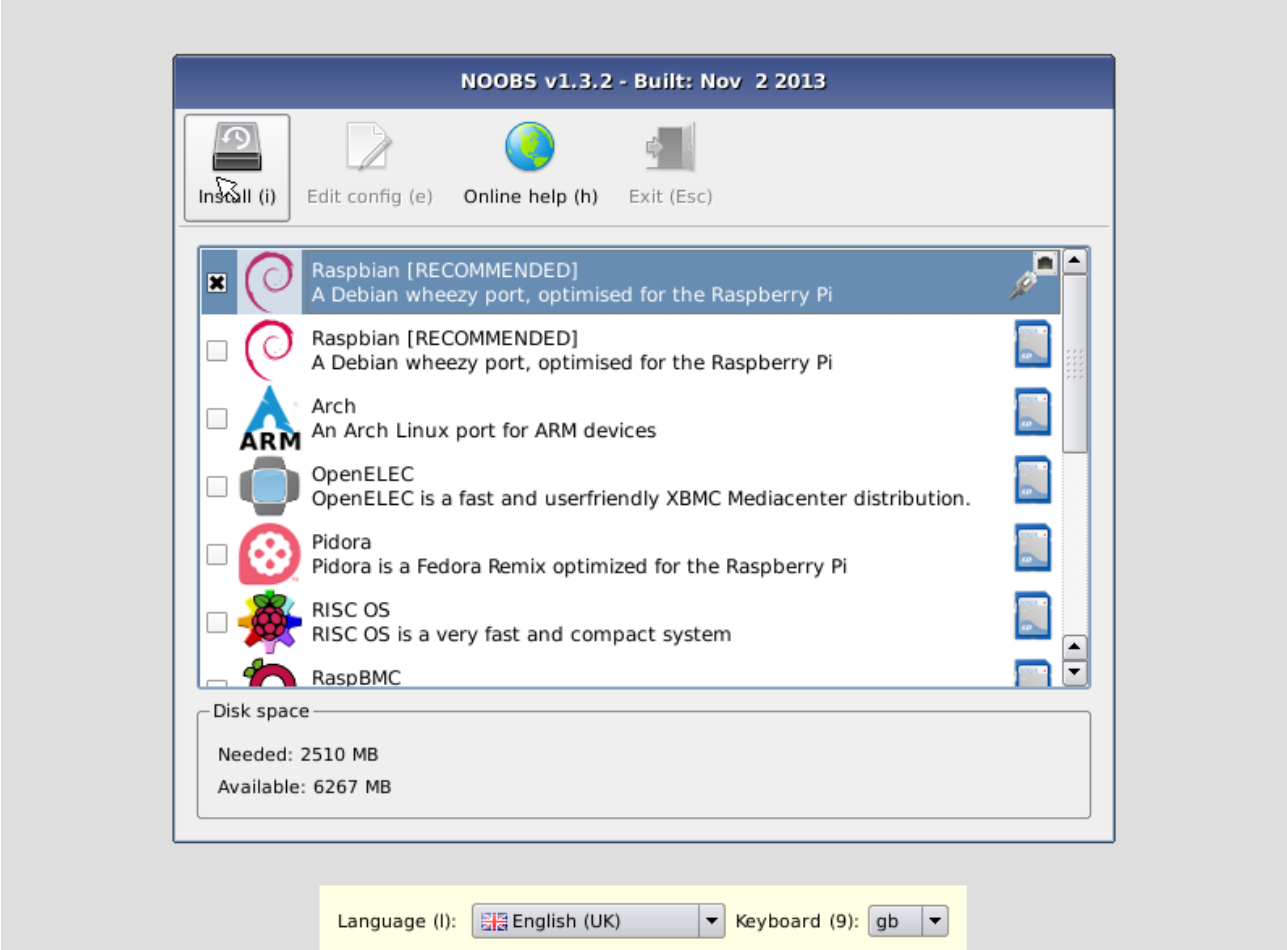

Figure A.1: NOOBS (New Out Of the Box Software) Initial Installation Screen [26]

Aside from using NOOBS for the OS installation, it is possible to download the Raspbian Jessie image from the official Raspberry Pi website, unzip the image file, and copy it to the SD card. However, NOOBS was used specifically for its convenience, since it is necessary to obtain disk imager utilities on Windows for this method. Nevertheless, the choice of installation approach does not in any way affect the functionality of the OS after it has been loaded.

```
sudo apt-get update
sudo apt-get install tightvncserver
sudo apt-get install xtightvncviewer
sudo apt-get install arduino
sudo apt-get install python-matplotlib
```

A.2 Code

A.2.1 ECG Code for Arduino Pro Mini

The following source code is sourced from SparkFun's AD8232 Heart Rate Monitor GitHub repository [63]. This is saved as Heart_Rate_Display_Arduino.ino.

```
/*****************************************************************************
Heart_Rate_Display.ino
Demo Program for AD8232 Heart Rate sensor.
Casey Kuhns @ SparkFun Electronics
6/27/2014
https://github.com/sparkfun/AD8232_Heart_Rate_Monitor
The AD8232 Heart Rate sensor is a low cost EKG/ECG sensor.  This example shows
how to create an ECG with real time display.  The display is using Processing.
This sketch is based heavily on the Graphing Tutorial provided in the Arduino
IDE. http://www.arduino.cc/en/Tutorial/Graph
Resources:
This program requires a Processing sketch to view the data in real time.
Development environment specifics:
IDE: Arduino 1.0.5
Hardware Platform: Arduino Pro 3.3V/8MHz
AD8232 Heart Monitor Version: 1.0
This code is beerware. If you see me (or any other SparkFun employee) at the
local pub, and you've found our code helpful, please buy us a round!
Distributed as-is; no warranty is given.
*****************************************************************************/

void setup() {
// initialize the serial communication:
Serial.begin(9600);
pinMode(10, INPUT); // Setup for leads off detection LO +
pinMode(11, INPUT); // Setup for leads off detection LO -

}

void loop() {
```

```
if((digitalRead(10) == 1)||(digitalRead(11) == 1)){
Serial.println('!');
}
else{
// send the value of analog input 0:
Serial.println(analogRead(A0));
}
//Wait for a bit to keep serial data from saturating
delay(1);
}
```

The method to upload is as the following figure:

Figure A.2: Upload sketch to Arduino Pro-Mini [14]

A.2.2 ECG Code for Processing 2.2.1

The following source code is sourced from SparkFun's AD8232 Heart Rate Monitor GitHub repository [63]. This is saved as Heart_Rate_Display.pde.

```
/*****************************************************************************
Heart_Rate_Display.ino
Demo Program for AD8232 Heart Rate sensor.
```

```
Casey Kuhns @ SparkFun Electronics
6/27/2014
https://github.com/sparkfun/AD8232_Heart_Rate_Monitor
The AD8232 Heart Rate sensor is a low cost EKG/ECG sensor.  This example shows
how to create an ECG with real time display.  The display is using Processing.
This sketch is based heavily on the Graphing Tutorial provided in the Arduino
IDE. http://www.arduino.cc/en/Tutorial/Graph
Resources:
This program requires a Processing sketch to view the data in real time.
Development environment specifics:
IDE: Arduino 1.0.5
Hardware Platform: Arduino Pro 3.3V/8MHz
AD8232 Heart Monitor Version: 1.0
This code is beerware. If you see me (or any other SparkFun employee) at the
local pub, and you've found our code helpful, please buy us a round!
Distributed as-is; no warranty is given.
*************************************************************************/

import processing.serial.*;

Serial myPort;        // The serial port
int xPos = 1;         // horizontal position of the graph
float height_old = 0;
float height_new = 0;
float inByte = 0;



void setup () {
// set the window size:
size(1000, 400);

// List all the available serial ports
println(Serial.list());
// Open whatever port is the one you're using.
myPort = new Serial(this, Serial.list()[2], 9600);
// don't generate a serialEvent() unless you get a newline character:
myPort.bufferUntil('\n');
// set inital background:
```

```
background(0xff);
}


void draw () {
// everything happens in the serialEvent()
}


void serialEvent (Serial myPort) {
// get the ASCII string:
String inString = myPort.readStringUntil('\n');

if (inString != null) {
// trim off any whitespace:
inString = trim(inString);

// If leads off detection is true notify with blue line
if (inString.equals("!")) {
stroke(0, 0, 0xff); //Set stroke to blue ( R, G, B)
inByte = 512;  // middle of the ADC range (Flat Line)
}
// If the data is good let it through
else {
stroke(0xff, 0, 0); //Set stroke to red ( R, G, B)
inByte = float(inString);
}

//Map and draw the line for new data point
inByte = map(inByte, 0, 1023, 0, height);
height_new = height - inByte;
line(xPos - 1, height_old, xPos, height_new);
height_old = height_new;

// at the edge of the screen, go back to the beginning:
if (xPos >= width) {
xPos = 0;
background(0xff);
```

```
}
else {
// increment the horizontal position:
xPos++;
}

}
}
```

The method to run is as in the following figure:

Figure A.3: Run the Processing sketch [14]

A.2.3 PPG with Processing 2+

```
import processing.serial.*;

Serial myPort;          // The serial port
int xPos = 1;           // horizontal position of the graph
float inByte = 0;
```

```
float oldinByte = 0;
PrintWriter output;   // Create an object for exporting data
PrintWriter output_average;

int LM_SIZE = 750;
float[] LM =  new float[LM_SIZE];      // LastMeasurements
int index = 0;
static float sum = 0;
int count = 0;

float movingAverage = 0;
double previous_time = 0;
double current_time = 0;
double time_difference = 0;
double heart_rate = 0;
float input;

int peak = 0;
int trough = 0;

void setup () {

// Create a new file in the sketch directory
//output = createWriter("infrared_data.txt");
//output_average = createWriter("movingAverage.txt");

// set the window size:
size(980, 420);

// List all the available serial ports
// if using Processing 2.1 or later, use Serial.printArray()
println(Serial.list());

// I know that the first port in the serial list on my mac
// is always my  Arduino, so I open Serial.list()[0].
// Open whatever port is the one you're using.
myPort = new Serial(this, Serial.list()[2], 9600);
```

```
// don't generate a serialEvent() unless you get a newline character:
myPort.bufferUntil('\n');

// set inital background:
background(0);
}
void draw () {
// draw the line:
stroke(127, 34, 255);

//if (inByte > 30 && inByte < 600) {
line(xPos-1, (height-oldinByte), xPos, (height-inByte));
//text("received: " + inByte,10,50);
// at the edge of the screen, go back to the beginning:
if (xPos >= width) {
xPos = 1;
background(0);
} else {
// increment the horizontal position:
xPos++;
}
//}

}

void serialEvent (Serial myPort) {
// get the ASCII string:
String inString = myPort.readStringUntil('\n');

if (inString != null) {

// Store the old value for line drawing purpose
oldinByte = inByte;
// trim off any whitespace:
//inString = trim(inString);
// convert to an int and map to the screen height:
inByte = float(inString);
input = inByte;
```

```
//println("Input = " + input);
movingAverage = runningAverage(input);
//println(" Average = " + movingAverage);
//output.println(input);
//output_average.println(movingAverage);
inByte = map(inByte*30, 0, 800, 0, height-500);

// Wait 3 seconds before calculating heart rate
if ( millis() > 3000){
// Calculating heart rate

// Detects peak of the period
if (input > movingAverage && (peak == 0)){

current_time = millis();
time_difference = current_time - previous_time;
//println("Current_time = " + current_time);
//println("previous_time = " + previous_time);
heart_rate = 60.0/(time_difference*0.001);
previous_time = current_time;
println("heart rate = " + heart_rate);
peak = 1;
trough = 0;
}

// Detects trough of the period
if (input < movingAverage && (trough == 0)){
peak = 0;
trough = 1;

}
}

}
}

// runningAverage for heart rate counting
float runningAverage(float M) {
```

```
// keep sum updated to improve speed.
sum -= LM[index];
LM[index] = M;
sum += LM[index];

//println("Sum = " + sum);

index++;
index = index % LM_SIZE;
if (count < LM_SIZE) count++;

return (sum / count);
}
```

A.2.4 Arduino Code for Thermistor

```
/*
AnalogReadSerial
Reads an analog input on pin 0, prints the result to the serial monitor.
Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V
    and ground.

This example code is in the public domain.
*/

#include <math.h>
#include <stdlib.h>
#include <stdio.h>

double T;
double TK;
double lnratio;
double a, b, c, d;

// the setup routine runs once when you press reset:
void setup() {
```

```
        // initialize serial communication at 9600 bits per second:
        Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
        // read the input on analog pin 0:
        double sensorValue = analogRead(A0);
        // print out the value you read:
        a = 3.3540154e-3;
        b = 2.5627725*pow(10,-4);
        c = 2.0829210*pow(10,-6);
        d = 7.3003206*pow(10,-8);

        // lnratio = log(sensorValue/(1023-sensorValue));
        for (double i=1;i<1024;i++) {
                lnratio = log(i/(1024-i));
                T = 1/(a+b*(lnratio)+c*pow(lnratio,2)+d*pow(lnratio,3));
                TK = T-273.15;

                Serial.println(TK);
                //Serial.print(' ');
                //Serial.println(sensorValue);

                delay(1);       // delay in between reads for stability
                }
}
```

A.2.5 Arduino Code for Phase Shift Oscillator

```
#include <math.h>
double T = 0;   /*Temperature*/
double t = 0;   /*Period*/
double F = 0;   /*Frequency*/

void setup() {
```

```
        /* Define pin 3 as an input */
        pinMode(3,INPUT);
        Serial.begin(9600);
}

void loop() {

        while(digitalRead(3) == HIGH) {
        };
        while(digitalRead(3) == LOW) {
        };
        t = micros();
        while(digitalRead(3) == HIGH) {
        };
        while(digitalRead(3) == LOW) {
        };

        t = micros() - t;
        F = 1.0E6/t;
        T = 102.58*log(F) - 659.57;

        Serial.println(T);
        delay(1);
}
```

A.2.6 MATLAB ECG Simulation

The simulated ECG signal in Figure 7.1 in Section 7 was obtained by running the following code in MATLAB.

```
>>ECGwaveGen(70,5,500,2500)
```

The ECGwaveGen.m file below was obtained from physionet.org [53].

```
function [QRSwave]=ECGwaveGen(bpm,duration,fs,amp)
%[QRSwave]=ECGwaveGen(bpm,dur,fs,amp) generates an artificial ECG/EKG waveform
%    Heart rate (bpm) sets the qrs event frequency (RR interval).
%    Duration of the entire waveform (dur) is in units of seconds.
```

```
%    Sample frequency (fs) sets the sample frequency in Hertz.
%    Amplitude (amp) of the QRS event is measured in micro Volts. The
%    waveform consists of a QRS complex and a T-wave. No attempt to
%    represent a P-wave has been made.
%
%    There are two additional parameters that can be changed from within the
   function.
%    They are the parameters that set the QRS width (default 0.1 secs) and the t
   -wave
%    amplitude (default 500 uV).

%Created January 22, 2001 by Floyd Harriott, primary email (fharriott@stellate.
   com), secondary email (fsh@po.cwru.edu)
%Modified March 19, 2002 by Floyd Harriott, extended default duration so that
   default settings produce a QRS event rather than
%   an error. Allows for the random insertion of PVCs.  This file must be edited
    to include PVCs.

%Algorithm is based in part on the jounal article:
%Ruha, Antti and Seppo Nissila, "A Real-Time Microprocessor QRS Detector System
   with a 1-ms Timing Accuracy
%      for the Measurement of Ambulatory HRV", IEEE Trans. Biomed. Eng. Vol. 44,
    No. 3, 1997
%The artificial ECG signal they describe is based on the recommendations in the
   Association for the Advancement
%of Medical Instrumentation (AAMI) "Standard for Cardiac Monitors, Heart Rate
   Meters and Alarms (draft), Aug. 1981
%Feel free to make modifications, corrections and or suggestions.

if (exist('fs') ~= 1)  fs=  200;   end  %default value, Hz
if (exist('bpm') ~= 1)  bpm =  72;   end %default value, beats per minute
if (exist('amp') ~= 1)  amp = 1000;   end %default value, micro volts
if (exist('duration') ~= 1)  duration =  (60/bpm-0.35)+60/bpm+1/fs; end  %
   default value gives one cycle, seconds

global t_line; %seconds
global sample_freq; % always equal to fs
```

```
%Changeable Parameters
d=0.1; %.07 to .120 seconds, QRS width
at=500; %amplitude of t-wave, 400 to 1200 uv

%Should not touch
org_amp=amp;
sample_freq=fs; %duplicated simply to make a global version
RR=(60/bpm); %RR interval, seconds
d1=0.4375*d;
d2=0.5*d;
d3=d-(d1+d2);
dt=0.180; %width of t wave, seconds
qt=0.35; %time from beginning of QRS to end of t-wave
t_line=0:1/fs:duration; %time line, seconds
QRS_wave=zeros( size(t_line) ); %QRS waveform
deadspace=RR-qt; %time between t-wave and next QRS
if deadspace < 0
err_msg=['Bpm must be equal to or less than ' int2str(60/qt) ' inorder to fit
    one cycle.'];
error(err_msg);
end


%Calculate PVC parameters and segment
PVCchance=0.1; %How often does PVC happen., percent eg. 0.1=10%
PVCamp=amp;    %PVC amplitude, eg. same as normals (amp)
earlyfactor=0.25; %percentage, how much early should PVC happen then normal RR
    interval
PVCwidth=0.12; %seconds, QRS width of PVC, usually .12 to .17
PVCseg=[QRSpulse(d,60/((1-earlyfactor)*RR-0.4375*PVCwidth),fs,RandAmp(org_amp))
    QRSpulse(PVCwidth,bpm*(1-earlyfactor),fs,PVCamp) QRSpulse(d,bpm,fs, RandAmp(
    org_amp))]; %PVC segment
tPVC=size(PVCseg,2)/fs; %amount of time taken up by PVC segment in seconds

t1=deadspace; %Where does the first QRS start? eg deadspace, or 0

%need enough time to display at least one interval.
```

```
if (t1+60/bpm+1/sample_freq > duration)
err_msg=['The␣waveform␣length␣(duration)␣must␣be␣more␣than␣' sprintf('%.2f%',t1
    +60/bpm+1/sample_freq) '␣second(s)␣in␣order␣to␣display␣one␣QRS␣event.'];
error(err_msg);
end


%GENERATION LOOP
while ( t1+60/bpm+1/sample_freq <= duration) %space to insert another qrs pulse
    in time line

%amp=RandAmp(org_amp); %random size on qrs event
amp=org_amp;



%Segment 1  (Q-R)
qrs_start=t1;
t2=t1+d1;
i_t1=time2index(t1); i_t2=time2index(t2);
left=0; right=0.875*amp;
m1=(right-left)/(t2-t1);
QRS1=m1*index2time(i_t1:i_t2)-(m1*t1-left);
QRSwave(i_t1:i_t2)=QRS1;

%Segment 2  (R-?)
t1=t2; t2=t1+d2;
i_t1=time2index(t1); i_t2=time2index(t2);
left=right; right=-.125*amp;
m2=(right-left)/(t2-t1);
QRS1=m2*index2time(i_t1:i_t2)-(m2*t1-left);
QRSwave(i_t1:i_t2)=QRS1;


%Segment 3 bottom_top (?-S)
t1=t2; t2=t1+d3;
i_t1=time2index(t1); i_t2=time2index(t2);
left=right; right=0;
```

```
if (i_t2-i_t1 >0) %at low sampling freq. there may be no sample for this segment
m3=(right-left)/(t2-t1);
QRS1=m3*index2time(i_t1:i_t2)-(m3*t1-left);
QRS1=QRS1( find(QRS1<=0));
QRSwave(i_t1:i_t1+size(QRS1,2)-1)=QRS1;
elseif i_t2-i_t1==0
m3=(right-left)/(t2-t1);
QRS1=m3*index2time(i_t1:i_t2)-(m3*t1-left);
QRSwave(i_t1)=QRS1(1);
end

%Segment 4, S-T interval
t1=t2; t2=t1+qt+qrs_start-(dt+t2);
i_t1=time2index(t1); i_t2=time2index(t2);
left=right; right=0;

%Segment 5, t-wave
t1=t2; t2=t1+dt;
i_t1=time2index(t1); i_t2=time2index(t2);
t=-1:2/(i_t2-i_t1):1;
QRS1=at*sqrt(1-t.^2);
QRSwave(i_t1:i_t2)=QRS1;

%Segment 6, remaining deadspace
t1=t2; t2=t1+deadspace;
i_t1=time2index(t1); i_t2=time2index(t2);



%Do we insert a PVC here? Roll the die and find out.
insertPVC=rand(1); %uncomment following 5 lines if PVCs are desired.
%if insertPVC<=PVCchance & t2+tPVC+2/sample_freq <= duration %enough space to
    insert PVC
%   t1=t2; t2=t1+tPVC;
%  i_t1=time2index(t1); i_t2=time2index(t2);
% QRSwave(i_t1:i_t1+size(PVCseg,2)-1)=PVCseg;
%end

%stem(QRSwave); % view ECG waveform
```

```
t1=t2; %end of this segment becomes beginning of next segment

end %while loop, appending qrs pulses

%_____________________________________%
function index=time2index(t)
%TIME2INDEX converts time (s) to an index value

global t_line;

indexArray=find(t_line>=t);
index=indexArray(1);

%_____________________________________%
function time=index2time(i)
%INDEX2TIME converts a time line index to a time value (seconds)
global sample_freq

time=(i-1).*1/sample_freq;



%_____________________________________%
function RAmp=RandAmp(orgAmp)

RAmp=orgAmp+0.4*orgAmp*rand(1);
```

Appendix B

PCB Design and Schematics

SEPARATE DESIGN AND SCHEMATICS

B.1 ECG Shield for AD8232 and Arduino Pro Mini

B.2 Blood Temperature

B.3 AD8232 Heart Rate Monitor SparkFun Implementation

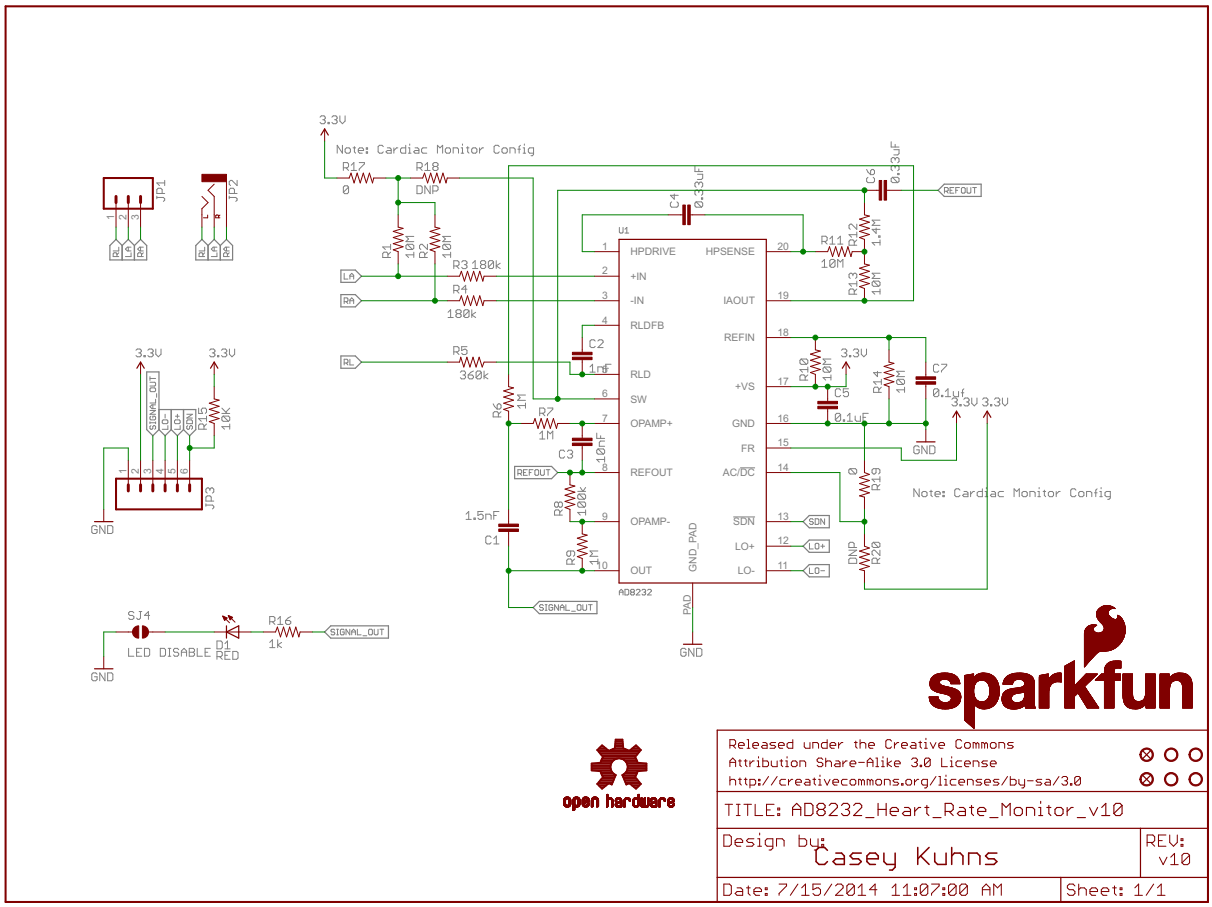

Figure B.1: AD8232 SparkFun Implementation Schematic Diagram [21]

Further details of the design can be found at
https://github.com/sparkfun/AD8232_Heart_Rate_Monitor.

B.4 Raspberry Pi 3 Model B

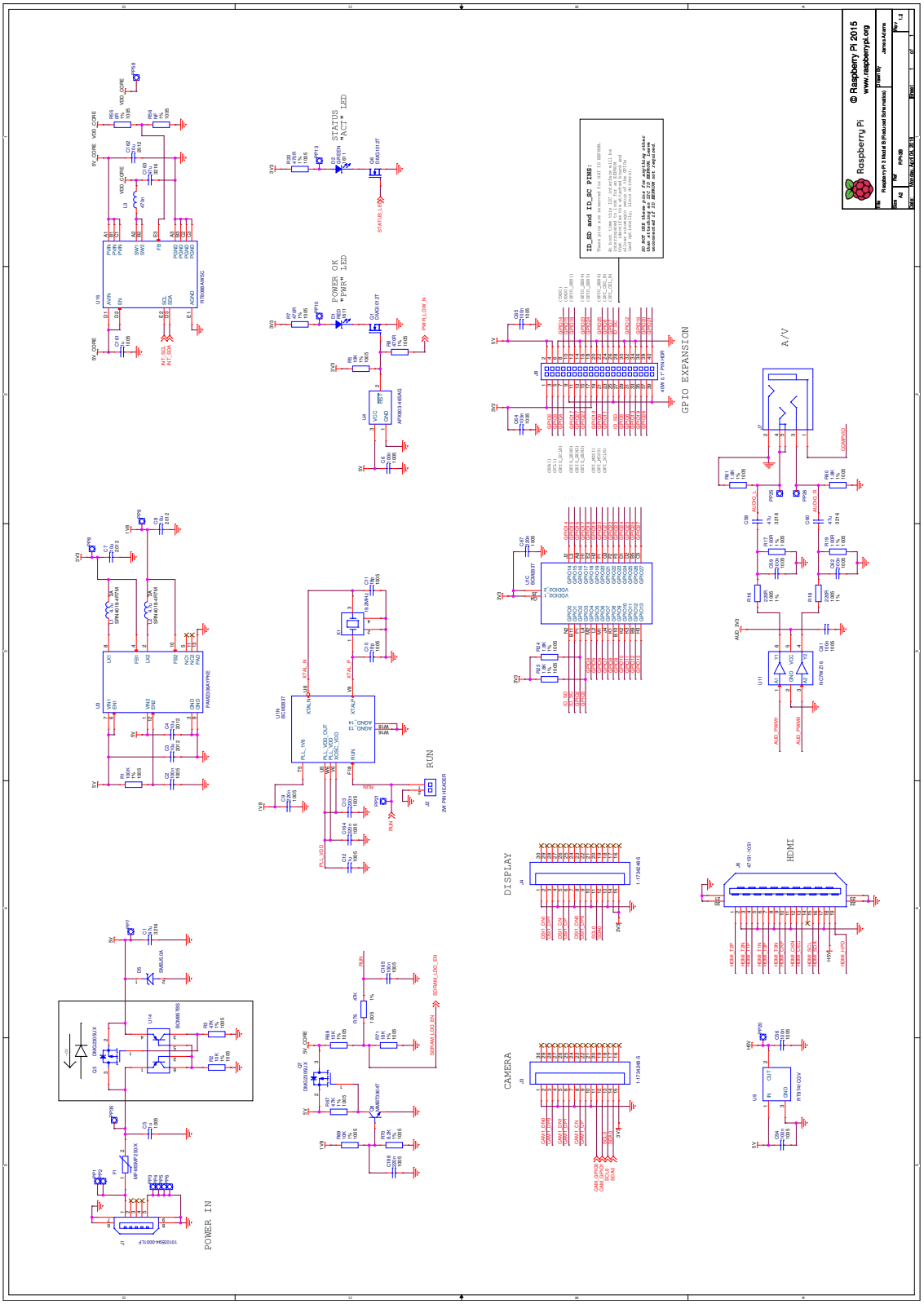

Figure B.2: Raspberry Pi 3 Model B Schematic Diagram [28]

Further details of the design can be found at
https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/RPI-3B-V1_2-SCHEMATIC-REDUCED.pdf.

B.5 Transmitter Schematics for PPG

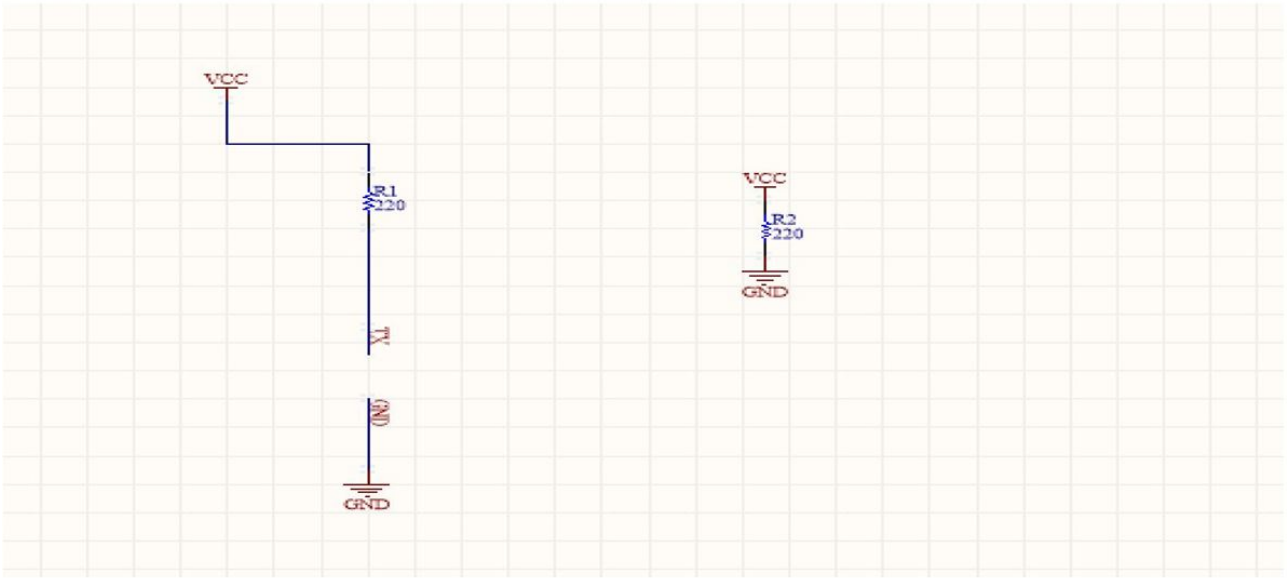

Figure B.3: Transmitter Schematics for PPG

B.6 Transmitter PCB for PPG

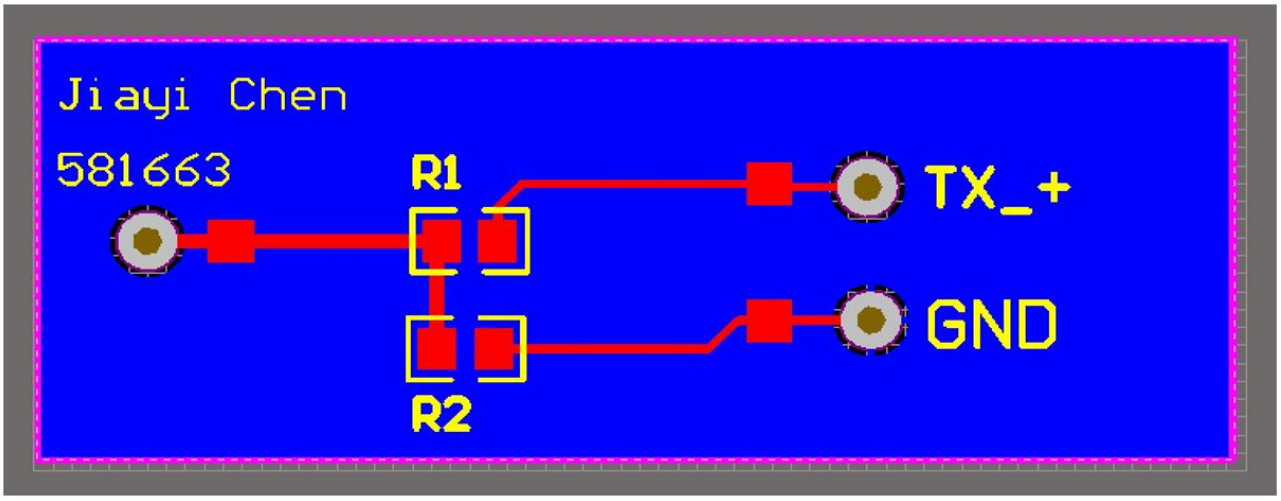

Figure B.4: Transmitter PCB for PPG

B.7 Receiver Schematics for PPG

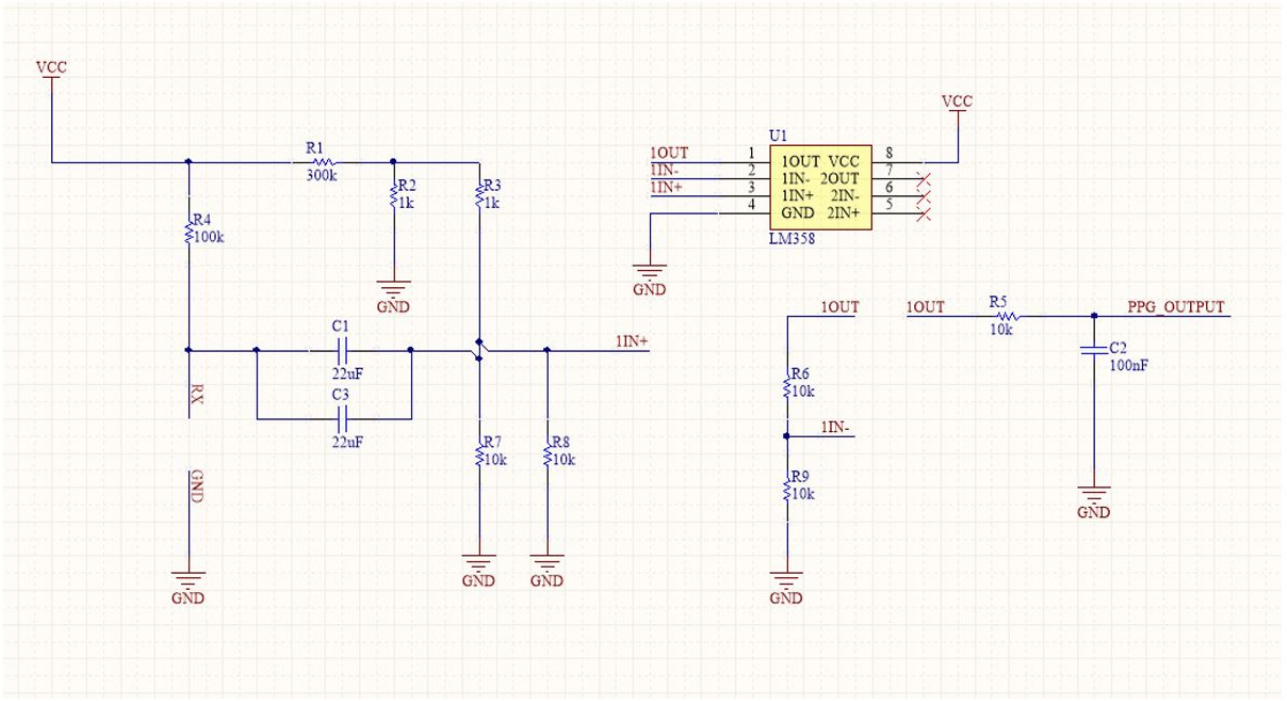

Figure B.5: Receiver Schematics for PPG

B.8 Receiver PCB for PPG

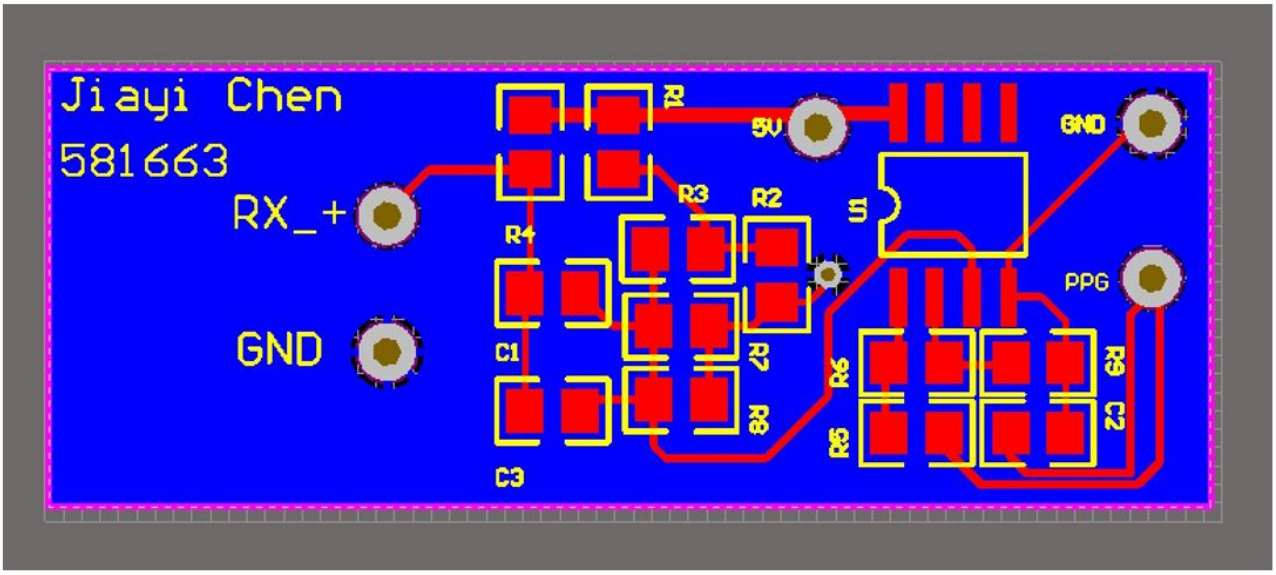

Figure B.6: Receiver PCB for PPG

Appendix C

Measurements and Datasheets

C.1 Datasheet

Data for material type : F

Temp Range (°C)	Ratio	Beta
0 to 50	9.08	3895
0 to 70	18.64	3917
25 to 50	2.78	3933
25 to 85	9.30	3969
25 to 100	14.64	3981
25 to 125	29.05	3999
37.8 to 104.4	9.67	4000

To calculate Rt/R25 at temperatures other than those listed in the table, use the following equation:
$Rt/R25 = \exp\{A + B/T + C/T^2 + D/T^3\}$
where T = temperature in K
where K = °C + 273.15

Temp Range (°C)	A	B	C	D
-50 to 0	-1.4122478E+01	4.4136033E+03	-2.9034189E+04	-9.3875035E+06
0 to 50	-1.4141963E+01	4.4307830E+03	-3.4078983E+04	-8.8941929E+06
50 to 100	-1.4202172E+01	4.4975256E+03	-5.8421357E+04	-5.9658796E+06
100 to 150	-1.6154078E+01	6.8483992E+03	-1.0004049E+06	1.1961431E+08

To calculate the actual thermistor temperature as a function of the thermistor resistance, use the following equation:
$1/T = a + b(\text{Ln } Rt/R25) + c(\text{Ln } Rt/R25)^2 + d(\text{Ln } Rt/R25)^3$

Rt/R25 range	a	b	c	d
68.600 to 3.274	3.3538646E-03	2.5654090E-04	1.9243889E-06	1.0969244E-07
3.274 to 0.36036	3.3540154E-03	2.5627725E-04	2.0829210E-06	7.3003206E-08
0.36036 to 0.06831	3.3539264E-03	2.5609446E-04	1.9621987E-06	4.6045930E-08
0.06831 to 0.01872	3.3368620E-03	2.4057263E-04	-2.6687093E-06	-4.0719355E-07

Temperature (°C)	Rt/R25 nominal	Temp Coef α (%/°C)	β Deviation† (±%)
-50	68.60	7.21%	2.30%
-45	48.16	6.96%	2.68%
-40	34.23	6.71%	2.87%
-35	24.62	6.48%	2.92%
-30	17.91	6.26%	2.86%
-25	13.17	6.05%	2.71%
-20	9.782	5.85%	2.50%
-15	7.339	5.66%	2.25%
-10	5.558	5.47%	1.97%
-5	4.247	5.30%	1.68%
0	3.274	5.13%	1.37%
5	2.544	4.97%	1.07%
10	1.992	4.81%	0.78%
15	1.572	4.67%	0.50%
20	1.250	4.53%	0.24%
25	1.000	4.39%	0.00%
30	0.8056	4.26%	0.21%
35	0.6530	4.14%	0.40%
40	0.5326	4.02%	0.56%
45	0.4369	3.91%	0.69%
50	0.3604	3.80%	0.80%
55	0.2989	3.69%	0.87%
60	0.2491	3.59%	0.92%
65	0.2087	3.49%	0.93%
70	0.1756	3.40%	0.92%
75	0.1485	3.31%	0.88%
80	0.1261	3.23%	0.81%
85	0.1075	3.14%	0.72%
90	0.09209	3.06%	0.59%
95	0.07916	2.99%	0.45%
100	0.06831	2.91%	0.28%
105	0.05916	2.85%	0.08%
110	0.05141	2.77%	0.12%
115	0.04483	2.70%	0.36%
120	0.03922	2.64%	0.61%
125	0.03442	2.57%	0.87%
130	0.03030	2.51%	1.16%
135	0.02675	2.47%	1.46%
140	0.02369	2.41%	1.82%
145	0.02103	2.35%	2.14%
150	0.01872	2.35%	2.46%

By plotting Equation 5.3 throughout the temperature range, we obtain Figure C.2.

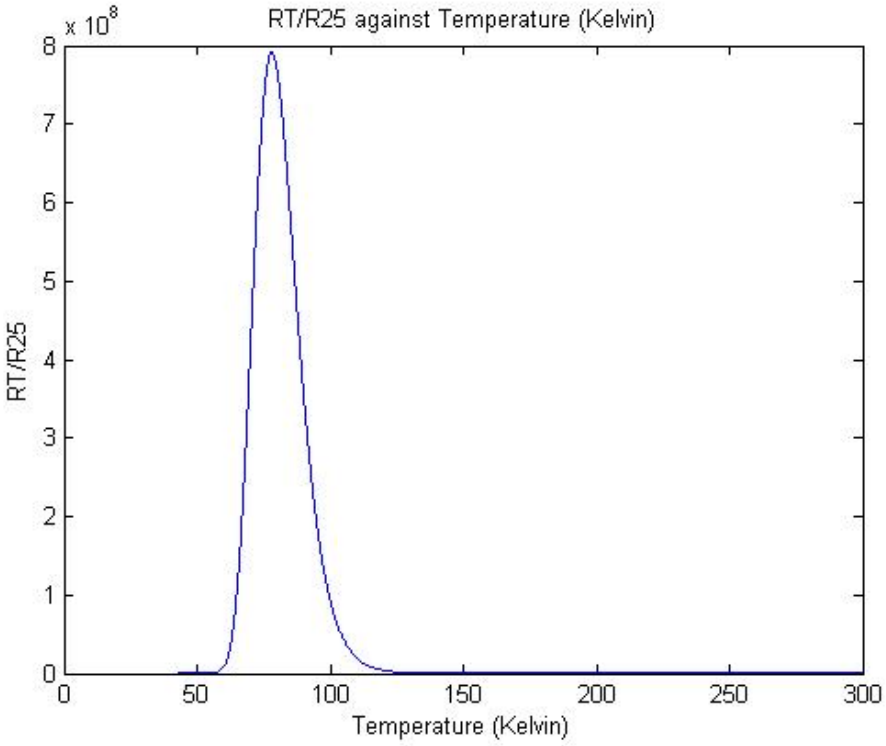

Figure C.2: RT/R25 against Temperature (Kelvin)

C.2 Phase Shift Oscillator Calibration

For the calibration of the phase shift oscillator frequency for specific resistances of the thermistor, the output frequency measured by the Arduino Uno is recorded as found in Tables C.1, C.2, and C.3 below for each resistor value chosen. This process is repeated 100 times for each resistor to find the average output frequency corresponding to that resistance of R_1.

16 k ohm	10k ohm	12k ohm	23k ohm	9k ohm	25k ohm	6.7k ohm	22k ohm	13k ohm
714.29	791.14	757.58	666.67	811.69	656.17	868.06	672.04	744.05
712.25	793.65	757.58	670.24	809.06	659.63	874.13	664.89	753.01
716.33	791.14	759.88	670.24	809.06	656.17	856.16	664.89	744.05
714.29	791.14	762.2	668.45	811.69	656.17	862.07	672.04	744.05
712.25	796.18	759.88	670.24	806.45	657.89	868.06	672.04	744.05

714.29	791.14	759.88	668.45	809.06	659.63	868.06	672.04	753.01
712.25	793.65	757.58	666.67	811.69	659.63	868.06	664.89	744.05
716.33	793.65	762.2	670.24	811.69	656.17	868.06	672.04	744.05
716.33	793.65	759.88	670.24	811.69	657.89	868.06	672.04	744.05
714.29	791.14	762.2	668.45	809.06	656.17	868.06	672.04	744.05
716.33	793.65	757.58	670.24	811.69	657.89	868.06	672.04	744.05
712.25	791.14	759.88	668.45	816.99	657.89	874.13	664.89	753.01
714.29	793.65	757.58	668.45	811.69	656.17	874.13	672.04	744.05
714.29	793.65	757.58	668.45	811.69	657.89	868.06	664.89	744.05
714.29	791.14	757.58	668.45	811.69	657.89	868.06	672.04	753.01
714.29	793.65	757.58	666.67	809.06	656.17	868.06	672.04	744.05
714.29	793.65	757.58	666.67	811.69	659.63	868.06	664.89	744.05
716.33	793.65	762.2	666.67	811.69	657.89	868.06	664.89	744.05
714.29	793.65	757.58	666.67	811.69	656.17	868.06	664.89	753.01
714.29	793.65	759.88	668.45	811.69	656.17	868.06	672.04	753.01
714.29	793.65	759.88	670.24	811.69	657.89	868.06	664.89	753.01
712.25	793.65	757.58	668.45	811.69	657.89	874.13	664.89	753.01
716.33	793.65	759.88	670.24	811.69	656.17	862.07	672.04	753.01
714.29	793.65	759.88	670.24	811.69	656.17	856.16	672.04	744.05
712.25	793.65	757.58	668.45	811.69	657.89	868.06	672.04	744.05
714.29	791.14	757.58	670.24	811.69	657.89	868.06	664.89	744.05
714.29	791.14	759.88	670.24	811.69	659.63	868.06	672.04	744.05
712.25	791.14	759.88	668.45	809.06	656.17	868.06	672.04	744.05
712.25	793.65	759.88	670.24	811.69	657.89	874.13	664.89	744.05
712.25	793.65	759.88	670.24	811.69	657.89	868.06	664.89	744.05
712.25	793.65	759.88	668.45	806.45	657.89	868.06	664.89	753.01
714.29	796.18	759.88	670.24	811.69	657.89	868.06	672.04	744.05
712.25	793.65	762.2	668.45	811.69	656.17	874.13	664.89	753.01
714.29	791.14	759.88	668.45	809.06	659.63	868.06	664.89	744.05
714.29	791.14	757.58	666.67	811.69	656.17	868.06	664.89	744.05

714.29	793.65	759.88	668.45	811.69	656.17	868.06	672.04	744.05
716.33	793.65	759.88	670.24	811.69	659.63	868.06	672.04	753.01
712.25	793.65	759.88	668.45	811.69	657.89	868.06	664.89	744.05
712.25	791.14	757.58	670.24	811.69	659.63	868.06	664.89	744.05
712.25	793.65	757.58	668.45	809.06	656.17	868.06	672.04	753.01
716.33	793.65	759.88	670.24	809.06	657.89	868.06	664.89	744.05
712.25	796.18	757.58	670.24	809.06	659.63	868.06	672.04	744.05
712.25	793.65	757.58	668.45	811.69	657.89	868.06	672.04	744.05
716.33	793.65	757.58	670.24	809.06	656.17	868.06	664.89	744.05
712.25	791.14	759.88	670.24	803.86	657.89	868.06	664.89	744.05
714.29	793.65	762.2	666.67	811.69	659.63	868.06	664.89	753.01
712.25	791.14	759.88	666.67	809.06	657.89	868.06	664.89	744.05
714.29	793.65	757.58	668.45	811.69	656.17	874.13	664.89	744.05
716.33	793.65	757.58	668.45	811.69	657.89	868.06	664.89	744.05
712.25	793.65	762.2	668.45	806.45	654.45	868.06	664.89	744.05
712.25	793.65	759.88	668.45	811.69	657.89	868.06	672.04	744.05
714.29	793.65	757.58	668.45	811.69	657.89	874.13	672.04	744.05
712.25	793.65	757.58	670.24	811.69	657.89	868.06	672.04	744.05
714.29	793.65	759.88	666.67	811.69	657.89	868.06	672.04	753.01
714.29	793.65	757.58	668.45	811.69	659.63	868.06	672.04	744.05
712.25	793.65	757.58	668.45	811.69	659.63	868.06	664.89	744.05
714.29	793.65	757.58	670.24	809.06	657.89	868.06	672.04	753.01
712.25	796.18	759.88	668.45	811.69	657.89	868.06	664.89	744.05
712.25	793.65	759.88	668.45	811.69	657.89	868.06	672.04	753.01
714.29	796.18	757.58	668.45	809.06	656.17	868.06	664.89	744.05
712.25	793.65	757.58	668.45	809.06	656.17	868.06	672.04	753.01
712.25	793.65	762.2	668.45	811.69	656.17	868.06	672.04	744.05
714.29	793.65	757.58	670.24	811.69	656.17	868.06	664.89	744.05
714.29	793.65	762.2	670.24	814.33	659.63	868.06	664.89	753.01
716.33	793.65	762.2	670.24	809.06	659.63	868.06	664.89	744.05

714.29	793.65	759.88	670.24	809.06	659.63	868.06	664.89	744.05
714.29	793.65	759.88	668.45	811.69	656.17	868.06	672.04	744.05
714.29	796.18	759.88	670.24	811.69	656.17	868.06	672.04	744.05
712.25	793.65	757.58	666.67	809.06	657.89	868.06	664.89	753.01
712.25	791.14	762.2	666.67	811.69	657.89	868.06	672.04	744.05
714.29	793.65	757.58	670.24	809.06	656.17	868.06	664.89	744.05
712.25	793.65	759.88	666.67	811.69	657.89	868.06	664.89	744.05
714.29	793.65	759.88	670.24	809.06	661.38	868.06	664.89	744.05
714.29	793.65	759.88	668.45	809.06	657.89	868.06	664.89	744.05
714.29	793.65	759.88	670.24	811.69	659.63	868.06	664.89	744.05
712.25	793.65	759.88	666.67	811.69	657.89	868.06	664.89	744.05
712.25	793.65	759.88	668.45	811.69	661.38	868.06	664.89	744.05
712.25	793.65	757.58	666.67	811.69	656.17	868.06	664.89	744.05
710.23	791.14	759.88	668.45	809.06	661.38	868.06	664.89	744.05
714.29	793.65	762.2	666.67	811.69	659.63	868.06	664.89	744.05
714.29	788.64	759.88	668.45	811.69	661.38	868.06	664.89	744.05
712.25	793.65	759.88	668.45	809.06	657.89	868.06	664.89	753.01
712.25	793.65	757.58	668.45	811.69	661.38	868.06	672.04	753.01
712.25	793.65	757.58	670.24	811.69	659.63	868.06	672.04	753.01
714.29	793.65	759.88	670.24	811.69	657.89	862.07	664.89	744.05
714.29	793.65	757.58	672.04	809.06	657.89	868.06	672.04	744.05
712.25	791.14	759.88	668.45	811.69	659.63	862.07	672.04	744.05
714.29	793.65	759.88	666.67	811.69	659.63	874.13	664.89	744.05
716.33	791.14	764.53	670.24	809.06	659.63	868.06	664.89	744.05
716.33	791.14	762.2	668.45	814.33	656.17	868.06	672.04	744.05
714.29	791.14	759.88	666.67	806.45	661.38	868.06	672.04	744.05
712.25	793.65	759.88	668.45	811.69	656.17	868.06	672.04	744.05
716.33	793.65	757.58	670.24	811.69	659.63	868.06	664.89	744.05
714.29	793.65	759.88	670.24	811.69	657.89	868.06	664.89	744.05
714.29	793.65	764.53	666.67	806.45	657.89	868.06	664.89	744.05

716.33	793.65	762.2	670.24	811.69	656.17	868.06	672.04	744.05
718.39	791.14	762.2	666.67	811.69	659.63	868.06	664.89	744.05
714.29	793.65	762.2	670.24	811.69	657.89	868.06	664.89	753.01
714.29	793.65	759.88	670.24	811.69	657.89	868.06	664.89	744.05
714.29	793.65	757.58	668.45	811.69	656.17	868.06	664.89	753.01

Table C.1: Output Frequency based on R1 Values

3.6k ohm	16k ohm	18k ohm	6.2k ohm	9k ohm	7.5k ohm	18k ohm	18k ohm	22k ohm
1041.67	710.23	694.44	868.06	809.06	850.34	694.44	694.44	670.24
1008.06	710.23	694.44	919.12	806.45	844.59	694.44	694.44	666.67
1008.06	710.23	694.44	919.12	811.69	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	811.69	844.59	702.25	694.44	666.67
1008.06	726.74	694.44	919.12	809.06	844.59	694.44	694.44	670.24
1008.06	710.23	679.35	919.12	811.69	850.34	694.44	694.44	668.45
1041.67	710.23	694.44	868.06	809.06	844.59	694.44	694.44	670.24
976.56	710.23	694.44	868.06	811.69	850.34	694.44	702.25	670.24
1008.06	710.23	694.44	919.12	809.06	850.34	694.44	702.25	670.24
1008.06	710.23	694.44	919.12	806.45	844.59	694.44	694.44	668.45
1008.06	726.74	694.44	868.06	811.69	856.16	694.44	694.44	666.67
1008.06	710.23	694.44	868.06	811.69	850.34	702.25	694.44	666.67
1008.06	726.74	710.23	919.12	809.06	856.16	694.44	694.44	668.45
1008.06	726.74	694.44	868.06	811.69	850.34	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	809.06	850.34	694.44	694.44	666.67
1041.67	726.74	694.44	919.12	806.45	850.34	694.44	702.25	668.45
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	686.81	668.45
1008.06	710.23	694.44	868.06	809.06	862.07	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	811.69	850.34	702.25	702.25	668.45
1008.06	710.23	710.23	868.06	811.69	850.34	702.25	694.44	668.45

1008.06	710.23	694.44	919.12	809.06	844.59	694.44	694.44	668.45
1041.67	710.23	694.44	868.06	814.33	850.34	694.44	694.44	666.67
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	806.45	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	919.12	811.69	844.59	694.44	702.25	668.45
1008.06	710.23	694.44	868.06	811.69	844.59	702.25	694.44	666.67
1008.06	710.23	694.44	868.06	811.69	850.34	694.44	694.44	666.67
1008.06	710.23	694.44	919.12	811.69	844.59	702.25	694.44	668.45
1008.06	710.23	679.35	919.12	809.06	844.59	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	694.44	668.45
976.56	710.23	694.44	868.06	809.06	844.59	694.44	694.44	670.24
1008.06	710.23	694.44	919.12	811.69	850.34	694.44	694.44	666.67
1008.06	726.74	694.44	868.06	809.06	844.59	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	814.33	850.34	694.44	694.44	666.67
1008.06	710.23	694.44	919.12	811.69	850.34	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	809.06	850.34	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	670.24
1008.06	710.23	710.23	868.06	806.45	856.16	694.44	694.44	666.67
1008.06	710.23	694.44	868.06	809.06	850.34	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	809.06	856.16	694.44	686.81	668.45
1008.06	710.23	694.44	919.12	811.69	850.34	694.44	694.44	670.24
1008.06	710.23	694.44	919.12	811.69	850.34	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	806.45	850.34	702.25	694.44	668.45
1008.06	710.23	679.35	868.06	811.69	850.34	694.44	694.44	668.45
1008.06	710.23	694.44	919.12	806.45	856.16	702.25	686.81	668.45
1008.06	710.23	694.44	919.12	811.69	850.34	694.44	694.44	668.45
976.56	710.23	694.44	868.06	811.69	850.34	694.44	694.44	668.45
1041.67	710.23	694.44	919.12	809.06	850.34	694.44	694.44	666.67
1008.06	710.23	694.44	919.12	811.69	844.59	694.44	694.44	666.67
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	668.45

1008.06	710.23	694.44	868.06	811.69	850.34	702.25	694.44	666.67
1008.06	726.74	694.44	868.06	809.06	844.59	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	666.67
976.56	710.23	694.44	868.06	811.69	844.59	694.44	694.44	666.67
1041.67	710.23	694.44	919.12	809.06	850.34	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	806.45	850.34	694.44	694.44	666.67
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	809.06	850.34	694.44	694.44	666.67
1008.06	726.74	694.44	919.12	809.06	850.34	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	694.44	670.24
1008.06	710.23	694.44	919.12	811.69	850.34	694.44	694.44	666.67
976.56	726.74	694.44	919.12	806.45	850.34	694.44	694.44	670.24
976.56	710.23	694.44	868.06	809.06	850.34	702.25	694.44	670.24
1041.67	710.23	694.44	868.06	809.06	844.59	702.25	694.44	668.45
1008.06	710.23	694.44	919.12	811.69	844.59	694.44	694.44	666.67
1008.06	710.23	694.44	919.12	809.06	850.34	702.25	694.44	670.24
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	811.69	850.34	702.25	694.44	670.24
1008.06	710.23	694.44	919.12	809.06	850.34	694.44	694.44	666.67
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	666.67
976.56	726.74	694.44	868.06	811.69	844.59	694.44	694.44	666.67
1008.06	710.23	694.44	919.12	809.06	850.34	694.44	694.44	670.24
1008.06	726.74	694.44	868.06	811.69	844.59	694.44	694.44	666.67
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	666.67
1008.06	710.23	694.44	919.12	809.06	844.59	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	702.25	668.45
1041.67	726.74	694.44	919.12	811.69	850.34	702.25	694.44	666.67
1008.06	710.23	694.44	868.06	806.45	850.34	694.44	694.44	670.24

1008.06	726.74	694.44	868.06	811.69	844.59	694.44	702.25	666.67
1008.06	710.23	694.44	919.12	811.69	850.34	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	809.06	850.34	694.44	694.44	668.45
1008.06	710.23	710.23	868.06	811.69	850.34	694.44	694.44	666.67
1008.06	710.23	679.35	919.12	811.69	844.59	702.25	702.25	670.24
1008.06	710.23	694.44	868.06	809.06	850.34	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	811.69	844.59	694.44	694.44	666.67
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	694.44	668.45
1041.67	710.23	694.44	919.12	811.69	844.59	694.44	694.44	668.45
976.56	710.23	694.44	868.06	811.69	844.59	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	809.06	850.34	694.44	694.44	670.24
1008.06	710.23	694.44	919.12	806.45	844.59	694.44	694.44	672.04
1008.06	710.23	679.35	868.06	811.69	844.59	694.44	694.44	668.45
1008.06	710.23	710.23	868.06	811.69	850.34	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	806.45	850.34	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	809.06	844.59	694.44	694.44	670.24
1008.06	710.23	694.44	868.06	809.06	850.34	694.44	694.44	668.45
1008.06	710.23	694.44	868.06	809.06	850.34	702.25	694.44	672.04
1008.06	710.23	694.44	868.06	811.69	850.34	694.44	694.44	670.24

Table C.2: Output Frequency based on R1 Values (cont.)

3.9k ohm			
984.25	976.56	984.25	976.56
984.25	984.25	976.56	992.06
992.06	976.56	984.25	984.25
976.56	984.25	992.06	984.25
992.06	992.06	984.25	976.56
984.25	992.06	976.56	984.25
984.25	984.25	984.25	992.06
976.56	976.56	984.25	984.25
992.06	992.06	984.25	984.25
992.06	984.25	992.06	976.56
992.06	992.06	976.56	992.06
984.25	984.25	984.25	984.25
976.56	992.06	984.25	984.25
992.06	984.25	984.25	976.56
976.56	976.56	976.56	992.06
992.06	992.06	992.06	984.25
976.56	992.06	992.06	984.25
992.06	992.06	984.25	984.25
976.56	984.25	976.56	976.56
984.25	976.56	984.25	992.06
984.25	984.25	984.25	976.56
976.56	984.25	984.25	992.06
992.06	984.25	984.25	984.25
984.25	984.25	984.25	984.25
992.06	984.25	992.06	984.25

Table C.3: Output Frequency based on R1 Values (cont.)