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

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

1 Introduction and Background **1**

1.1 The Spaghetti Syndrome and its Implications - the need for a Wireless Vital Signs Monitoring System 1

1.2 Benefits of a Wireless Implementation 2

1.3 Overview of the Project 2

1.4 Anaesthetic Parameters for Patient Monitoring 3

1.4.1 Theory of Electrocardiography 3

1.4.2 Theory of Electroencephalography 6

1.4.3 Theory of Photoplethysmography 6

1.4.4 Theory of Blood Pressure 6

1.4.5 Theory of Blood Temperature 6

2 Survey of Related Previous Work - Overview of Wireless Sensor Network Technology **7**

2.1 Wireless Sensor Networks 7

2.2 Body Area Networks 7

2.3 Methods of Wireless Transmission 7

2.3.1 Comparison between Virtual Nerwork Computing (VNC) and Raw Transmission 7

2.3.2 Virtual Network Computing 9

2.4 WiFi 9

2.5 Bluetooth . . . 9
2.6 Zigbee . . . 9
2.7 Comparison of Different Protocols . . . 9

3 Project Design and Development **10**

3.1 System Overview . . . 10
3.2 Design Specifications . . . 11
3.3 Hardware Design and Development . . . 11
3.4 Software Design and Development . . . 11
3.5 Analysis of Competitors - Similar Papers - Literature Review . . . 11
3.6 Analysis of Existing Systems . . . 11
3.6.1 Consideration of Bionomadix . . . 11
3.6.2 Consideration of Libelium and Waspmote . . . 11
3.6.3 Consideration of Cooking Hacks . . . 11

4 Terminals for Data Processing **12**

4.1 Operating System . . . 12
4.2 Raspberry Pi 3 Model B . . . 14
4.2.1 Raspbian Jessie . . . 16
4.2.2 Portable Power Supply for the Raspberry Pi 3 Model B . . . 16
4.3 Intel Nuc . . . 16

5 Sensors **17**

5.1 ECG . . . 17
5.1.1 AD8232 IC ECG Sensor . . . 18
5.2 Blood Temperature . . . 24
5.2.1 Thermistor . . . 24
5.2.2 Self-Heating - Phase Shift Oscillator . . . 26

5.2.3 Astable Operation of 555 Timer . . . 26
5.3 Interface . . . 27
5.3.1 USB . . . 27
5.3.2 Arduino Uno / Arduino Pro Mini - Analog to Digital Conversion . . . 27

6 Data Processing, Visualisation, and Transmission 28
6.1 Processing 2.2.1 . . . 30
6.2 Serial Input . . . 30
6.3 Code . . . 30
6.4 Visualisation . . . 30
6.5 Wireless Transmission . . . 30

7 Simulation 31

8 Testing 32

9 Results 33
9.1 Wireless Vital Signs Monitoring System . . . 33
9.1.1 Setup . . . 33
9.2 Interface Output . . . 33
9.3 ECG . . . 33
9.3.1 AD8232 and Arduino Pro Mini Setup . . . 33
9.3.2 ECG Electrode Placement . . . 33
9.3.3 ECG Output . . . 35
9.3.4 Motion Artifacts . . . 36
9.4 Statistics . . . 37

10 Challenges and Limitations 38
10.1 Power Efficiency . . . 38

10.2 Interference . . . 38
10.3 Security Risk . . . 39
10.4 Data Management . . . 39
10.5 Regulation and Compliance Standards . . . 39
10.6 Reliability . . . 39
10.7 Range . . . 39
10.8 Raspberry Pi USB . . . 39

11 Conclusion **40**

11.1 Summary of Main Results . . . 40
11.2 Practical Interpretation of our Results . . . 40

A Program Installation **45**

A.1 Operating System - Raspbian Jessie . . . 46
A.2 Code . . . 47
A.2.1 Arduino Code for Thermistor . . . 47
A.2.2 MATLAB ECG Simulation . . . 48

B PCB Design and Schematics **54**

B.1 ECG Shield for AD8232 and Arduino Pro Mini . . . 55
B.2 Blood Temperature . . . 55
B.3 AD8232 Heart Rate Monitor SparkFun Implementation . . . 55
B.4 Raspberry Pi 3 Model B . . . 56

C Measurements and Datasheets **58**

C.1 Datasheet . . . 59

List of Figures

1.1 The Cardiac Depolarization Route. AVN: Atrioventricular Node; SAN: Sinoatrial Node. [23] 4

1.2 The Basic Pattern of Electrical Activity across the Heart [5] 5

3.1 Wireless Vital Signs Monitoring System Block Diagram 11

4.1 Raspberry Pi 3 Model B [18] 14

5.1 AD8232 Functional Block Diagram [1] 19

5.2 AD8232 Pin Configuration [1] 19

5.3 AD8232 Simplified Schematic Diagram [1] 20

5.4 Circuit Configuration for ECG Waveform Monitoring using the AD8232 [1] . . . 21

5.5 AD8232 Heart Rate Monitor from SparkFun Electronics [9] 22

5.6 AD8232 Connection Diagram using Fritzing [9] 23

5.7 Thermistor in Voltage Divider Configuration using Fritzing 24

5.8 Thermistor Formulae and Properties [31] 25

5.9 Thermistor Formulae and Properties (cont.) [31] 25

7.1 Simulated ECG Signal 31

9.1 Typical Sensor Placements [5] 34

9.2 Typical Sensor Placements (3 Electrodes) [9] 34

9.3 ECG Output Test 2 35

9.4 ECG Output recorded by the Raspberry Pi over WiFi 35

9.5 ECG Output with Deep Breathing 36

9.6 ECG Output with Limb Movement 36

A.1 NOOBS (New Out Of the Box Software) Initial Installation Screen [15] 46

B.1 AD8232 SparkFun Implementation Schematic Diagram [13] 55

B.2 Raspberry Pi 3 Model B Schematic Diagram [17] 56

C.1 Snippet of Material Type F Thermistor Datasheet [31] 59

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 [24]. 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 [10]. 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 [32].

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 [6], 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 [28]. Both the action potentials generated by individual cells and sequence of activation affect the signal

registered during electrocardiography [28]. 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" [28]. 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" [28].

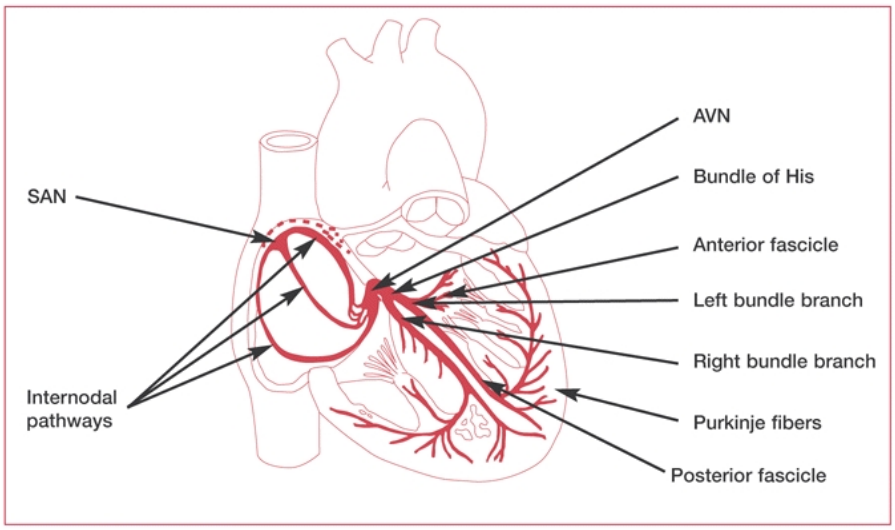

Figure 1.1: The Cardiac Depolarization Route. AVN: Atrioventricular Node; SAN: Sinoatrial Node. [23]

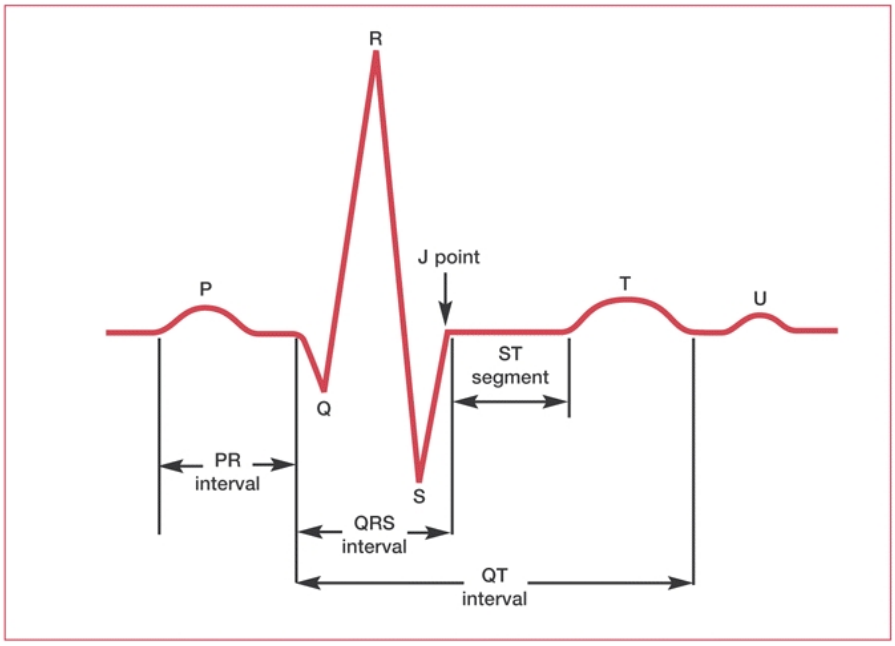

Figure 1.2: The Basic Pattern of Electrical Activity across the Heart [5]

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" [5]
- "Electrical activity away from a lead causes a downward deflection" [5]
- "Depolarization and repolarization deflections occur in opposite directions" [5]

Ashley and Niebauer (2004) (p.g. 19) [5] 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

1.4.3 Theory of Photoplethysmography

1.4.4 Theory of Blood Pressure

1.4.5 Theory of Blood Temperature

Chapter 2

Survey of Related Previous Work - Overview of Wireless Sensor Network Technology

2.1 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 [22].

2.2 Body Area Networks

2.3 Methods of Wireless Transmission

2.3.1 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.

2.3.2 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

2.4 WiFi

2.5 Bluetooth

2.6 Zigbee

2.7 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.

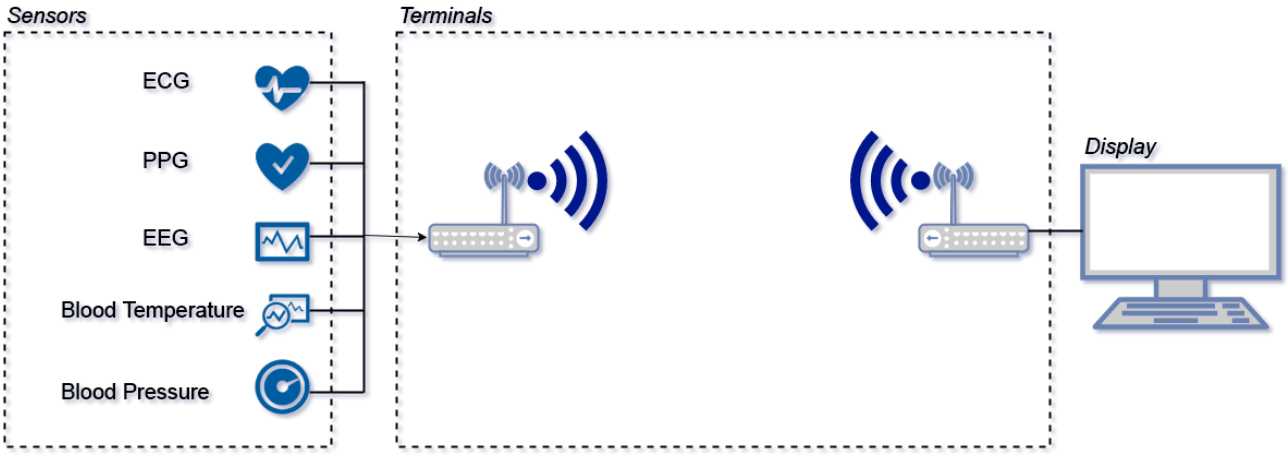

Figure 3.1: Wireless Vital Signs Monitoring System Block Diagram

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 [11]. 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 [11]. 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 [11]. 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 [12]

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**.

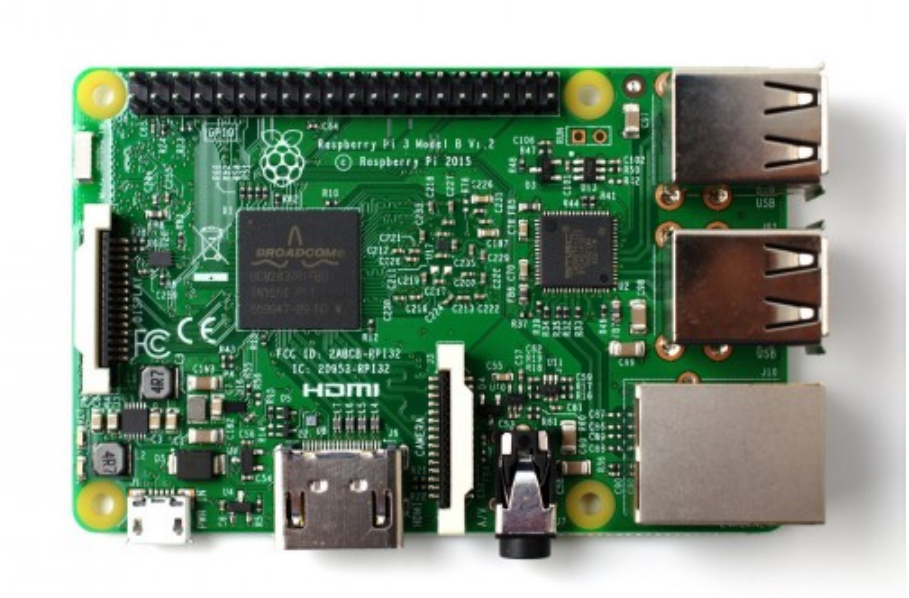

Figure 4.1: Raspberry Pi 3 Model B [18]

[16]

The specifications of the Raspberry Pi 3 Model B are as follows [16]:

- 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 5.3.2.

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.

[17]

4.2.1 Raspbian Jessie

Raspbian Jessie is [19] 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

4.2.2 Portable Power Supply for the Raspberry Pi 3 Model B

[14]

The Raspberry Pi is powered using a 5V micro-USB power supply [14]. 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 [14].

Buck - 9V

4.3 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 coupled with Phase Shift Oscillator

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 [25]. 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 [35] 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.

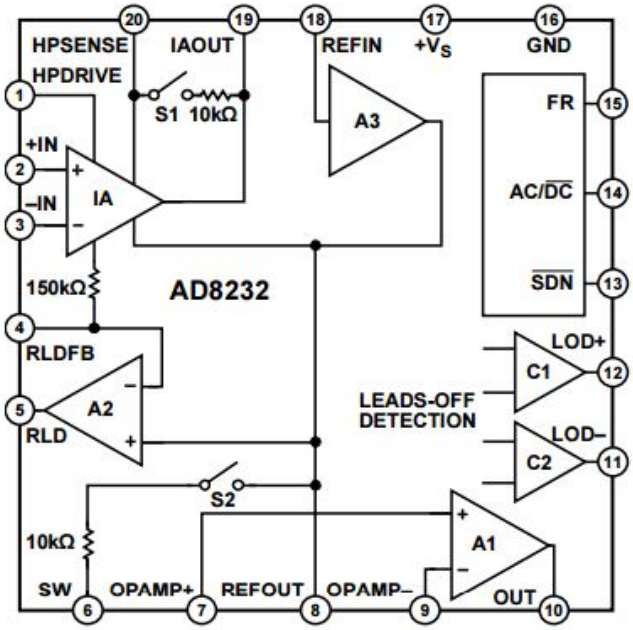

Figure 5.1: AD8232 Functional Block Diagram [1]

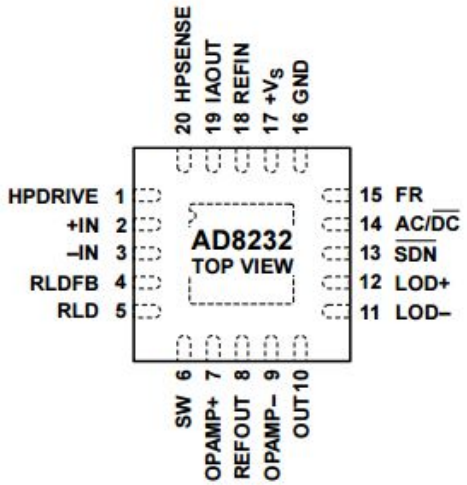

Figure 5.2: AD8232 Pin Configuration [1]

Theory of Operation of the AD8232

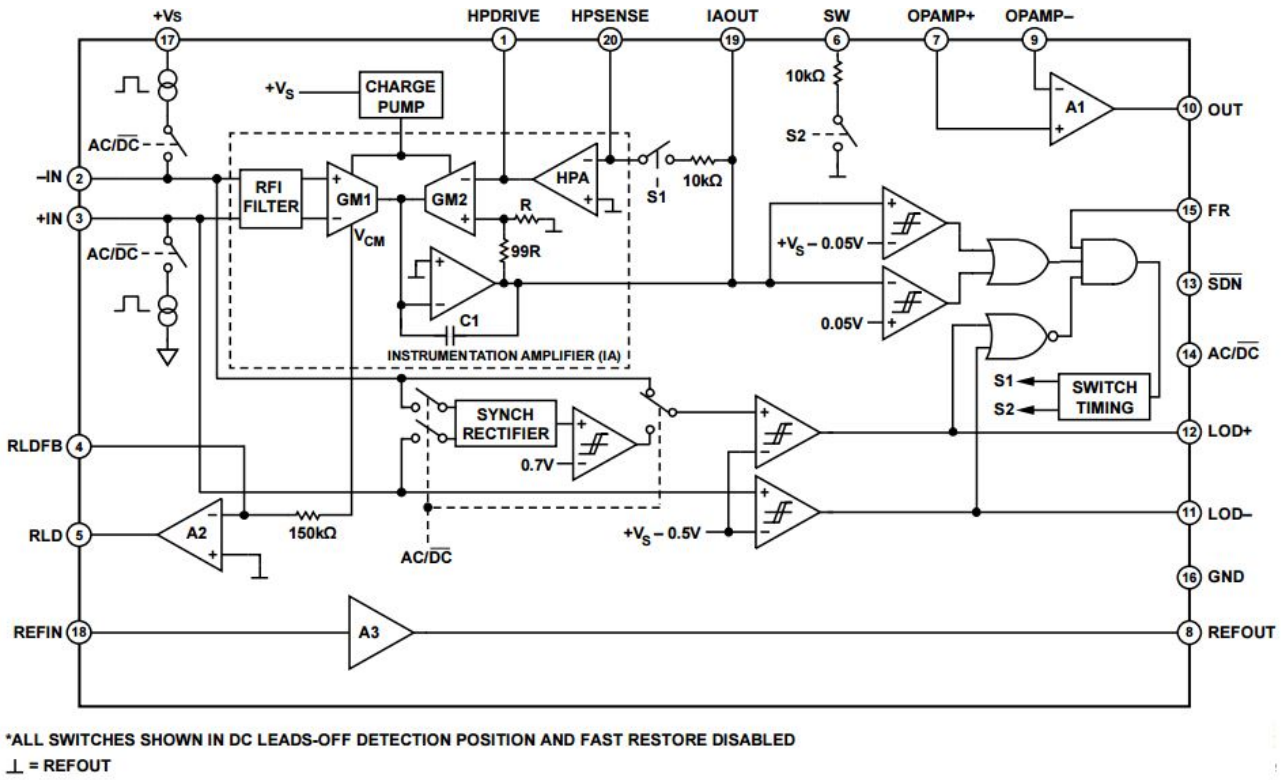

Figure 5.3: AD8232 Simplified Schematic Diagram [1]

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 [9].

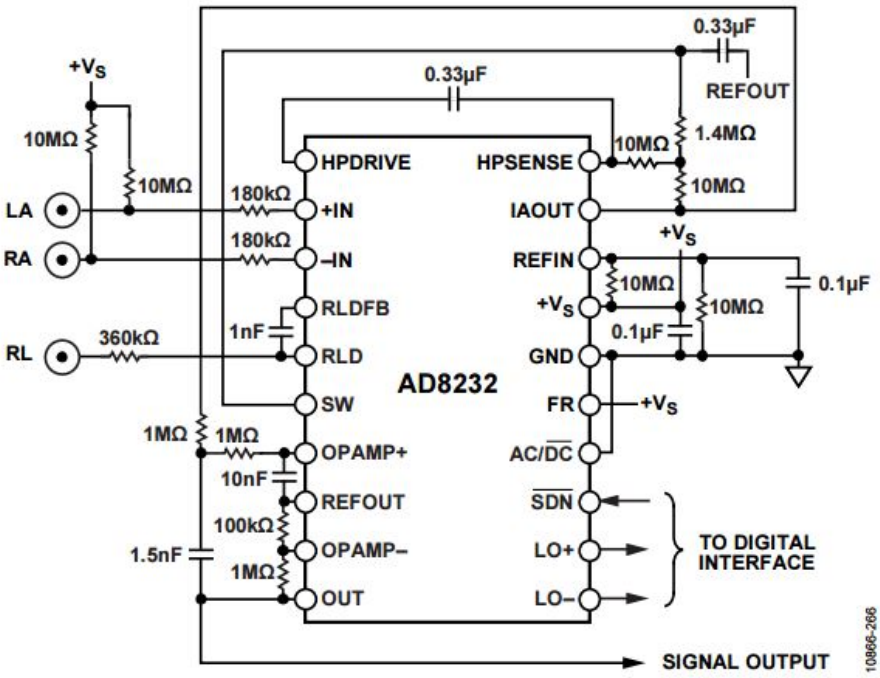

Figure 5.4: Circuit Configuration for ECG Waveform Monitoring using the AD8232 [1]

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.

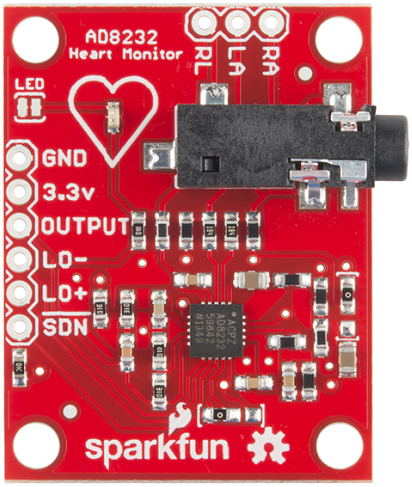

Figure 5.5: AD8232 Heart Rate Monitor from SparkFun Electronics [9]

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 [9].

The assembly of the complete AD8232 ECG sensor requires the following components [9]:

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.

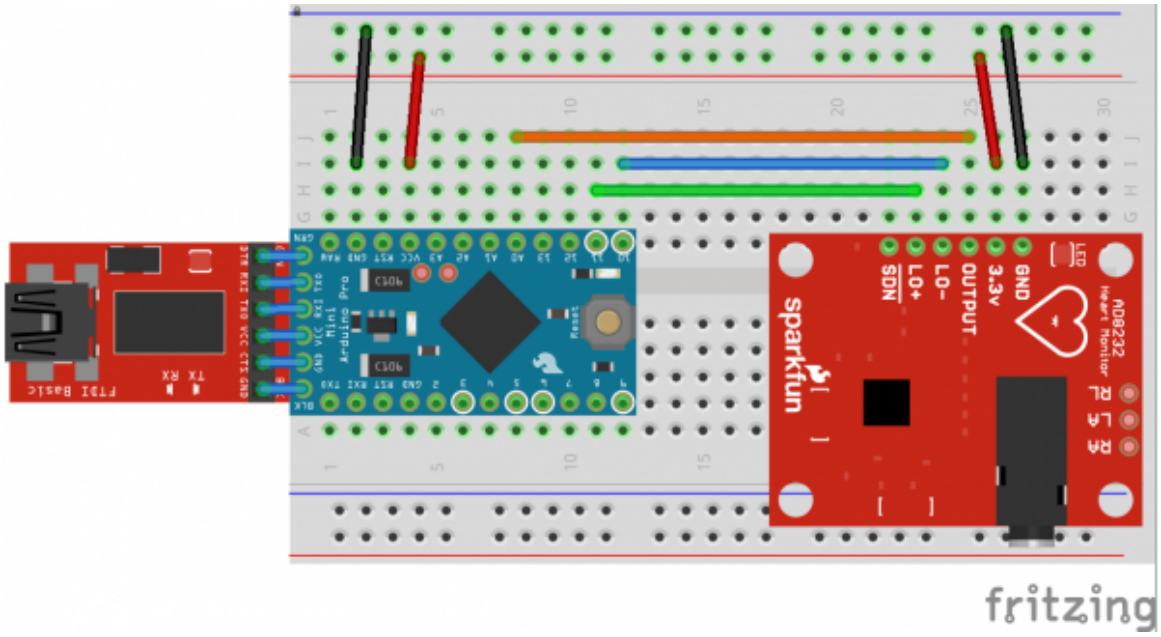

Figure 5.6: AD8232 Connection Diagram using Fritzing [9]

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 [9]

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

FIGUREEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

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.

- Uploaded and compiled sample Arduino code from SparkFun GitHub repository. - Need to change board type to Arduino Pro Mini 328 - 3.3V/8MHz.

5.2 Blood Temperature

5.2.1 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 [31]. The circuit in Figure 5.7 was constructed for testing the thermistor.

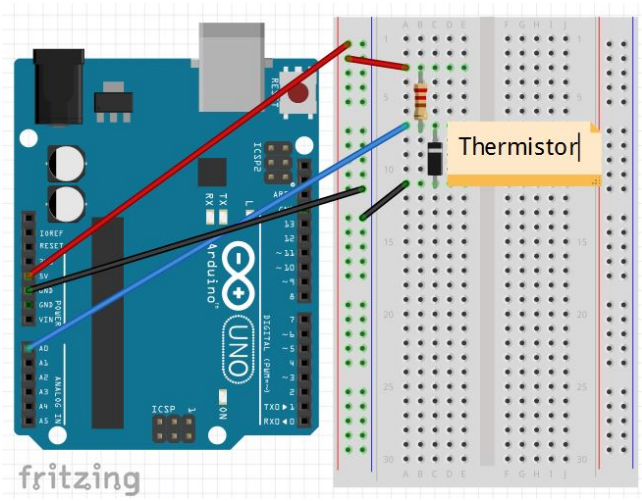

Figure 5.7: Thermistor in Voltage Divider Configuration using Fritzing

The datasheet of the material type F thermistor [31] 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 5.8: Thermistor Formulae and Properties [31]

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

†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 5.9: Thermistor Formulae and Properties (cont.) [31]

$$\frac{Rt}{R25} = exp(A + \frac{B}{T} + \frac{C}{T^2} + \frac{D}{T^3}) \tag{5.1}$$

$$\frac{1}{T} = a + b(ln\frac{Rt}{R25}) + c(ln\frac{Rt}{R25})^2 + d(ln\frac{Rt}{R25})^3 \tag{5.2}$$

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 5.1 and 5.2 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.2.1.

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.

5.2.2 Self-Heating - Phase Shift Oscillator

How does the phase shift oscillator actually work

How does the PSO help in reducing self-heating Frequency becomes temperature dependent - Voltage not high across the thermistor

PSO - Bipolar transistor Implementation [36]

5.2.3 Astable Operation of 555 Timer

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

555 Timer Datasheet.

5.3 Interface

5.3.1 USB

5.3.2 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

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 [20].
- **X Tight VNC Viewer** - Allows the Display-connected Terminal to access and view the Sensor-connected Terminal via the VNC server [20].
- **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 [26].
- **Processing 2.2.1** - Software sketchbook which receives serial inputs and renders real-time graphs of given information [8].
- **Java 7** - Allows Java-based applications to run on the Platform (specifically required for Processing 2.2.1) [29].

```
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
```

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

- Also, need to use Processing 2.2.1, not 3.1.1. - Devices will not function with new version of Processing, displaying size() error. - Might be necessary to uninstall package 'libgles2-mesa' before using Processing, to prevent startup errors related to the P2D and P3D renderers. - Downloaded Processing 2.2.1 from https://processing.org/download/?processing for Linux 32 platforms. - processing-2.2.1-linux32.tgz is a 98.4 MB file.

```
sudo apt-get update'
sudo apt-get install oracle-java7-jdk
sudo update-alternatives --config java
```

- Processing 2.2.1 runs well on Java 7, not Java 8. - By default the Raspbian Jessie ships with Java 8. - Need to install Java 7 using the following commands: 1. 'sudo apt-get update' 2. 'sudo apt-get install oracle-java7-jdk' 3. 'sudo update-alternatives –config java' - Extracted the tar file using 'tar xvzf processing-2.2.1-linux32.tgz'. - Need to remove the x86 Java runtime and replace with RPi armhf version. Use: 1. 'rm -rf /processing-2.2.1/java 2. 'ln -s /usr/lib/jvm/jdk-7-oracle-armhf /processing-2.2.1/java'

- 'ln -s /usr/lib/jvm/jdk-7-oracle-armhf /processing-2.2.1/java' does not work. Folder name is different. - Copy '/usr/lib/jvm/jdk-7-oracle-arm-vfp-hflt' to the 'java' folder in processing-2.2.1. - Change 'myPort = new Serial(this, Serial.list()[1], 9600);' depending on which serial port is used.

6.1 Processing 2.2.1

6.2 Serial Input

6.3 Code

6.4 Visualisation

6.5 Wireless Transmission

For this project, wireless communications between the Sensor-connected Terminal and the client requires the use of VNC as seen in Section 2.3.2.

To set up the Raspberry Pi as a VNC server, TightVNCServer was used . For the Windows client used to access the VNC server of the Raspberry Pi, Tight VNC Viewer was used [33].

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.2.2 [30].

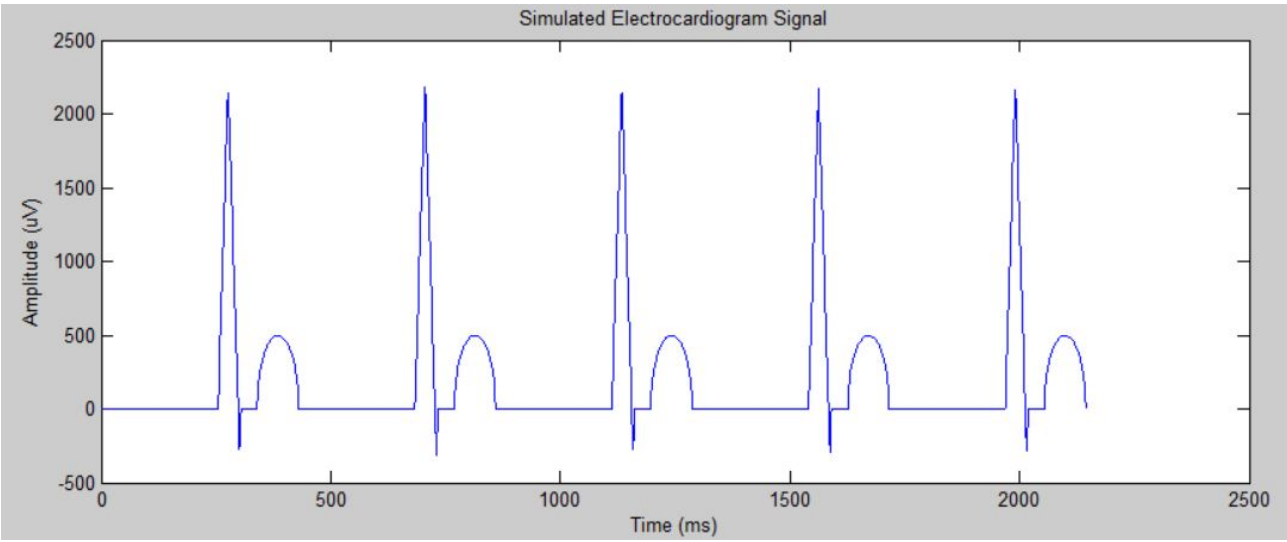

Figure 7.1: Simulated ECG Signal

Chapter 8

Testing

wireless VNC

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 [5]. The typical attachment sites are found in Figure 9.1 below.

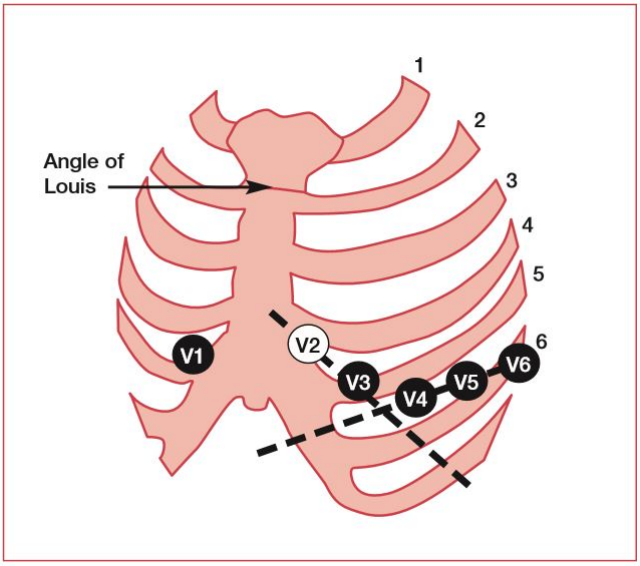

Figure 9.1: Typical Sensor Placements [5]

The AD8232 Heart Rate Monitor from SparkFun Electronics [9] 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.

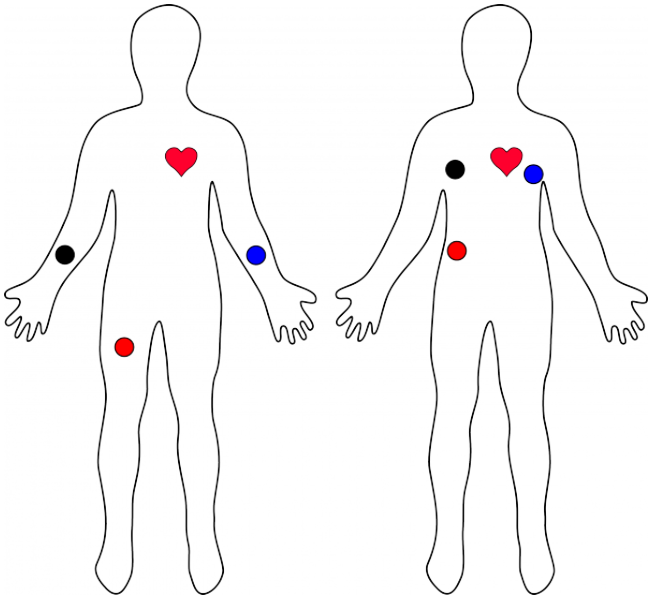

Figure 9.2: Typical Sensor Placements (3 Electrodes) [9]

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.

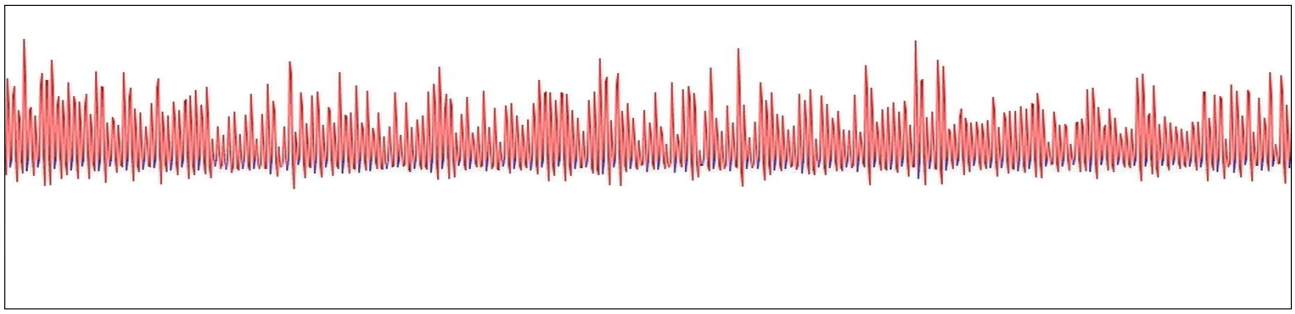

Figure 9.3: ECG Output Test 2

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.

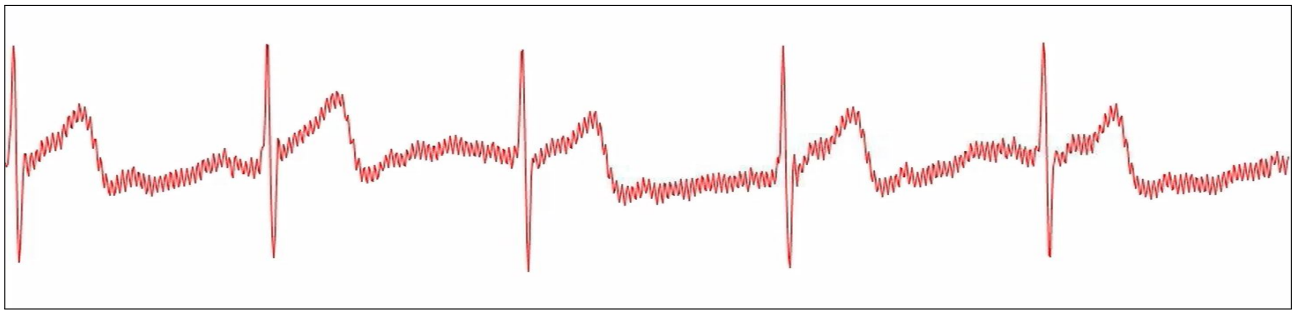

Figure 9.4: ECG Output recorded by the Raspberry Pi over WiFi

9.3.4 Motion Artifacts

At its core, ECG records electrical activity from muscle contractions through electrodes located on the skin [5]. 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.

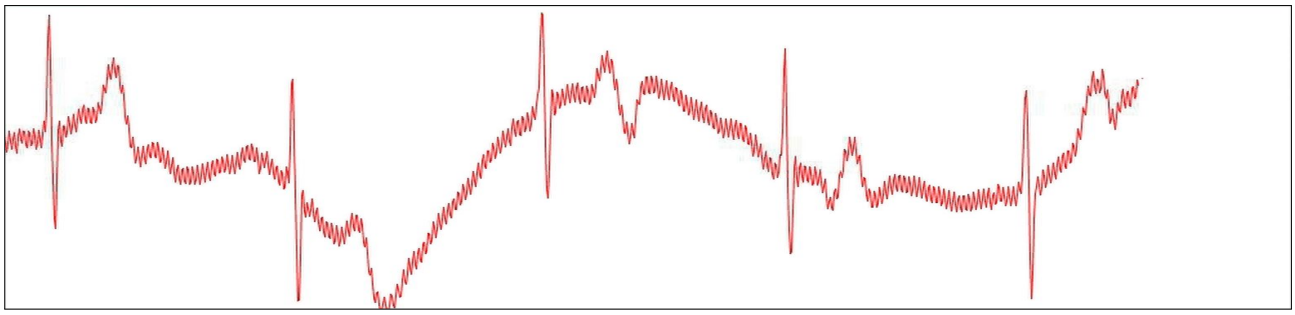

Figure 9.5: ECG Output with Deep Breathing

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

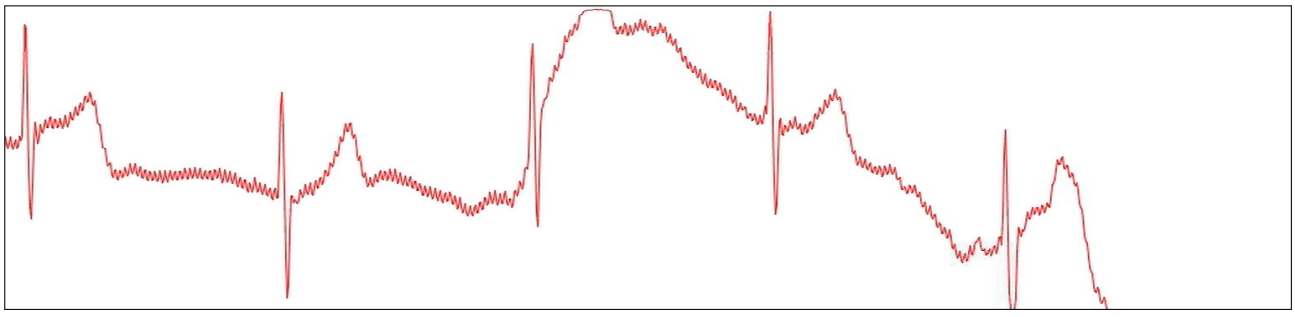

Figure 9.6: ECG Output with Limb Movement

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 Statistics

SRW

Chapter 10

Challenges and Limitations

10.1 Power Efficiency

According to Sohraby, Minoli, and Taieb [34], 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."

10.2 Interference

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

10.3 Security Risk

10.4 Data Management

10.5 Regulation and Compliance Standards

10.6 Reliability

Raspberry Pi has "uptime"

10.7 Range

10.8 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 [17]. The maximum speed of this root USB port is 480Mbps [14]. 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] Euan A Ashley and Josef Niebauer. *Conquering the ECG*. Remedica, 2004.

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

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

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

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

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

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

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

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

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

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

[22] 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.

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

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

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

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

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

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

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

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

[31] Electrocomponents PLC. Ntc thermistors. `http://docs-europe.electrocomponents.com/webdocs/14c2/0900766b814c2f5a.pdf`. Accessed: 2016-9-26.

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

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

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

[35] 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.

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

Appendix A

Program Installation

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) [15]. 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.

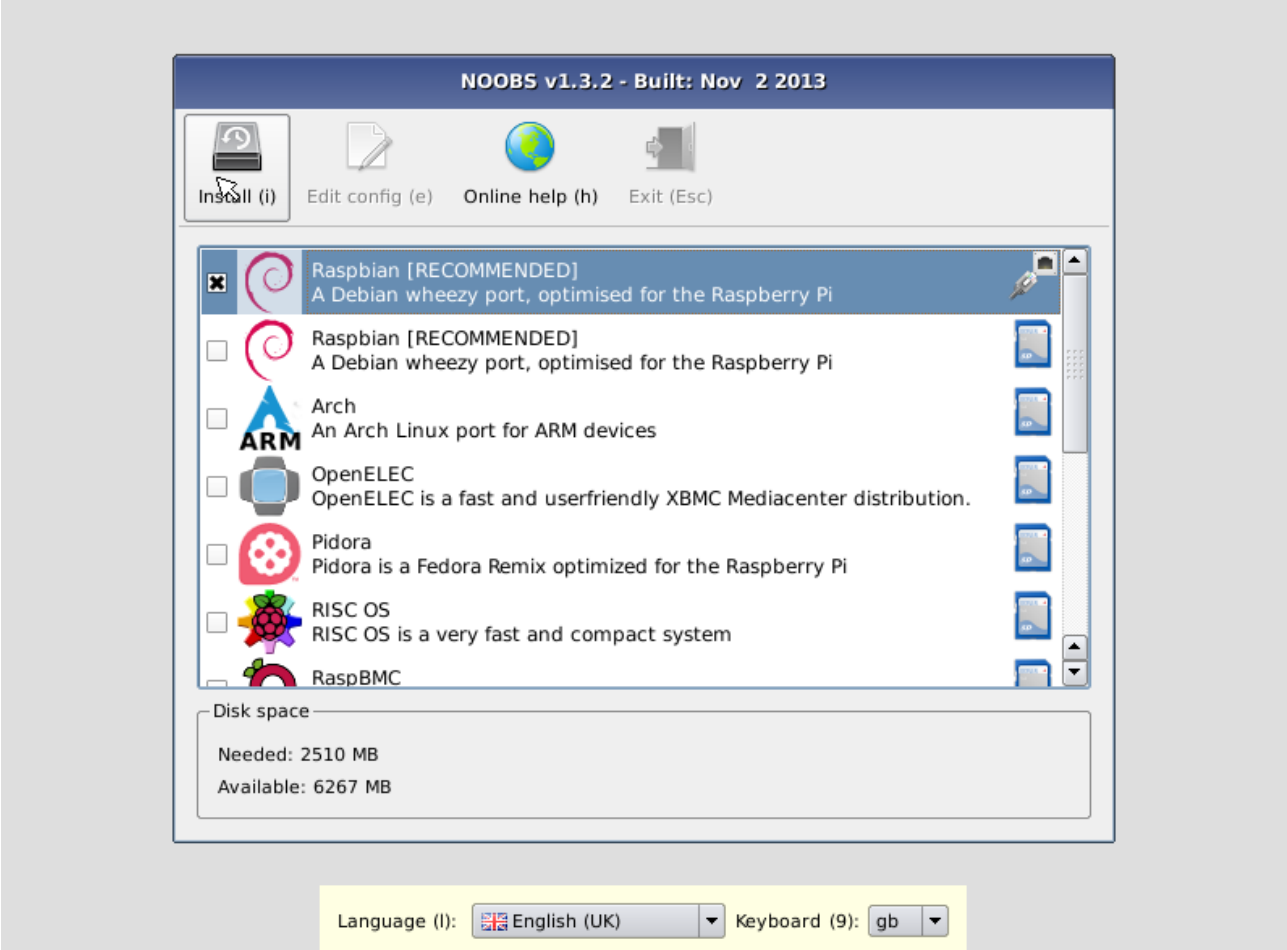

Figure A.1: NOOBS (New Out Of the Box Software) Initial Installation Screen [15]

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.

A.1 Operating System - Raspbian Jessie

```
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 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.2.2 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 [30].

```
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

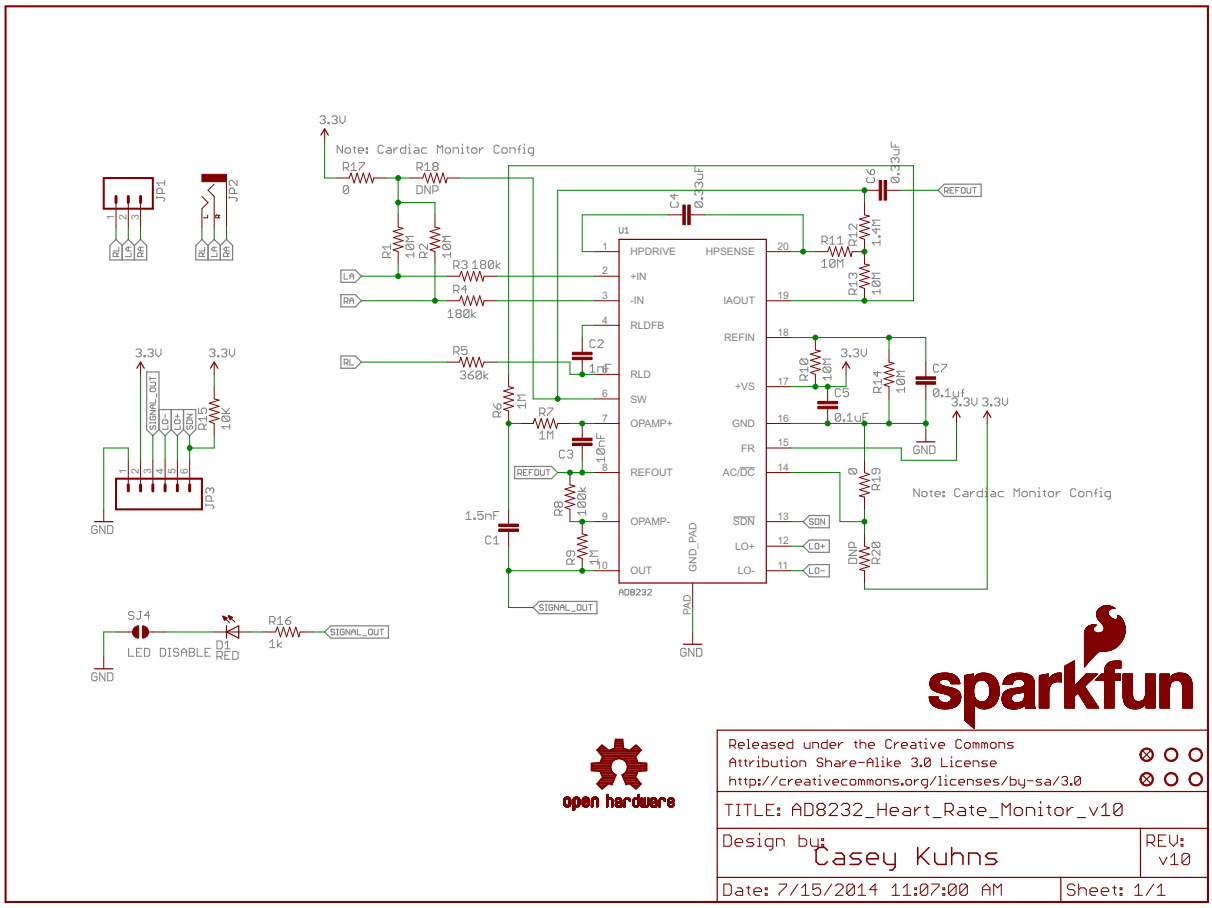

Figure B.1: AD8232 SparkFun Implementation Schematic Diagram [13]

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

B.4 Raspberry Pi 3 Model B

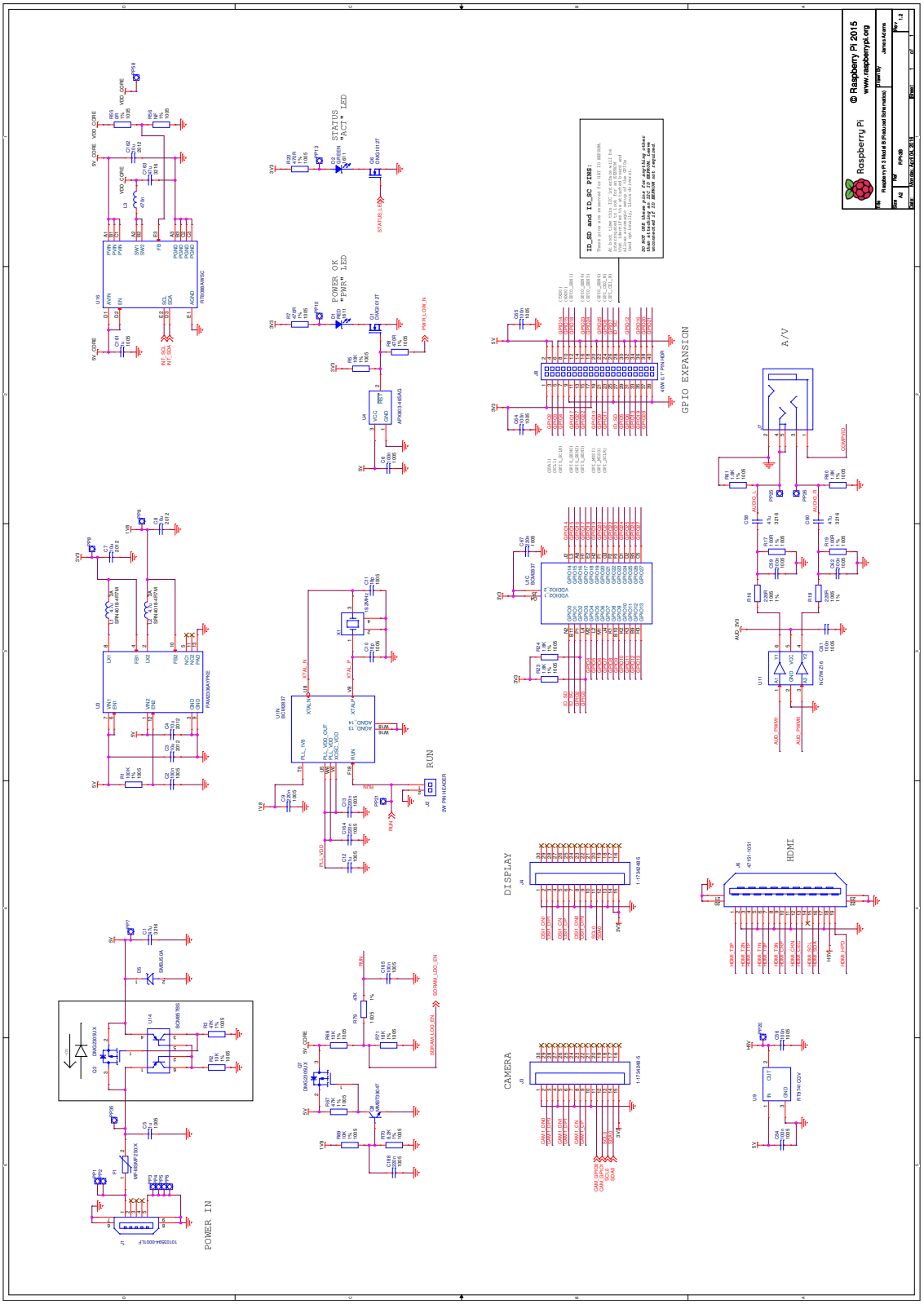

Figure B.2: Raspberry Pi 3 Model B Schematic Diagram [17]

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

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%