**A REPORT**

**ON**

# UBIQUITOUS HEALTH MONITORING AND DRIVER FATIGUE DETECTION USING PPG

BY

SHRIHARI VISWANATH (2016A8PS0396P) ELECTRONICS & INSTRUMENTATION ENGINEERING

Prepared in fulfillment of the
Internship

AT

**CSIR- Central Electronics Engineering and Research Institute, Chennai**


**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**
**PILANI**
JULY, 2018

**Station:** Central Electronics Engineering Research Institute **Centre:** Chennai

**Duration:** 9 Weeks

**Date Of Start**: 21-May-2018

**Date Of Submission:** 18-July-2018

**ID No. :** 2016A8PS0396P

**Name :** Shrihari Viswanath

**Discipline:** B.E. (hons) Electronics and Instrumentation Engineering

**Project Title:** AI based healthcare system. (Driver fatigue detection)

**Name and Designation of the expert:** Dr. Bala Pesala, Principal Scientist, CSIR CEERI.

**Project Areas:** Signal processing, Filter design, PCB development, Circuit prototyping

**Abstract:**

Photoplethysmography (PPG) is one of the primary technology that has made it possible to build wearable products that can capture and process bio signals generated by the human body. The aim of this project is to make a prototype for a wearable that can acquire continuous PPG signals using a sensor. The hardware implementation of this prototype has to be done to acquire, amplify and filter the signal and then interface it with a micro-controller. Finally it should be displayed on a mobile application to view health parameters in real time. The long term goal of this project is to detect fatigue in a driver.

**Signature of Student** **Signature Of Scientist**

**Date** **Date**

# ACKNOWLEDGEMENTS

We express our sincere thanks to Dr. Bala Pesala, Principal Scientist,CSIR for guiding us right from the inception. We sincerely acknowledge him for extending his valuable guidance, support for literature and providing valuable critique of our work.

We would like to express our utmost thanks to Mr. Satish Bindal, for taking out time from his busy schedules to share his experience and knowledge with us.

We also express our special gratitude to Dr. Madan Kumar Lakshmanan for imparting knowledge during the weekly meetings and beyond.

We also acknowledge Ms. Meera Subramaniam for making the experience memorable and helping us with the issues we faced in the process.

We would also like to express our gratitude to Mr. Palaniandi, Technical officer, CEERI who ensured that we had a comfortable working environment at CEERI.

# TABLE OF CONTENTS

*List of Abbreviations*……...*i*
*Organisation Profile*……..*ii*

1 Introduction…….....8
- 1.1 Physiology and Function of the Heart
- 1.2 Cardiac Parameters
- 1.3 Pulse Oximetry
- 1.4 Transimpedance Amplifier
- 1.5 Bandpass Filter
- 1.6 Notch Filter
- 1.7 ADC
- 1.8 PCB
- 1.9 Protocentral Pulse Sensor Board

2 Scope/Aim of the project……...12

3 Methodology……..13
- 3.1 Pulse Oximetry Sensor
- 3.2 Transimpedance Amplifier
- 3.3 Filter Design
- 3.4 PCB Design
- 3.5 Soldering Process
- 3.6 Analog to Digital Conversion
- 3.7 Spectrum Analysis

4 Observations, Simulations and Results……27
- 4.1 Control Circuit
- 4.2 Protocentral PPG Board

5 Problems Faced……36
- 5.1 Stray Capacitance
- 5.2 Gain Control
- 5.3 PCB Size
- 5.4 Motion Artifact

6 Conclusion……41

7 References……42

8 Appendix……...43
- 8.1 Processing Code for Data
- 8.2 Processing Code for SpO2
- 8.3 Processing Code for Graphing

## LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| 1. | PPG | Photoplethymography |
| 2. | HRV | Heart rate variability |
| 3. | TIA | Transimpedence amplifier |
| 4. | HR | Heart rate |

# ORGANISATION PROFILE

Central Electronics Engineering Research Institute (CEERI) is a part of the Council of Scientific and Industrial Research (CSIR) and is committed to research and Development in electronics to meet the requirements of its customers. CEERI has been continually identifying the thrust areas of key National/Industrial relevance and has been developing competitive technologies with a goal to achieve excellence and self-reliance in these areas. CEERI Centre in Chennai, a pioneering institution in the field of Quality Control Instrumentation for process industries, is a regional centre of Central Electronics Engineering Research Institute (CEERI), Pilani,Rajasthan. The centre has a rich experience and expertise in this field and has developed many online monitoring systems for various process on indigenous development of special sensors and systems suited to the online measurement and control and these involve varying technologies such as Near Infrared (NIR) Gauging. Beta Gauging, Optical, Electromechanical methods and the currently emerging Image Processing techniques. The centre has been extensively contributing its expertise in Lab VIEW for online apple sorting. The centre has also unveiled a system for sorting plastics through NIR spectroscopy.

VISION The Centre has ambitious plans to develop near-infrared instrumentation, electro optical and machine vision system technologies for quality assessment/assurance, grading, sorting of a variety of agricultural/horticultural products and processed foods to help in reducing wastage, optimize energy usage, enhance productivity, value addition and compete in global markets. Plans are afoot for application of near infrared spectroscopy and chemometrics to evolve standardization and classification of herbal, pharmaceutical, edible oil and such products for their quality and authenticity. Also, the Centre takes up sponsored, collaborative, grant-in-aid projects under contract research & development from private and government agencies for the development of instrumentation systems at an affordable cost. Apart from this, the Centre undertakes consultancy projects and technical services to fulfil the specific needs of the industries in its areas of expertise.

AREAS OF EXPERTISE: Machine Vision Technologies, Advanced Image Processing Solutions, Process Automation Systems, Customized Quality Control Systems, Advanced Photonic Sensor (including NIR) Technologies, Digital Signal Processing Solutions, Integrated Total Quality Management Systems, and Test & Measurement Systems for Process Industries

RECENT ACHIEVEMENTS: The Centre has developed an image analysis based system - Herbas, for the authentication of certain herbal plant raw material in collaboration with IICT, Hyderabad and Arya Vaidya Sala, Kottakkal. The system captures the image of the cross section of the stem using a trinocular microscope with a CCD camera and the software offers several capabilities for the user to authenticate the plant for its medicinal use. The system is used currently in Arya Vaidya Sala, Kottakkal.

Another program, which is very much useful in solid waste management, is the development of an on-line sorter to segregate different types of plastics coming from a pile of plastic waste collected. This helps in easy management of such waste for recycling. The Ministry of Environment & Forests, New Delhi has funded this project and laboratory level bench model system has been demonstrated to senior officials of Ministry and other stakeholders. The project on “Paper dirt speck analyzer”, to design and develop a PC based imaging system to analyze dirt speck and pinholes in paper samples using image analysis has been successfully completed and a prototype has been tested at a paper industry. The Centre has completed the

project on “Development of on-line mango sorting system using soft x-ray imaging”, sponsored by Dept. of Science & Technology, New Delhi by developing a soft X-ray imaging based technology for analysing export variety mangoes for internal disorders and rejecting the infected ones by using a suitable conveyor arrangement.

# 1. INTRODUCTION

With the advent of healthcare technologies, portability of health-care devices is a major concern. One such healthcare device is an Electrocardiograph monitor which requires electrodes in contact with the body. However, it is not portable and often causes irritation on continued use. This project aims at innovating a non-contact electrode system to monitors the user's Heart Rate and Heart Rate Variability. The long-term goal of the watch is to detect the causes of fatigue in the individual. The data collected using the sensors includes Heart Rate and Heart Rate Variability. These values would be monitored continuously for irregularities and used to detect fatigue in an individual.

## 1.1 PHYSIOLOGY AND FUNCTION OF THE HEART:

The heart has four chambers. The upper two chambers (left/right atria) are entry-points into the heart, while the lower two chambers (left/right ventricles) are contraction chambers sending blood through the circulation.

The cardiac cycle refers to a complete heartbeat from its generation to the beginning of the next beat, comprising several stages of filling and emptying of the chambers. The frequency of the cardiac cycle is reflected as heart rate (beats per minute, bpm).

## 1.2 CARDIAC PARAMETERS:

**Heart Rate (HR):** HR reflects the frequency of a complete heartbeat from its generation to the beginning of the next beat within a specific time window. It is typically expressed as beats per minute (bpm).

**Inter-Beat Interval (IBI).** The IBI is the time interval between individual beats of the heart.

**Heart rate variability** (**HRV**): The physiological phenomenon of variation in the time interval between heartbeats.

## 1.3 PULSE OXIMETRY

The pulse oximeter has revolutionized modern medicine with its ability to continuously and transcutaneously monitor the functional oxygen saturation of hemoglobin in arterial blood (SaO2). Pulse oximetry is so widely prevalent in medical care that it is often regarded as a fifth vital sign. It is important to understand how the technology functions as well as its limitations because erroneous readings can lead to unnecessary testing. Frequent false alarms in the intensive care unit can also undermine patient safety by distracting caregivers. To recognize the settings in which pulse oximeter readings of oxygen saturation (SpO2) may result in false estimates of the true SaO2, an understanding of two basic principles of pulse oximetry is required: (i) how oxyhemoglobin (O2Hb) is distinguished from deoxyhemoglobin (HHb) and (ii) how the SpO2 is calculated only from the arterial compartment of blood.

Pulse oximetry is based on the principle that O2Hb and HHb differentially absorb red and near-infrared (IR) light. It is fortuitous that O2Hb and HHb have significant differences in absorption at red and near-IR light because these two wavelengths penetrate tissues well

whereas blue, green, yellow, and far-IR light are significantly absorbed by non-vascular tissues and water.O2Hb absorbs greater amounts of IR light and lower amounts of red light than does HHb; this is consistent with experience – well-oxygenated blood with its higher concentrations of O2Hb appears bright red to the eye because it scatters more red light than does HHb. On the other hand, HHb absorb more red light and appears less red. Exploiting this difference in light absorption properties between O2Hb and HHb, pulse oximeters emit two wavelengths of light, red at 660 nm and near-IR at 940 nm from a pair of small light-emitting diodes located in one arm of the finger probe. The light that is transmitted through the finger is then detected by a photodiode on the opposite arm of the probe; i.e., the relative amount of red and IR light absorbed are used by the pulse oximeter to ultimately determine the proportion of Hb bound to oxygen.

The ability of pulse oximetry to detect SpO2 of only arterial blood is based on the principle that the amount of red and IR light absorbed fluctuates with the cardiac cycle, as the arterial blood volume increases during systole and decreases during diastole; in contrast, the blood volume in the veins and capillaries as well as the volumes of skin, fat, bone, etc., remain relatively constant. A portion of the light that passes through tissues without being absorbed strikes the probe's photodetector and, accordingly, creates signals with a relatively stable and non-pulsatile "direct current" (DC) component and a pulsatile "alternating current" (AC) component. A cross-sectional diagram of an artery and a vein during systole and diastole illustrates the non-pulsatile (DC) and pulsatile (AC) compartments of arteries and the relative absence of volume change in veins and capillaries. Pulse oximeters use amplitude of the absorbances to calculate the Red:IR Modulation Ratio (R); i.e., R = (Ared,AC/Ared,DC)/(AIR,AC/AIR,DC) where A = absorbance; in other words, R is a double-ratio of the pulsatile and non-pulsatile components of red light absorption to IR light absorption. At low arterial oxygen saturations, where there is increased HHb, the relative change in amplitude of the red light absorbance due to the pulse is greater than the IR absorbance, i.e., Ared,AC > AIR,AC resulting in a higher R value; conversely, at higher oxygen saturations, AIR,AC > Ared,AC and the R value is lower. A microprocessor in pulse oximeters uses this ratio (calculated over a series of pulses) to determine the SpO2 based on a calibration curve that was generated empirically by measuring R in healthy volunteers whose saturations were altered from 100% to approximately 70%. Thus, SpO2 readings below 70% should not be considered quantitatively reliable although it is unlikely any clinical decisions would be altered based on any differences in SpO2 measured below 70%. **[2][5]**

Spo2 calculations:

$$\mathrm{SpO_2} = \alpha - \beta \frac{w_{\lambda 1}}{w_{\lambda 2}} = \alpha - \beta \frac{I_{AC}^{\lambda 1}/I_{DC}^{\lambda 1}}{I_{AC}^{\lambda 2}/I_{DC}^{\lambda 2}}$$

$$\mathrm{SpO_2} = \alpha - \beta R$$

A common problem in pulse-oximetry is the decreased accuracy for individuals with a dark skin, especially the overestimation of $SpO_2$ during de-saturation. Because of the higher melanin content of their skin, more light is absorbed, resulting in reflected light with a lower pulsatile amplitude compared to skin with a lower pigmentation level. Skin reflectance is rather uniform in the infrared part of the light spectrum. Consequently, whereas both the AC and DC components decrease for increasing melanin content, the ratio of both remains stable, resulting in a stable signature-vector $\vec{P}_{bv}$ over the entire range of skin pigmentation

level. As a result, the calibration coefficients are expected to be independent of the skin pigmentation level.

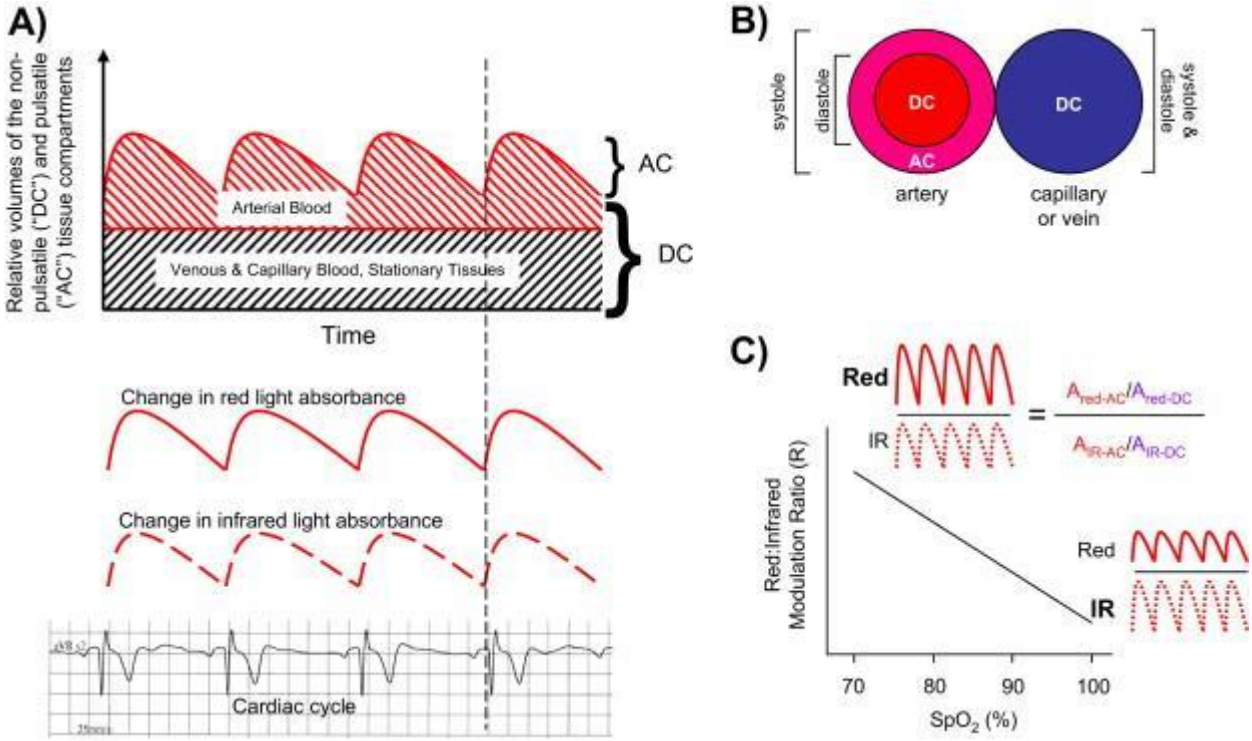



**Fig.1.1** *Schematic diagram of light absorbance by a pulse oximeter*

## 1.4 TRANSIMPEDANCE AMPLIFIER

In electronics, a transimpedance amplifier (TIA) is a current-to-voltage converter, most often implemented using an operational amplifier. The TIA can be used to amplify the current output of photo detectors and other types of sensors to a usable voltage. Current-to-voltage converters are used with sensors that have a current response that is more linear than the voltage response.

## 1.5 BANDPASS FILTER

A band-pass filter (also bandpass filter, BPF) is a device that passes frequencies within a certain range and rejects (attenuates) frequencies outside that range.

A band-pass filter can be characterized by its Q factor. The Q-factor is the reciprocal of the fractional bandwidth. A high-Q filter will have a narrow passband and a low-Q filter will have a wide passband. These are respectively referred to as *narrow-band* and *wide-band* filters.

The Butterworth filter is a type of signal processing filter designed to have a frequency response as flat as possible in the passband. It is also referred to as a maximally flat magnitude filter.

## 1.6 NOTCH FILTER

In signal processing, a band-stop filter or band-rejection filter is a filter that passes most frequencies unaltered, but attenuates those in a specific range to very low levels. It is the opposite of a band-pass filter. A notch filter is a band-stop filter with a narrow stopband (high Q factor). Here it is used to attenuate the power line interference at 50Hz.

## 1.7 ADC

In electronics, an analog-to-digital converter (ADC, A/D, or A-to-D) is a system that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal. An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number representing the magnitude of the voltage or current. Typically the digital output is a two's complement binary number that is proportional to the input, but there are other possibilities.

## 1.8 PCB

A printed circuit board (PCB) mechanically supports and electrically connects electronic components or electrical components using conductive tracks, pads and other features etched from one or more sheet layers of copper laminated onto and/or between sheet layers of a non-conductive substrate. The sensor SFH7050 is with pins under pads. Hence the PCB design becomes the integral part of the hardware implementation for the purpose of testing the sensor**. [11]**

## 1.9 PROTO CENTRAL PULSE SENSOR BOARD

The Pulse sensor board from ProtoCentral is a new open medical hardware. Maxim's MAX30100 is is the sensor used in this board. Measuring 14mm wide and 40 mm long, this board is easily wearable on the finger to measure the pulse of blood. It has a Velcro strap to hold it.

The MAX30100 is an integrated pulse oximetry and heartrate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals. The MAX30100 operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.

Features:

- Integrates a complete pulse oximeter including the optics. No need for a separate probe.
- Easy-to-use I2C interface to connect to any host microcontroller
- Arduino library available
- Calculates Spo2 values with provided code
- Real-time display of PPG (Photoplethysmogram)

# 2. SCOPE/AIM AND OBJECTIVES OF THE PROJECT

Fatigue is a problem with severe consequences for drivers and people who work in high-risk situations such as mining, operation of heavy machinery, and for workers in factories, chemical/nuclear plants, etc. Current methods to implement fatigue detection in automotive sector include techniques such as blink frequency detection, posture/movement analysis etc. However, the reliability of these methods is not very high due to varying lighting conditions and inherent person to person variation. Unlike factors like drunkenness which can easily be estimated using breath analysers, fatigue and drowsiness resulting from fatigue are relatively hard to detect. Hence, it is vital to have devices/systems to check a driver's fatigue levels continuously.

Heart Rate Variability (HRV) is an emerging field of research for physiological monitoring. Variations in the heart rate can indicate the activity of the nervous system which can be a potential indicator of fatigue. An ideal PPG signal can be continuously monitored to look for any abnormalities and fatigue detection can be done. This can be done without causing any disturbance in one's day to day life by making a wearable watch prototype.

# 3. METHODOLOGY

Stages involved in the project

1. Obtain raw PPG signal from the Biomon sensor
2. Amplify the signal and filter out noises from the signal
3. Process the signal and then display the data on a mobile application

## 3.1 PULSE OXIMETRY SENSOR

This application note describes the use of the SFH 7050 as the sensor element for a photoplethysmography system. The sensor is designed for reflective photoplethysmography (PPG) as advances in signal processing and high efficient LEDs enable small and powerful reflective photoplethysmography sensors (like the SFH 7050, see Fig. 1). This is especially important as reflected PPG signals can be measured on body areas where transmittive PPG can't be applied allowing wearable PPG sensors.

By analyzing the PPG signal various parameters can be derived. Among them is the heart rate. In addition the oxygenation saturation level of arterial blood can be determined by measuring the absorption at two different wavelengths. Oxygen saturation ($SpO_2$) is a vital parameter as it is the level of oxyhemoglobin ($HbO_2$) in arterial blood (usually $SpO_2$ expresses the percentage of saturation). In the human body, $SpO_2$ is defined as the ratio of $HbO_2$ concentration to the total haemoglobin concentration present in the arterial blood.

The SFH 7050 is a fully integrated optoelectronic sensor, specially designed and optimized for reflective photoplethysmography. It features three LEDs – green (535 nm), red (660 nm) and IR (940 nm) - and a large area photodiode (PD) to maximize signal level (see Fig. 2 for LED spectra / PD responsivity). The device design includes a light barrier to minimize internal crosstalk thus enhancing the signal-to-noise ratio.
The sensor allows measurement of heart rate only (by powering only one LED) and other vital parameters by using the red and infrared LEDs or both.

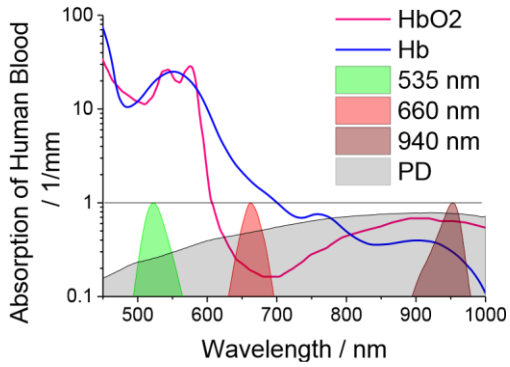



***Fig 3.1*** *Absorption of human blood (oxyhemoglobin, $HbO_2$ and hemoglobin, Hb) vs. wavelength of light.*

### 3.1.1 DETECTOR

The photodiode features a low dark current, suitable for low noise applications. Additionally the photodiode is highly linear to enable accurate $SpO_2$ measurements. The photodiode current is typically amplified and converted into a voltage with an external transimpedance amplifier. The low capacitance and the fast response of the photodiode make it suitable for short pulse operation to minimize power consumption.

### 3.1.2 APPLICATION ENVIRONMENT

The SFH 7050 is designed to operate close to human skin, any additional air-gap between human skin and the sensors surface can reduce the signal strength. Additionally, the infrared LED can be used as a proximity sensor to indicate the presence of skin. This allows to start measurements when the sensor is close to the skin or to display an out of reach message. Operating the SFH 7050 with a cover glass might cause optical crosstalk. Crosstalk needs to be reduced or avoided as it reduces the signal-to-noise ratio. For larger air-gaps a proper optical aperture design or light baffles between the LEDs and the detector might be required.

### 3.1.3 OPERATING THE SFH7050

There are (slightly) different measurement requirements concerning the application scenario:

- heart rate only
- heart rate plus pulse oximetry

In case of heart rate only designs the DC component of the photocurrent can be neglected; only the periodicity of the AC component ($I_{max} - I_{min}$, frequency) is of interest. For pulse oximetry the DC as well as the AC components ($I_{min}$, $I_{max}$) are needed. Thus in general, a pure heart rate device is easier to implement as it requires only one LED. For most body locations the green (535 nm) LED might be the preferred choice. However, there is the option to drive the red (660 nm) as well as the IR (940 nm) LED. The IR-LED might be of advantage as its light is invisible to the human eye. This can be a key criteria as in dark environments the green or red glow - if not shielded properly – might distract the user. In addition, the 940 nm IRLED features the lowest forward voltage (slightly lower than the 660 nm LED).

### 3.1.4 INTERFACING THE LED's

In a pure heart rate monitoring system it is sufficient to drive only one LED. In order to compensate for ambient light and to save power the LED can be operated in pulsed mode. During the LED off time the ambient light can be measured and subtracted from the signal obtained during LED on time.

The LED pulse repetition rate and pulse width is a tradeoff between signal quality (AC) and overall power consumption. Usual systems sample with rates between 25 and 500 Hz per channel with a pulse width ranging from 500 µs down to 5 µs.

In a pulse oximetry application, the red and infrared LEDs are driven alternatively. Between the red and infrared light pulse, an ambient light measurement can be performed as well. In terms of sampling characteristics the same consideration apply as for heart rate monitoring. However, a low noise level is more critical here due to the different use of the signal.

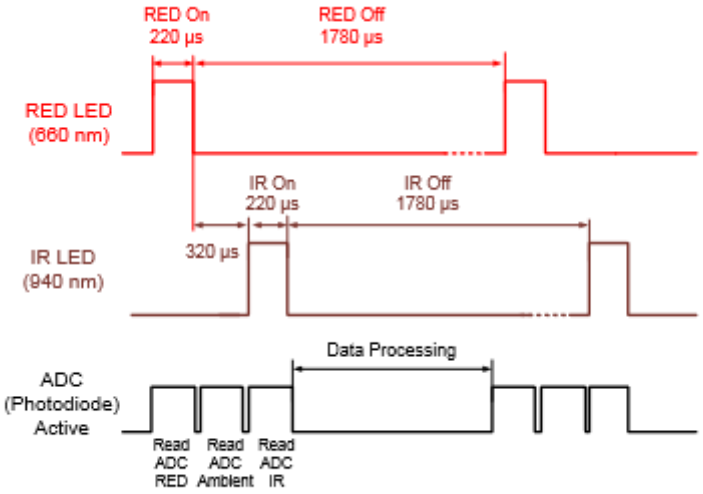



***Fig 3.2*** *A typical timing diagram*

### 3.1.5 INTERFACING THE PHOTODIODE

There are numerous interface options to connect the photodiode of the SFH 7050 to an analog-to-digital converter (e.g. microcontroller) for further signal processing. The most prominent is the use of a transimpedance amplifier (TIA) followed by a gain stage with filtering before analog-to digital conversion.
The gain (i.e. feedback resistor value) of the TIA stage should be set as large as possible to optimize the SNR. On the other hand a high feedback resistor reduces the available bandwidth.

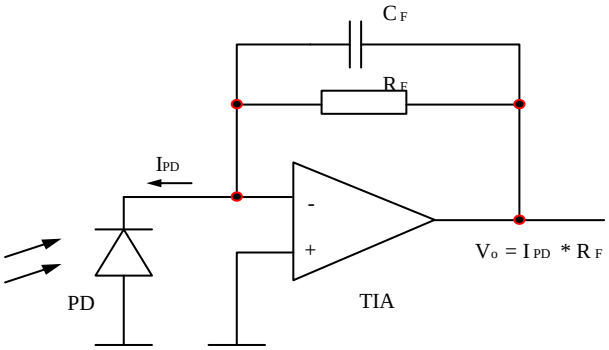



***Fig 3.3****Transimpedance amplifier (TIA) setup for interfacing the photodiode of the SFH 7050.*

The key specifications for the TIA are extremely low input current, input current noise, and input voltage noise, as well as high-voltage operation. These characteristics are necessary to maximize the SNR so that the small currents of interest can be measured amid the large ambient currents from the reflected LED light. High-voltage operation means that a larger feedback resistor can be used to easily amplify the ambient and received LED originated current before removing the reflective DC and ambient light portion with a high-pass filter (after the sample & hold circuit). The remaining small signal of interest is then amplified to maximize the use of ADC's dynamic range. This gain stage should be programmable to compensate for changing environmental factors and the aging of optical components. The key specifications for the ADC

are high resolution, SNR and short acquisition time. The acquisition time should be short enough to capture the modulated signal with the required resolution.

Other hardware realizations include e.g. differential current sensing TIA configuration and so called zeroing circuits to remove most of the ambient light contribution (DC component). As a general design rule the SFH 7050 should be placed as close as possible to the TIA and the PCB tracks should be kept away from the LED supply lines to minimize any electrical crosstalk (noise). Additionally good electromagnetic shielding is recommended.

Finally chipsets are available which include the complete analog signal processing (incl. LED driver) as well as analog-to-digital conversion (e.g. from Texas Instruments or Analog Devices). The SFH 7050 can be directly connected to these chipsets to enable fast and easy evaluation and design. **[1]**

## 3.2 TRANSIMPEDANCE AMPLIFIER

Photodiode in the Biomon sensor converts optical signals into current signal. These signals are usually small in magnitude (nAto μA) and it is amplified by a transimpedance amplifier (TIA) which converts the photocurrent into a proportional output voltage simultaneously amplifying the signal. It is also known as I-to-V converter. An active transimpedance amplifier is based on active elements like BJTs, FETs, and operational amplifier (Op-amp). Transimpedance amplifier using op-amp is the widely used one. Fig.1 shows the active op-amp based transimpedance amplifier circuit with photodiode. In the circuit shown in Fig.1, +V and –V are the supply voltages for the op-amp. $R_f$ and$C_f$ (necessary to maintain stability) are the feedback resistor and feedback capacitor respectively. $I_{PD}$ the photodiode current. The photodiode is connected between ground and the inverting input of the op-amp. The other input of the op-amp is connected to ground.

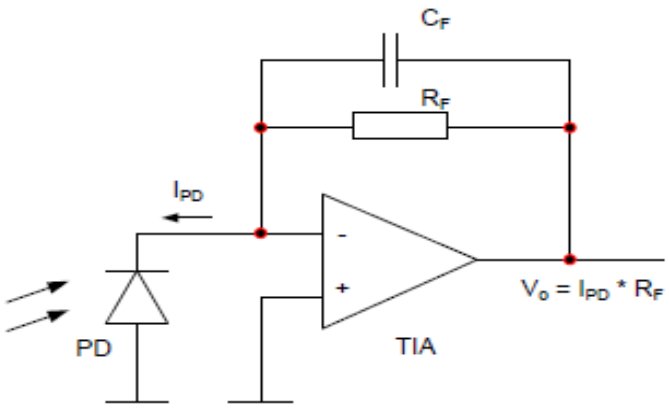



**Figure 3.4** *Transimpedance amplifier circuit*

### 3.2.1 DESIGN REQUIREMENTS

The design requirements for the TIA are:

- Transimpedance gain of at least 10000 *V/A*
- -3 dB bandwidth of at least 5 *MHz*.
- DC coupling of the photodiode to the TIA.
- Guaranteed stability with a photodiode capacitance of up to 10 *pF*.

### 3.2.2 DESIGN METHODOLOGY

**Gain:** The feedback resistor ($R_f$) value is used to define the gain of the TIA .For gain to be very high, feedback resistor is made as large as other constraints permit.The value of the feedback resistor is

$$R_F = \frac{V_{OUT\,(MAX)} - V_{OUT\,(MIN)}}{I_{IN\,(MAX)}}$$

Where $V_{OUT(MAX)}$= maximum output voltage

$V_{OUT(MIN)}$= minimum output voltage

$I_{IN(MAX)}$ = maximum input current of the photodiode

**Bandwidth:** The -3 dB bandwidth of the TIA is inversely proportional to the feedback resistor. Therefore, if the bandwidth is important then, small feedback resistor is used.

**Stability:** To stabilize the transimpedance amplifier, a large enough capacitor must be placed in parallel with the feedback resistor. The calculated value of feedback capacitor is

$$C_F = \sqrt{\frac{C_T}{2\pi R_F f_{GBW}}}$$

$C_T$= total capacitance of the photodiode and op-amp

$F_{GBW}$= gain-bandwidth product of the op-amp

$R_{F=}$ the feedback resistor

**Noise:** It is essential to take into account various noise sources (internal and external). Internal noise sources within the op-amp and external noise which the TIA may pick up such as power supply noise

### 3.2.3 SELECTION OF OPERATIONAL AMPLIFIER

The LMV793MF (Texas Instruments), is selected for this design because of its excellent combination of low bias current, offset voltage, power consumption and wide bandwidth.

#### 3.2.3.1Description

The LMV793 input operational amplifiers offer a low input voltage noise density of 5.8 nV/√Hz while consuming only 1.15 mA (LMV793) of quiescent current. The LMV793 is stable at a gain of 10 and have a gain bandwidth product (GBW) of 88 MHz. The LMV793 have a supply voltage range of 1.8V to 5.5V and can operate from a single supply. The LMV793/LMV794 each feature a rail-to-rail output stage capable of driving a 600Ω load and sourcing as much as 60 mA of current. The LMV793 provide optimal performance in low voltage and low noise systems. A CMOS input stage, with typical input bias currents in the range of a few femto-Amperes, and an input common mode voltage range, which includes ground, make the LMV793 ideal for low power sensor applications where high speeds are needed. The LMV793 is manufactured using TI's advanced VIP50 process. The LMV793 is offered in a 5-Pin SOT23package.

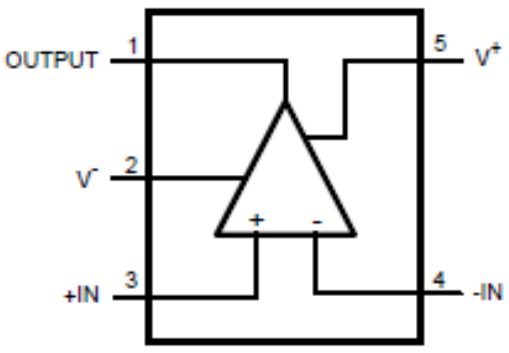



**Fig 3.5** *Diagram of LMV793*

#### 3.2.3.2 Features

- Input Referred Voltage Noise 5.8 nV/√Hz
- Input Bias Current 100 uA
- Gain Bandwidth Product 88 MHz
- Supply Current per Channel– LMV793 1.15 mA.

– LMV794 1.30 mA

- Rail-to-Rail Output Swing

– @ 10 kΩ Load 25 mV from Rail

– @ 2 kΩ Load 45 mV from Rail

- Ensured 2.5V and 5.0V Performance stage,
- Total Harmonic Distortion 0.04% @1 kHz, 600Ω
- Temperature Range −40°C to 125°C

#### 3.2.3.3 Calculation

$V_{OUT(MAX)}$ =4 V

$V_{OUT(MIN)}$ =0 V

$I_{IN(MAX)}$ =0.4uA

$C_T=C_{PD}$ + $C_{OPAMP}$=15pF+15pF=30pF

Gain Bandwidth product, $F_{GBW}$=88Mhz

Substituting these values on equations, the values are calculated as

$R_{f=}$10MΩ $C_{F=}$ 2.2pF **[8]**

## 3.3 FILTER DESIGN

### 3.3.1 BAND-PASS FILTER

The high pass and low pass sections work together as a Second order Butterworth Band pass filter. A band pass filter is used as a frequency selector. It allows one particular band of frequencies to pass. Thus the pass band is between the two cut off frequencies fL and fH i.e. between 0.2Hz and 8Hz (in this case). Any frequency outside this band gets attenuated. The pass band which is between fH and fL is called bandwidth of the filter denoted as BW.

BW = fH – fL

#### 3.3.1.1 Low Pass Filter

A low-pass filter passes low-frequency signals while attenuating (reducing the amplitude of) signals with frequencies higher than the cut-off frequency. The actual amount of attenuation

for each frequency varies from filter to filter. It is sometimes called a high-cut filter. The cut off here is set to 8Hz.

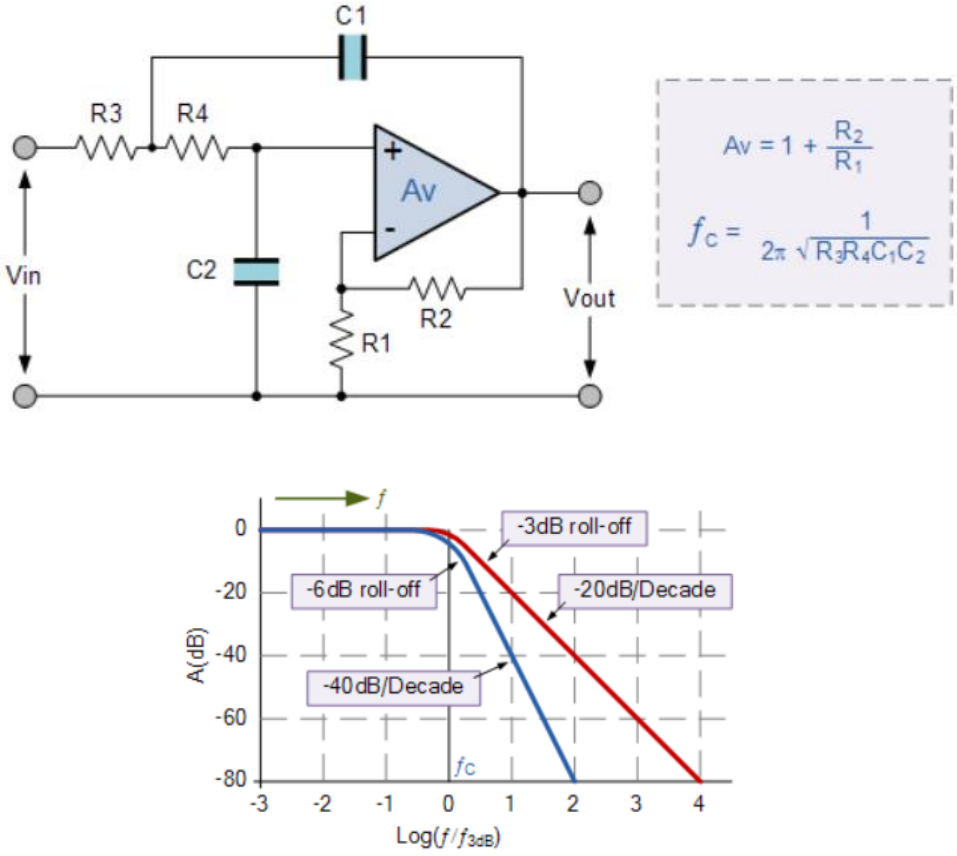
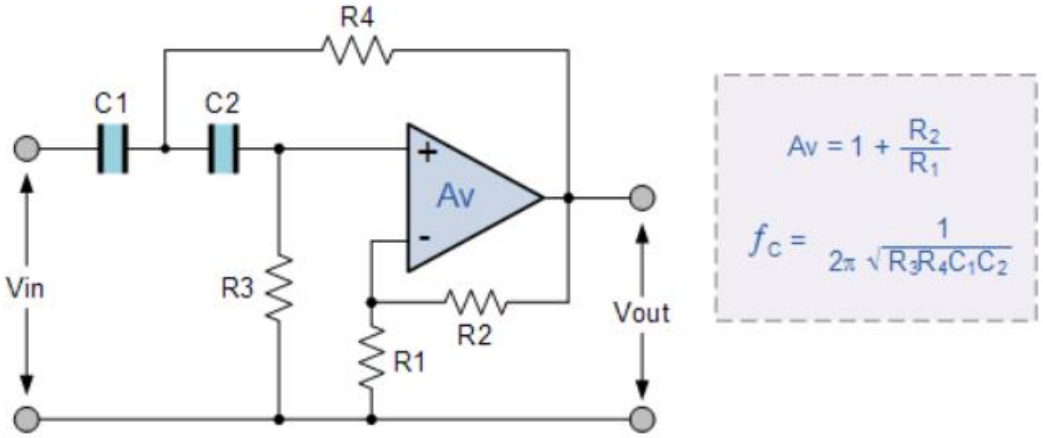



**Fig 3.6** *Active 2nd order low pass filter*

### 3.3.1.2 High Pass Filter

A high-pass filter, or HPF, is a Linear Time Invariant Filter that passes high frequencies but attenuates frequencies lower than the filter's cut off frequency. The actual amount of attenuation for each frequency is a design parameter of the filter. It is sometimes called a low-cut filter or bass-cut filter. The cut off here is set to 0.2Hz.



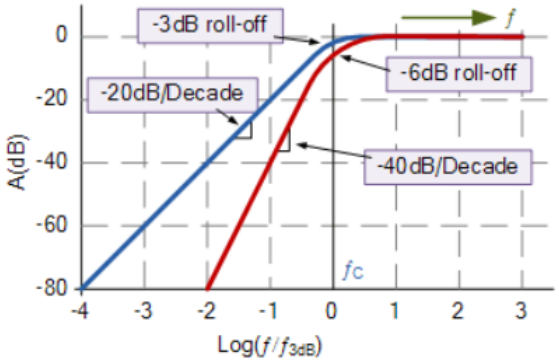



**Fig 3.7** *Active 2nd order high pass filter*

### 3.3.2 NOTCH FILTER

The twin T notch provides a large degree of rejection at a particular desired frequency. The notch filter is useful in rejecting unwanted signals that have a particular frequency. One of the 27 applications is to zero out unwanted mains noise at 50 or 60 Hz that may be entering a circuit. Theoretically, at the notch frequency the level of attenuation provided by the twin T notch filter is infinite. Similar to other RC circuits, the RC twin T notch filter circuit also has what may be termed as a soft cut-off. Practically, the response of the notch falls away slowly and affects a wide band of frequencies on either side of the cut-off frequency. The cut-off for this circuit is 50Hz power interference which is considered to be a noise in the PPG signal. The band selected is 45 to 75 Hz.

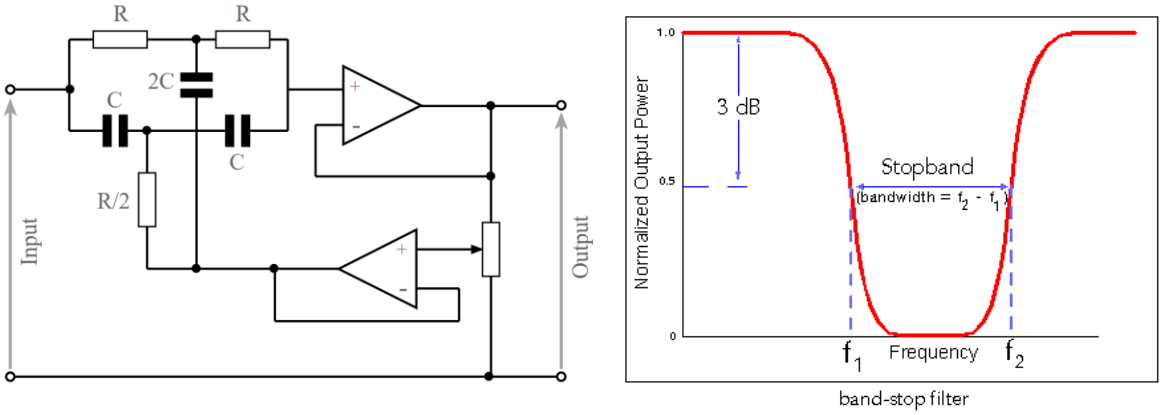



**Fig 3.8** *Active twin t notch filter*

## 3.4 PCB DESIGN

A printed circuit board (PCB) mechanically supports and electrically connects electronic components or electrical components using conductive tracks, pads and other features etched from one or more sheet layers of copper laminated onto and/or between sheet layers of a non-conductive substrate. The sensor SFH7050 is with pins under pads. Hence the PCB design becomes the integral part of the hardware implementation for the purpose of testing the sensor.

The Circuitmaker (Altium) is used for PCB designing because of its robustness and simplicity. Circuitmaker implements schematic capture and PCB design using the same engine as the

Altium Designer, providing an identical user experience. The schematic editor includes basic component placement and circuit design as well as advanced multi-channel design and hierarchical schematics. All schematics are uploaded to the altium server and can be viewed by anyone with a Circuitmaker account, stimulating design re-use. Circuitmaker supports integration with the the Octopart search engine and allows drag and drop placements of components from the Octopart search results if schematic models are attached to them. Users can build missing schematic symbols and commit them to the server called the Community Vault, making them available for other users. The continuously growing part database eliminates the need for custom schematic symbol or footprint design for common parts, increasing user friendliness for the beginners. The PCB design consisted of double layer with provision for entire circuit in the single board. The bottom layer consist of power supply tracks for all the components used

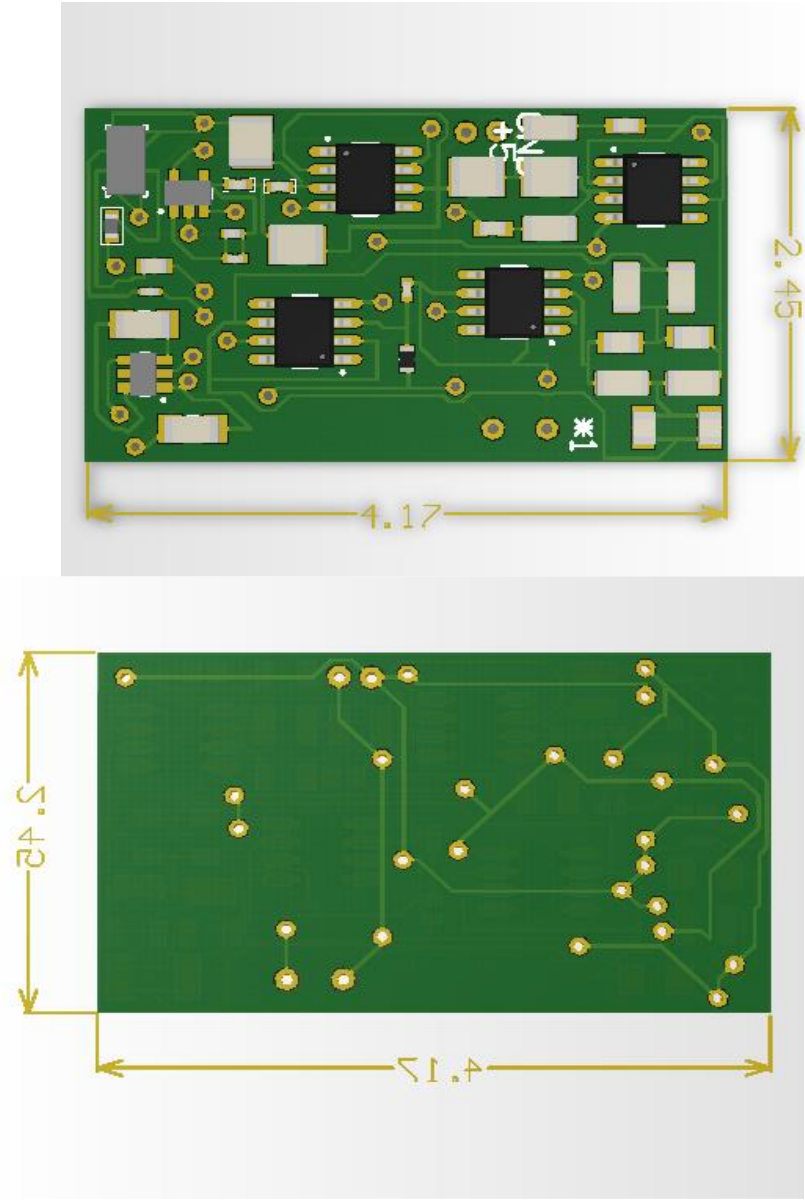



**Fig 3.9** *Top and Bottom view of the PCB*

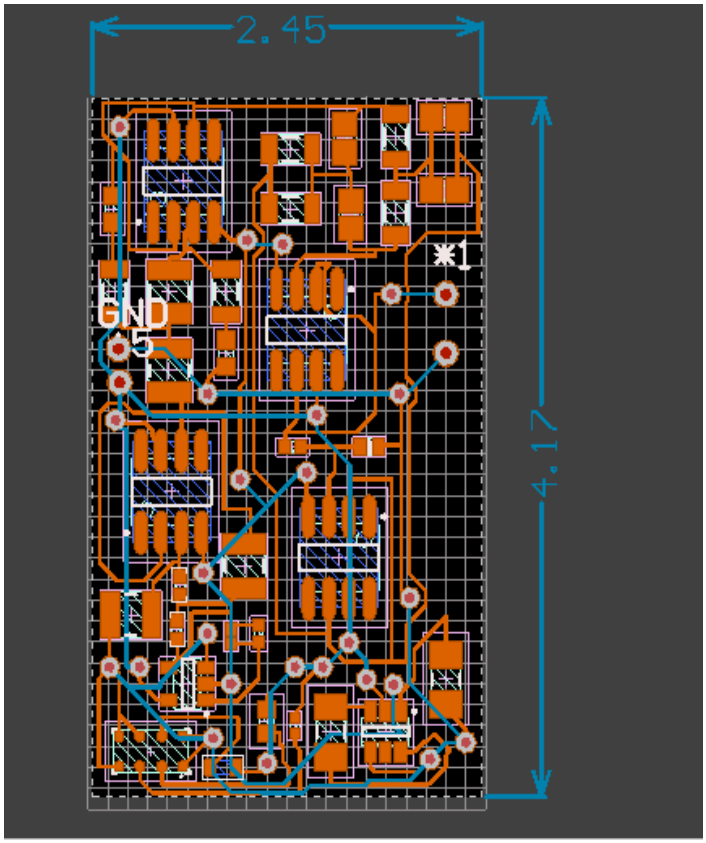



**Fig 3.10** *Overall PCB board*

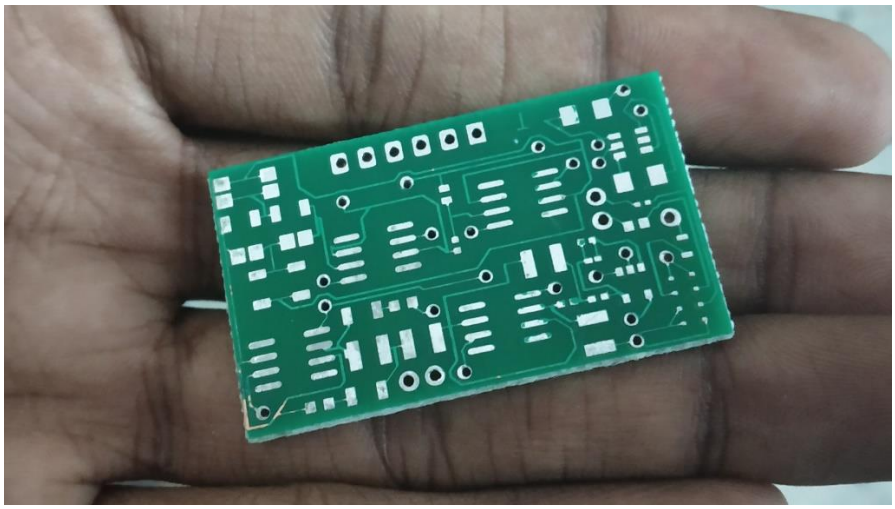


**Fig 3.11** *Final product- Printed board*

## 3.5 SOLDERING PROCESS

The next step in hardware implementation is soldering. The SFH7050 sensor has to be soldered by the special process called reflow soldering since the pads are underneath. The soldering profile of the sensor is given as:

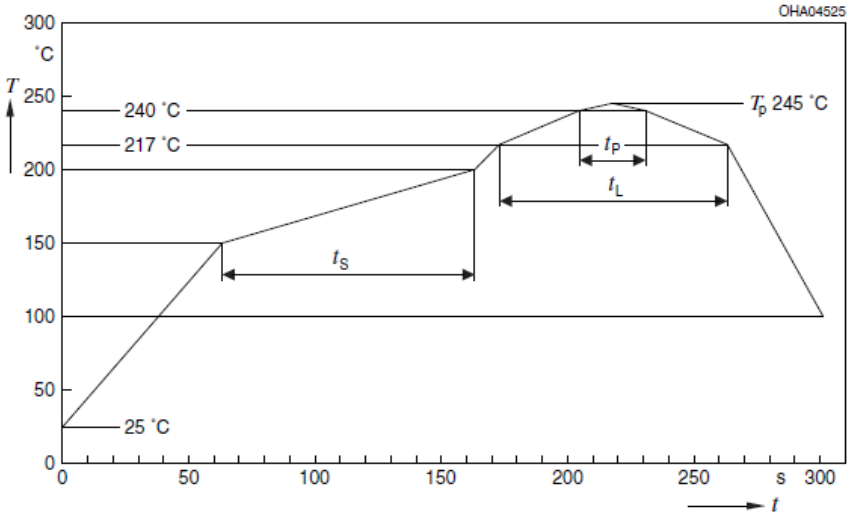



**Fig 3.12** *Reflow soldering profile* **[1]**

Reflow soldering is a process in which a solder paste (a sticky mixture of powdered solder and flux) is used to temporarily attach one or several electrical components to their contact pads, after which the entire assembly is subjected to controlled heat, which melts the solder, permanently connecting the joint. Heating may be accomplished by passing the assembly through a reflow oven or under an infrared lamp or by soldering individual joints with a hot air pencil. **[7]**

**Fig 3.13** *Reflow soldering oven*

## 3.6 ANALOG TO DIGITAL CONVERTOR

The ADS1115 devices (ADS111x) are precision, low-power, 16-bit, I 2Ccompatible, analog-to-digital converters (ADCs) offered in an ultra-small, leadless, X2QFN-10 package, and a VSSOP-10 package. The ADS111x devices incorporate a low-drift voltage reference and an oscillator. The ADS1114 and ADS1115 also incorporate a programmable gain amplifier (PGA) and a digital comparator. The PGA can be used to measure with improved resolution and control amplified input to the microcontroller. These features, along with a wide operating

supply range, make the ADS111x well suited for power- and space-constrained, sensor measurement applications. The ADS111x perform conversions at data rates up to 860 samples per second (SPS). The PGA offers input ranges from ±256 mV to ±6.144 V, allowing precise and small signal measurements.

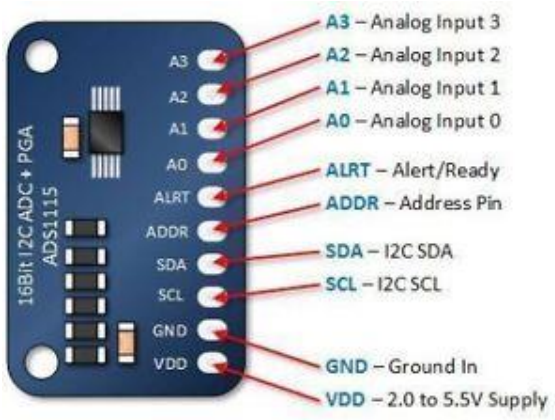



**Fig 3.14** *Adafruit ADS1115*

### 3.6.1 FEATURES

• Ultra-Small X2QFN Package: 2 mm × 1.5 mm × 0.4 mm
• Wide Supply Range: 2.0 V to 5.5 V
• Low Current Consumption: 150 µA (Continuous-Conversion Mode)
• Programmable Data Rate: 8 SPS to 860 SPS
• Single-Cycle Settling • Internal Low-Drift Voltage Reference
• Internal Oscillator • I 2C Interface: Four Pin-Selectable Addresses
• Four Single-Ended or Two Differential Inputs (ADS1115)
• Programmable Comparator (ADS1114 and ADS1115)
• Operating Temperature Range: –40°C to +125°C

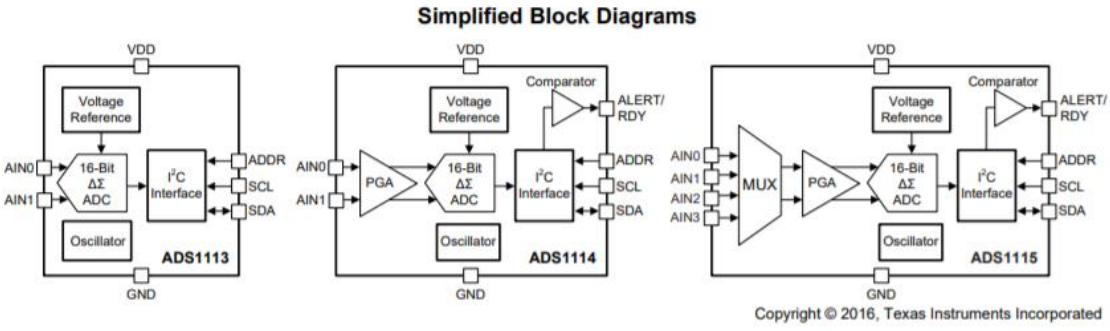



**Fig 3.15** *Block diagram of ADS1115*

## 3.7 SPECTRUM ANALYSIS

This was done to ensure that all the LEDs on the sensor are in fine condition and gave response according to what was given in the data sheet. The experiment was done using an optical spectrometer. The light of the LED was passed through a collimator which makes the input light parallel and was fed to the spectrometer.

### 3.7.1 SPECTROMETER

A spectrometer is a scientific instrument used to separate and measure spectral components of a physical phenomenon. Spectrometer is a broad term often used to describe instruments that measure a continuous variable of a phenomenon where the spectral components are somehow mixed. In visible light a spectrometer can for instance separate white light and measure individual narrow bands of color, called a spectrum, while a mass spectrometer measures the spectrum of the masses of the atoms or molecules present in a gas. The first spectrometers were used to split light into an array of separate colors. Spectrometers were developed in early studies of physics, astronomy, and chemistry. **[11]**

### 3.7.2 COLLIMATOR

A collimator is a device that narrows a beam of particles or waves. To narrow can mean either to cause the directions of motion to become more aligned in a specific direction (i.e., make collimated light or parallel rays), or to cause the spatial cross section of the beam to become smaller (beam limiting device). **[12]**

# 4. OBSERVATIONS, SIMULATIONS AND RESULTS

## 4.1 CONTROL CIRCUIT

### 4.1.1 TRANDIMPEDANCE AMPLIFIER

All the circuit simulations done are on LTspice. LTspice is freeware computer software implementing a SPICE simulator of electronic circuits, produced by semiconductor manufacturer Linear Technology, now part of Analog Devices. The AC analysis is shown below.

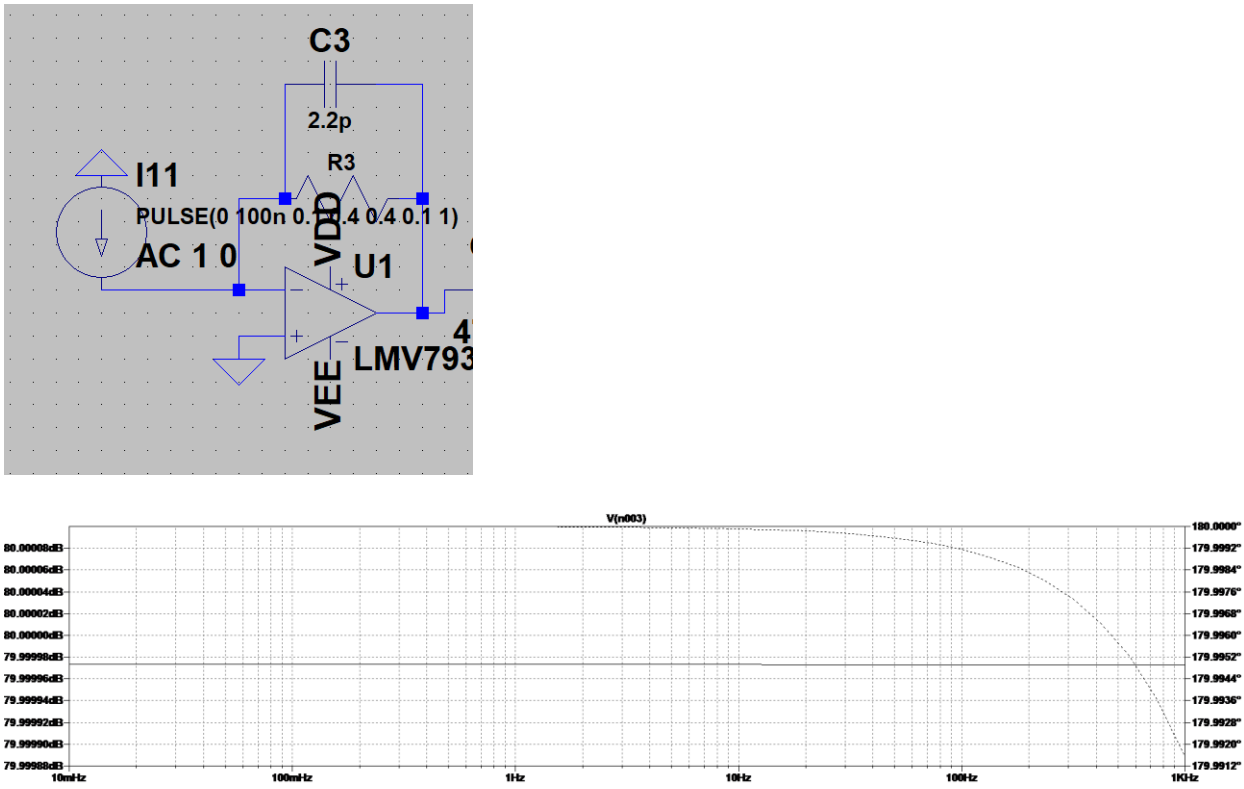
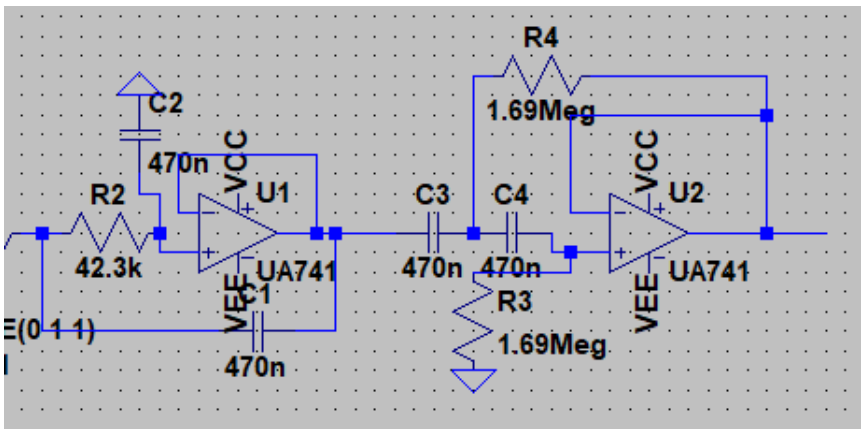



**Fig 4.1** *TIA on LTspice (x axis-frequency; y axis-Gain)*

### 4.1.2 BANDPASS FILTER



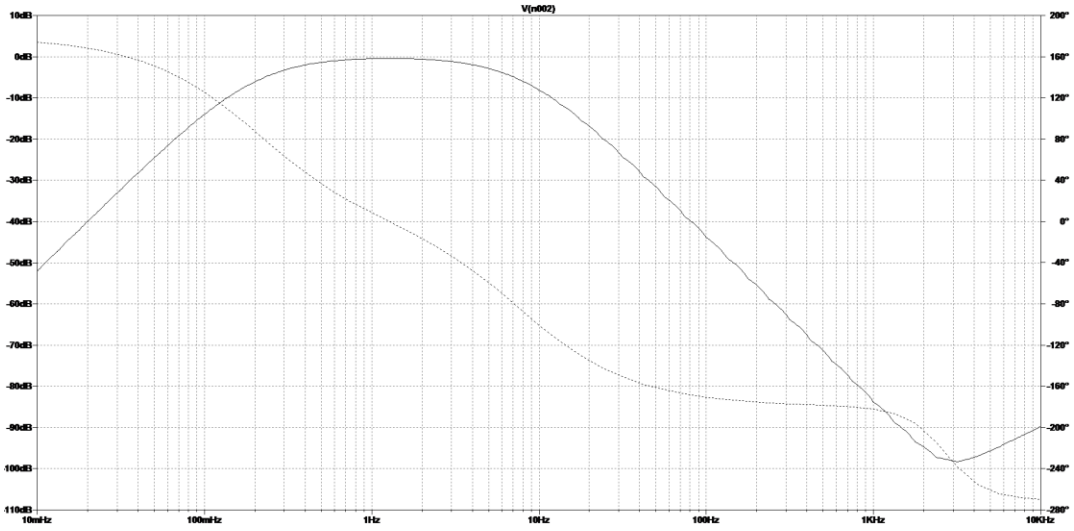



**Fig 4.2** *2nd order Butterworth bandpass filter on LTspice* **[3][4]**

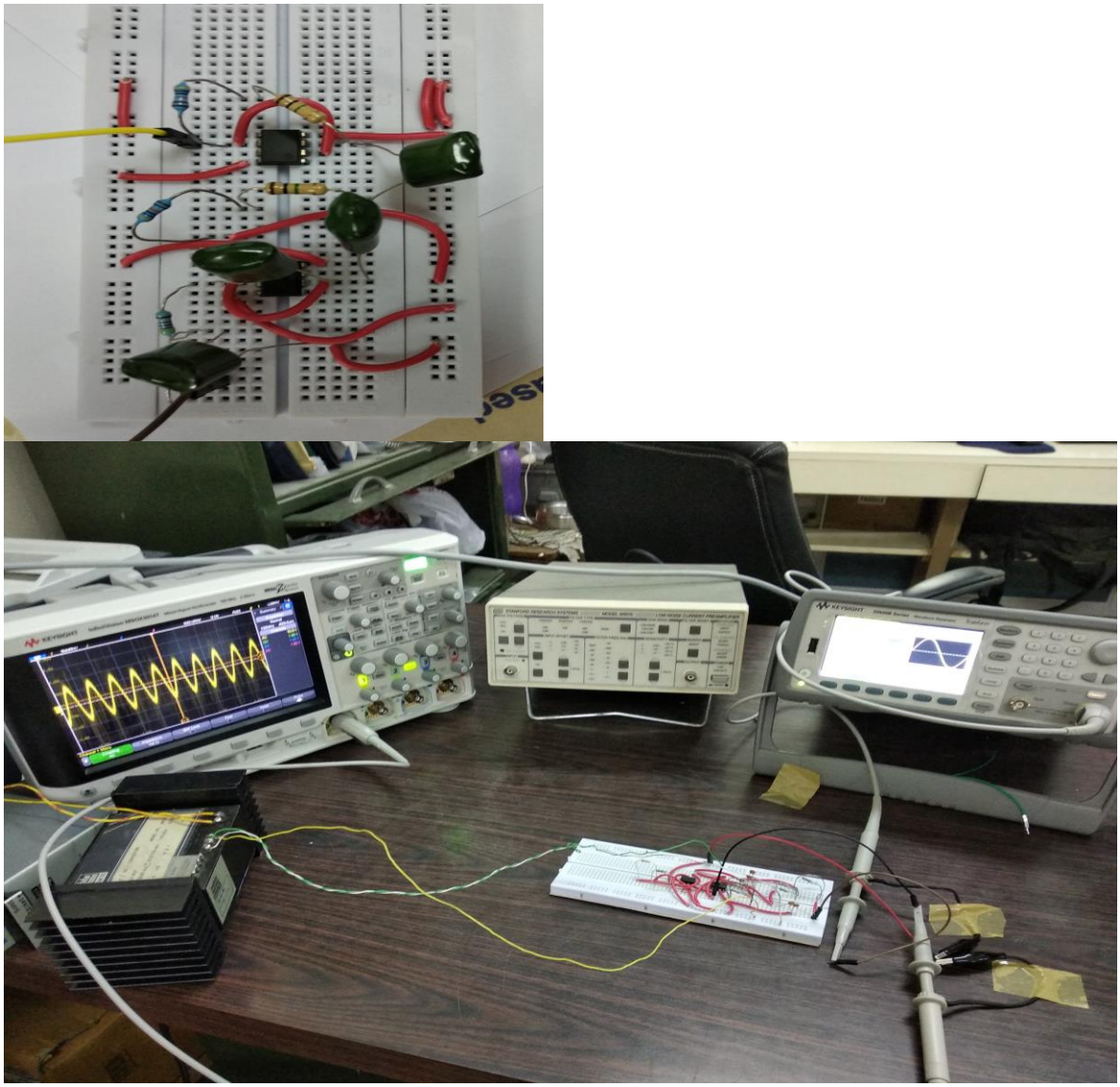
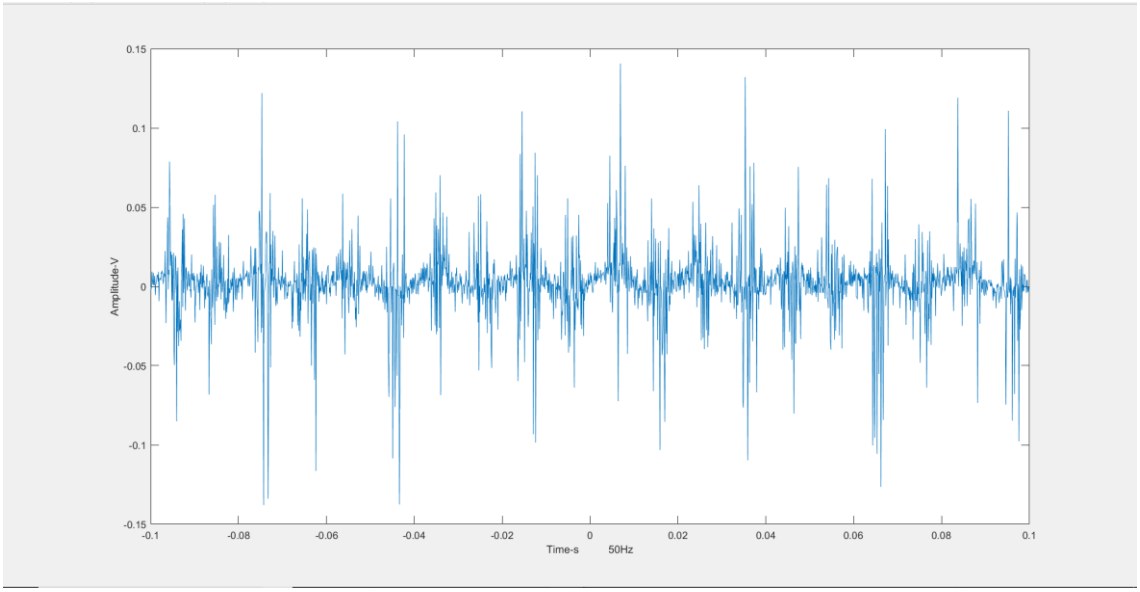


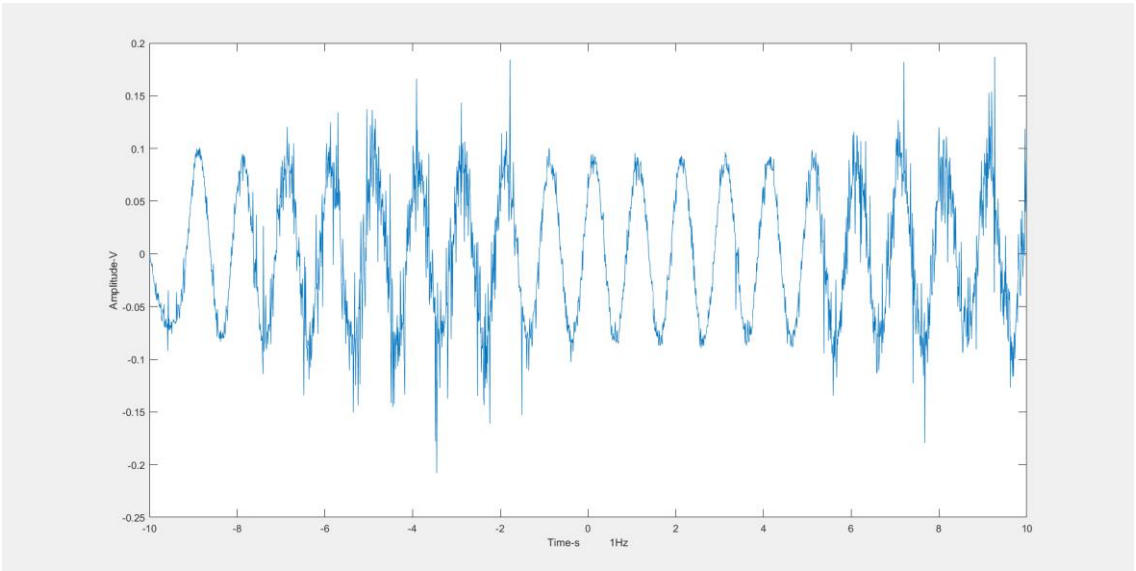
**Fig 4.3** *setup and testing of the filter prototype*

The setup seen above recorded the observations below.

The input given to the filter were sinusoidal waves of varied frequencies and same peak to peak voltage.

It can be observed that the voltage at 0.2Hz and 50 Hz are both attenuated compared to the one at 1Hz.



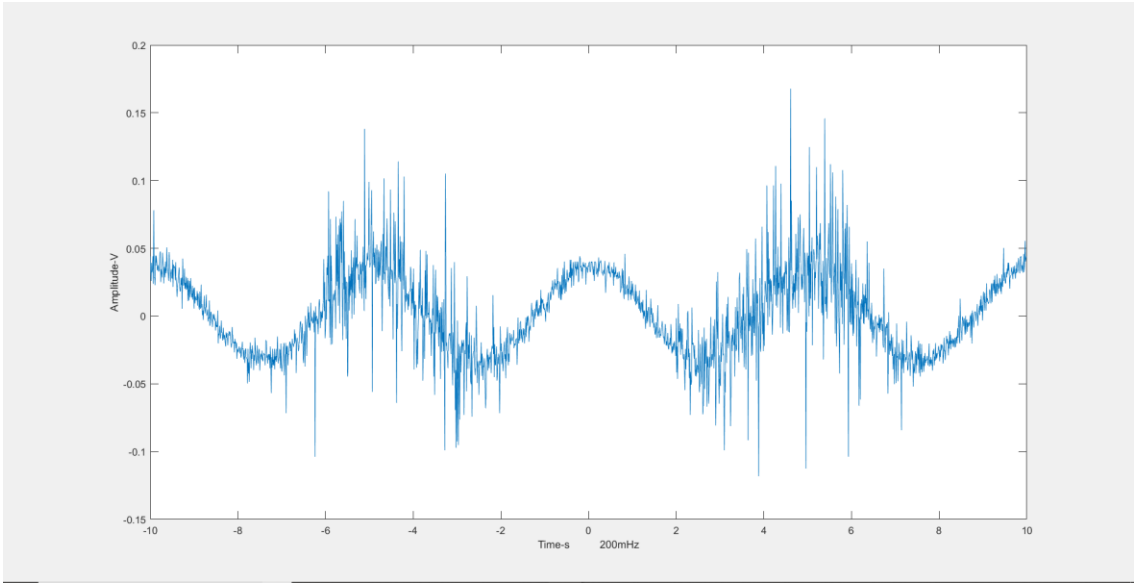



**Fig 4.4** *Time response of Bandpass filter at varied frequencies*

### 4.1.3 OSRAM SENSOR

This was as experiment done in which we gave input currents to the OSRAM LED's and came up with PPG signals on the oscilloscope. The signal was then digitally filtered on MATLAB (a Kaiser window filter with band 0.5 to 8 Hz).

```
D3= design.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',0.1,0.6,8,8.5,40,1,40,sampling_freq);
filt_eq_1 = design(D3,'kaiserwin');
filt3 = filt_eq_1.Numerator;
filt_coeff_ppg=filt3;
filtereddata=filtfilt( filt_coeff_ppg,1,data_to_be_filtered)
```

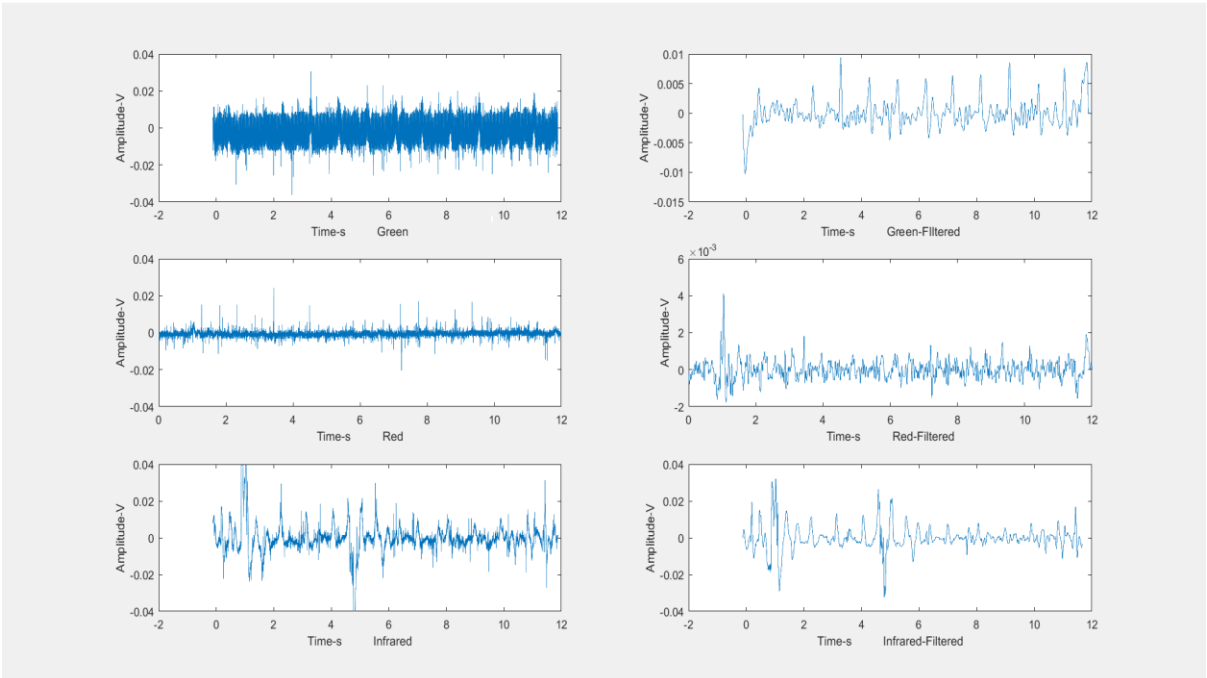
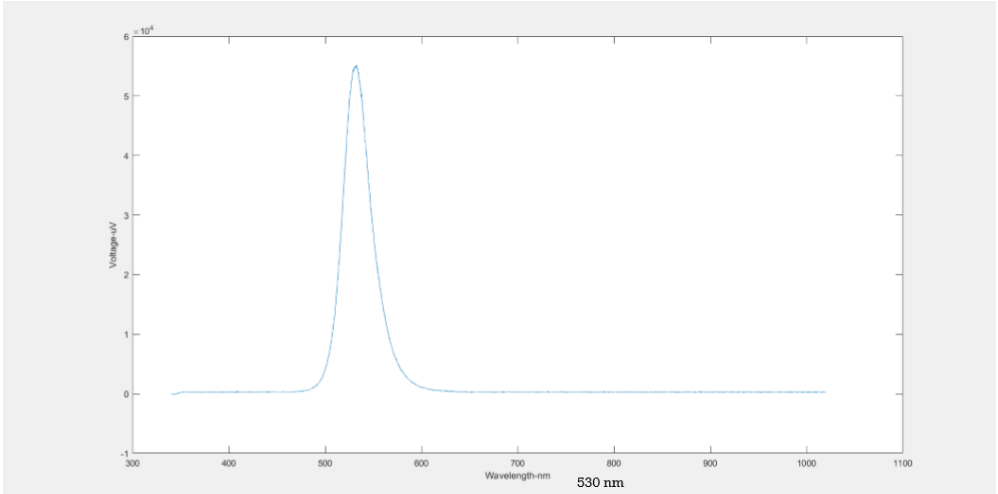


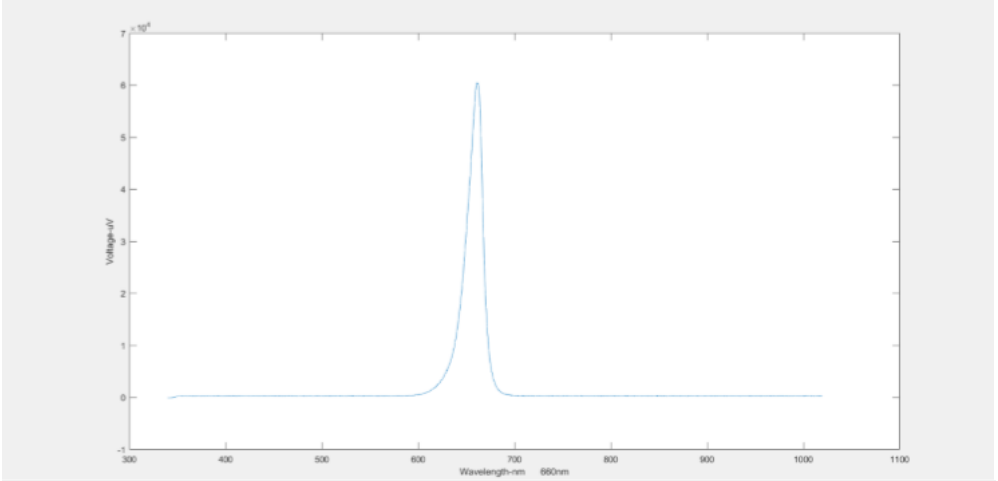

**Fig 4.5** *Time response of PPG signals from OSRAM SFH7050*

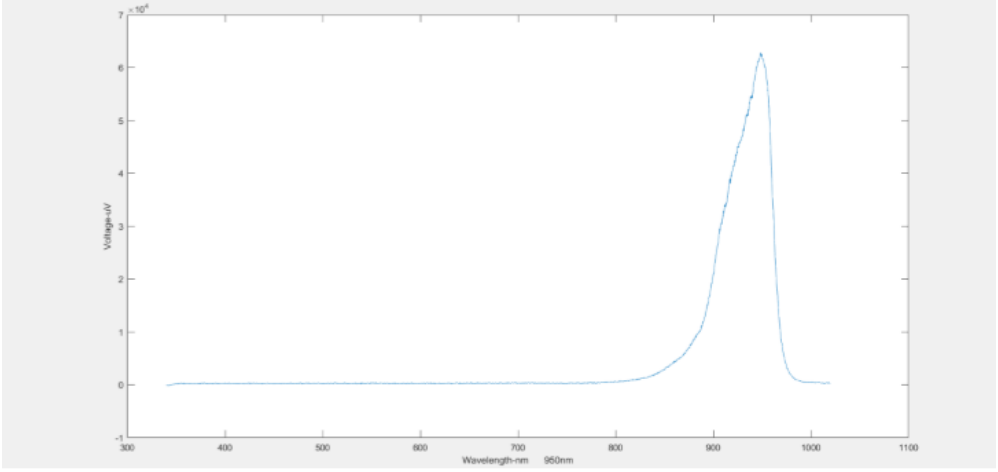
Also to make sure that the LED's were functioning properly a spectral analysis was done and a comparison was made with the one provided in the datasheet and both appeared to be the same. The spectral analysis involved grabbing light from the sensor through a collimator and passing that through to a spectrum analyzer which ranged from 300 to 1100 nm.

The observed spectrum of light for each of the three LED's is given below.




530 nm



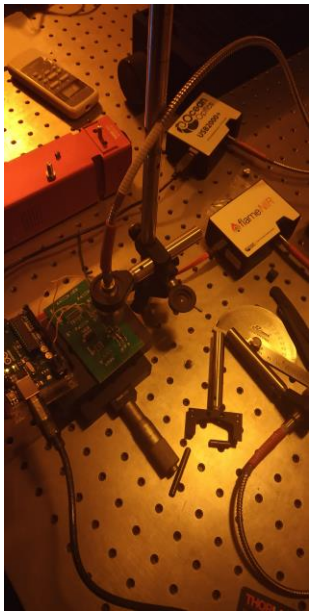
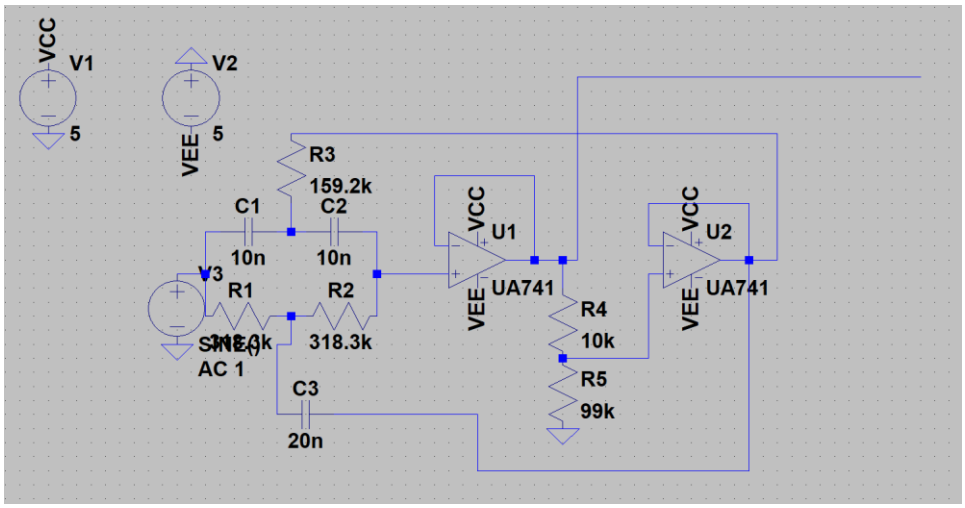


**Fig 4.6** *Spectrum analysis of the sensor's Light Emitting diodes and the experimental setup*

### 4.1.4 NOTCH FILTER



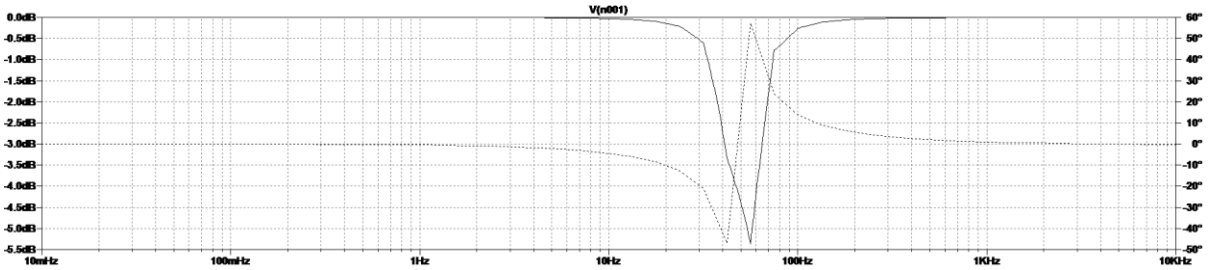




**Fig 4.7** *Active twin t notch filter on LTspice* **[6][7]**

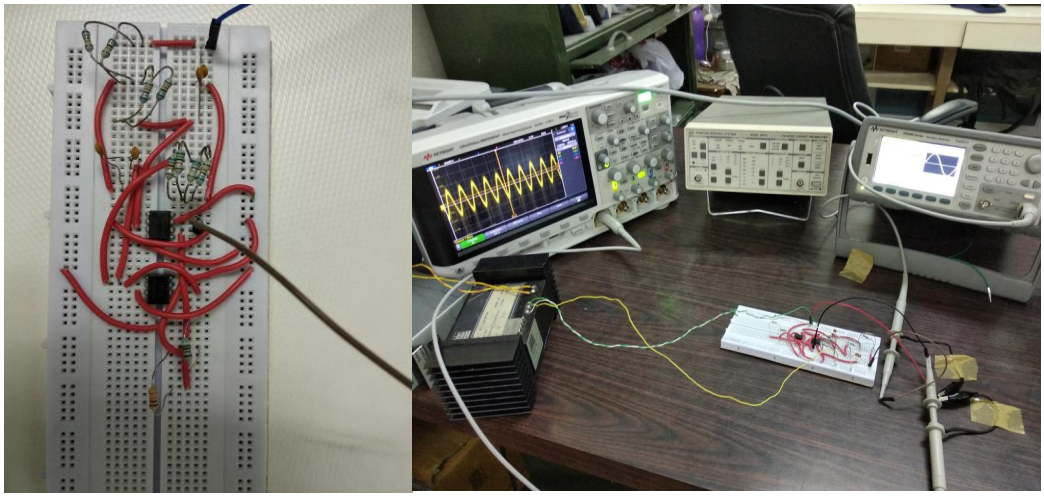


**Fig 4.8** *Notch filter prototyping setup and experiment setup*

The setup seen above recorded the observations below. The input given to the filter were sinusoidal waves of varied frequencies and same peak to peak voltage. It can be observed that the voltage at 50 Hz is attenuated compared to the one at 1Hz. Both the inputs were given with a peak to peak voltage of 3V.

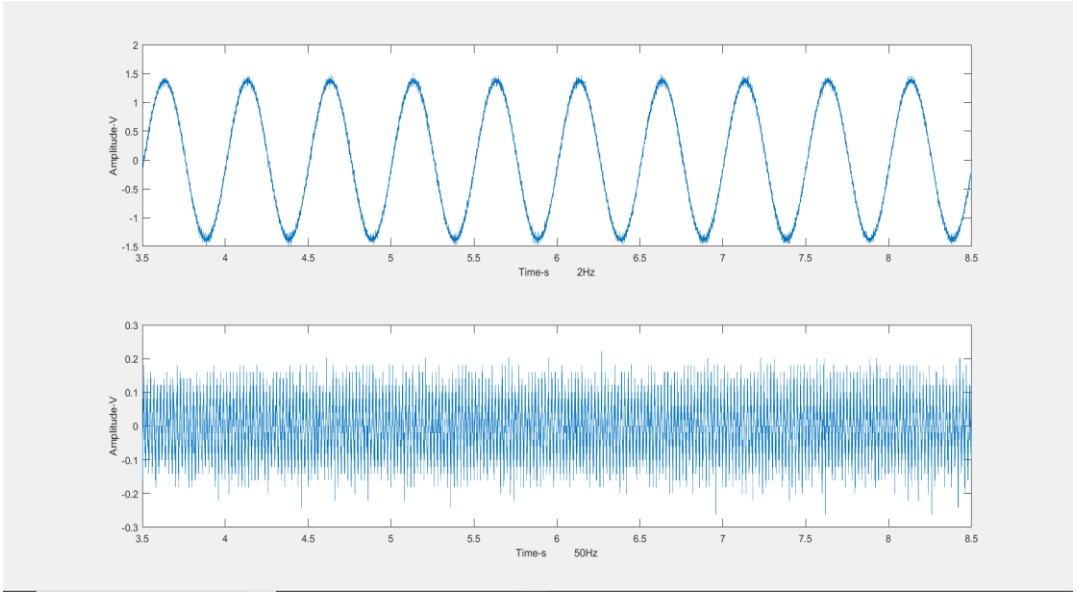



**Fig 4.9** *Time response of notch filter at 2 and 50 Hz*

We have successfully implemented the PCB design and hence completed the hardware implementation of the PPG detection wearable. Testing of the circuit is under the progress. This is the first step of the wide vision of creating a wearable application for heart monitoring effectively and efficiently. There is scope to improve the existing idea to a closer reality. In the future the PPG waveform can be obtained with more accuracy by increasing sampling rate of the system. The existing system can be integrated with a mobile application and integrated with a wearable. Also a software could be developed to obtain and analyze all the measured parameters with the references and detect abnormalities and communicate automatically to detect the fatigue in a driver.

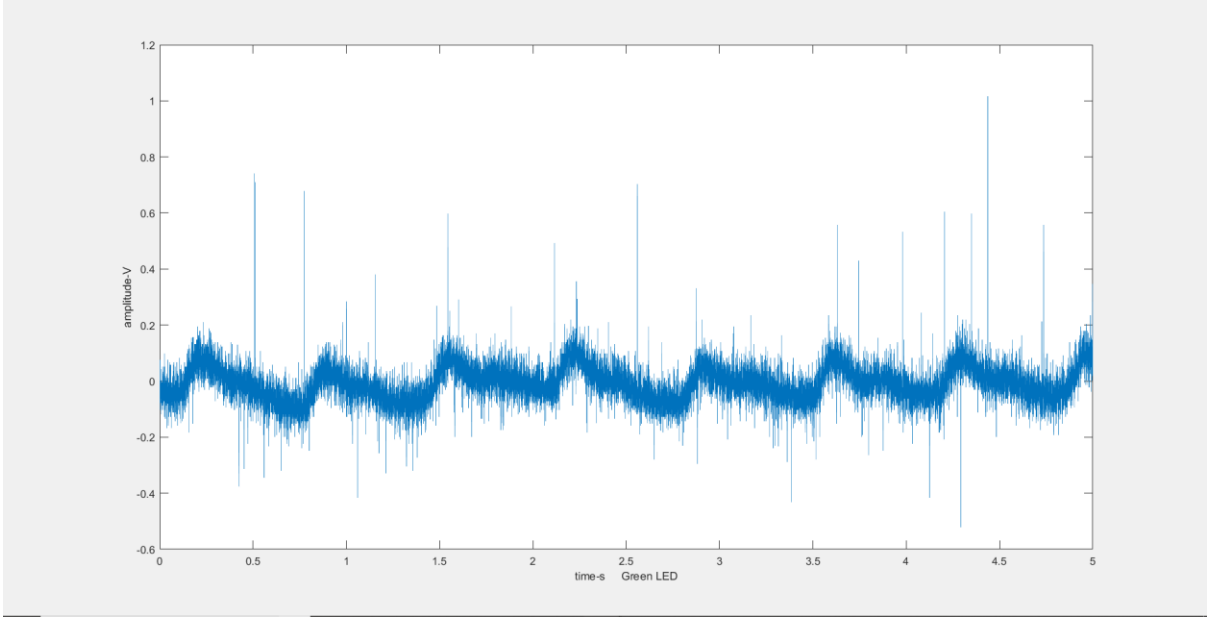



**Fig 4.10** *Signal from the OSRAM after notching at 50Hz*

## 4.2 PROTOCENTRAL PPG BOARD

The device used for testing was the Protocentral pulse oximetry and heart rate monitoring system based on the MAX30100 board. Measuring 14 mm wide and 40 mm long, this board is easily wearable on the finger to measure the pulse of blood.

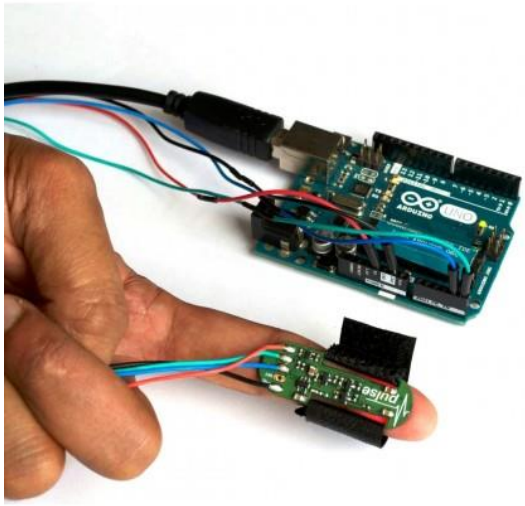


**Fig 4.11** *Protocentral board*

The sensor board was connected to the Arduino according to the given connection and the processing software was used to plot the PPG signal and calculate SpO2 values. Processing is a data visualization software, in existence since 2001, used by artists and scientists alike. It's an open source coding framework based on Java.

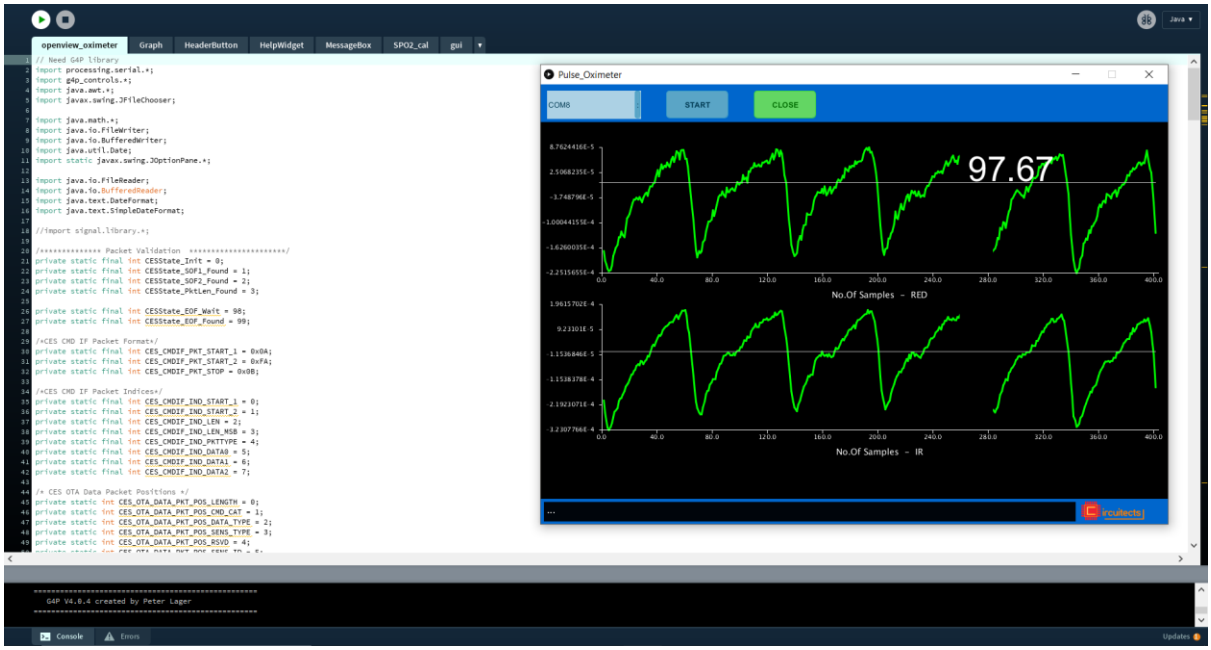



**Fig 4.12** *GUI and data from the sensor, shown on 'processing'*

# 5. PROBLEMS FACED

## 5.1 STRAY CAPACITANCE

When building an experiment on a simple prototyping breadboard, there is always stray capacitance between adjacent rows of connection points to the circuit. This is because the way the solder-less breadboard is built, it has rows of metal connection strips laid side by side separated by plastic splitters. Because the strips are fairly long and they are in parallel, they have a significant capacitance between them

Because of this, every time the prototyping was done the stray capacitance kept adding to the ones used thus rendering the filter obsolete. Since even a small change in the capacitance could shift the cutoff points or the notching point, thus the prototyping was full of errors. Given below are the pictures of the capacitance measured individually and measured on the breadboard:

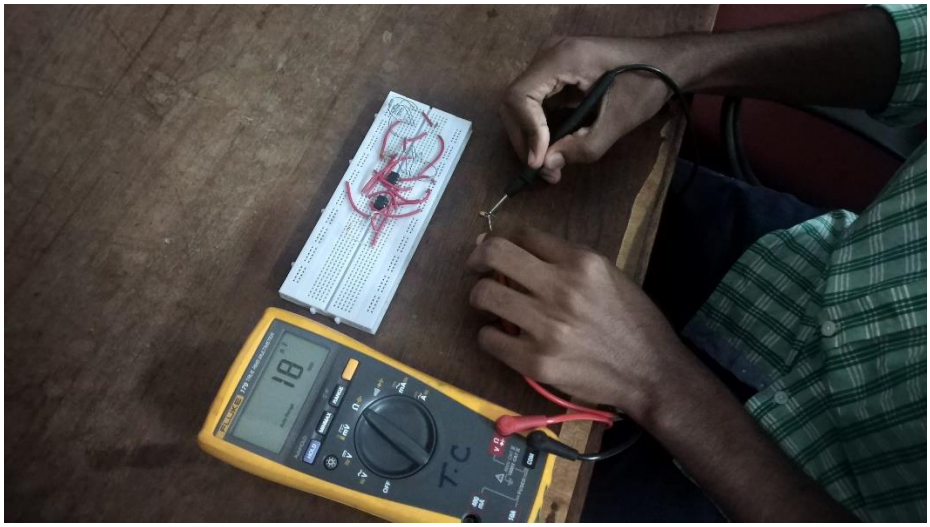


**Fig 5.1** *Measuring the capacitance individually*

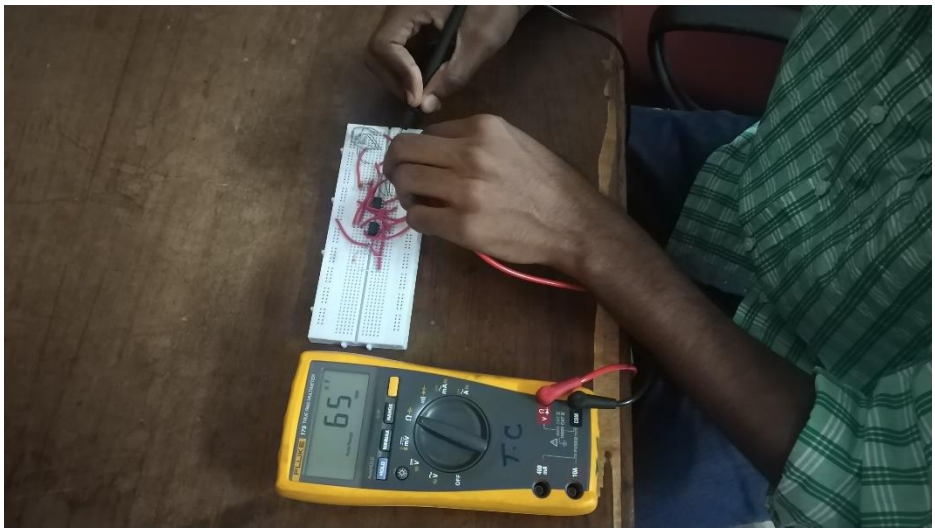


**Fig 5.2** *Measuring the capacitance off the breadboard*

The solution to this problem was to solder all the components on to a general PCB and observe the changes. There was a slight deviation in the measured capacitance and the filter was working better than when it did with the breadboard, more accurate and closer to the simulation.

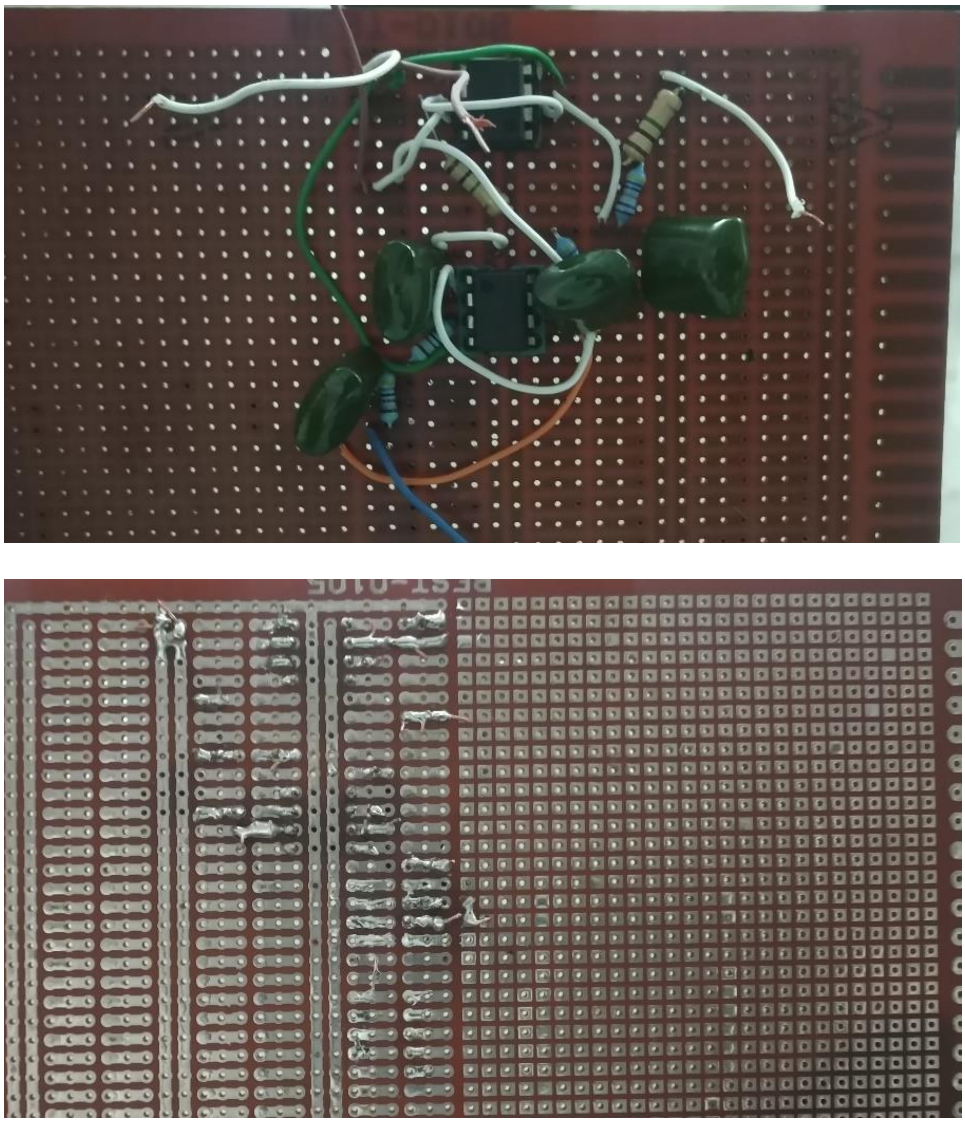


**Fig 5.3** *The general PCB with the notch filter*

## 5.2 GAIN CONTROL

The input to the filters, that is, the output from the transimpedance amplifier had to be controlled. Firstly, because of the limitation on the input voltages to the UA741 op-amps in the filters and secondly, because the ADC or the microcontroller connected to the output has a limited range of measurement.

The solution was to obtain the measured maximum current from the OSRAM Biomon sensor and limit it to 5V. A mirror was placed on top of the LED one by one for each colour and the maximum current was measured by reverse calculating using the gain from the Stanford Research System's transimpedance amplifier and the voltage measured finally. After all calculations, the gain set by us for the TIA was 10,000 V/A.

**Fig 5.4** *Setup of the experiment*

## 5.3 SIZE OF PCB

Initially, the PCB design was flawed, not optimized and extremely big. Since the design is for the purpose of using it on a smart watch it needed to be smaller.

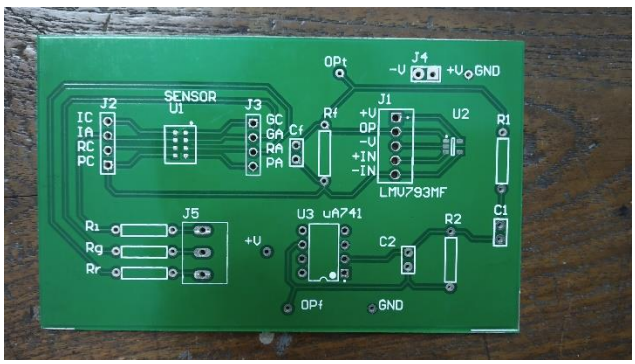



**Fig 5.5** *Initial PCB- 4.5cm*6cm*

The initial PCB design had just the basic TIA and a 1st order bandpass filter. The requirements were, a TIA, a second order bandpass filter, a notch filter and an invertor for the power supply.

We then further optimized the PCB, made it smaller and ended up with a 2.4cm*4 cm design as seen in the report previously.

## 5.4 MOTION ARTIFACT

A PPG sensor can be of transmission or reflection type, consisting of matched LED (source)–photodiode (detector) pair, and depends on the principles of transmission, absorption, and dispersion of light for its operation, as light passes through different absorbing substances such as skin pigmentation, biological tissue, bone, arterial and venous blood. A typical PPG signal contains two components. The first one is a large dc component, due to constant absorption of light when passing through the skin–tissue–bone. The second one is a small ac component at PR, due to the component of light passing through pulsating arteries caused by the heartbeat.

A pulse oximeter needs red and IR PPG signals with clearly separable dc and ac parts for error-free $SpO_2$ estimation. However, PPGs are usually corrupted by motion artifacts (MAs) due to voluntary or involuntary movements of the subject while acquiring the data. Since the pulsatile component is quite small in a PPG (0.1% of total signal amplitude), even a slightest movement of the patient will lead to MA, resulting in inaccurate estimation of $SpO_2$. **[13]**

This problem hasn’t been solved yet.

# 6. CONCLUSION

We have successfully implemented the PCB design and hence completed the hardware implementation of the PPG detection wearable. This is the first step of the wide vision of creating a wearable application for heart monitoring effectively and efficiently. There is scope to improve the existing idea to a closer reality. In the near future, a PPG waveform can be obtained with more accuracy by increasing sampling rate of the system. The existing system can be integrated with a mobile application and integrated with a wearable. Also a software could be developed to obtain and analyze all the measured parameters with the references and detect abnormalities and communicate automatically to detect the fatigue in a driver.

# 7. REFERENCES


1. Data sheet of OSRAM SFH 7050
2. https://www.sciencedirect.com/science/article/pii/S095461111300053X
3. https://www.electronics-tutorials.ws/filter/filter_7.html
4. https://www.electronics-tutorials.ws/filter/second-order-filters.html
5. https://www.electronics-tutorials.ws/filter/band-stop-filter.html
6. https://www.changpuak.ch/electronics/Active_Notch_Filter.php
7. https://en.wikipedia.org/wiki/Reflow_soldering
8. Data sheet of LMV-793
9. Code for Protocentral Board:
https://github.com/Protocentral/Pulse/tree/master/Software/Processing/openview_oximeter
10. https://en.wikipedia.org/wiki/Printed_circuit_board
11. https://en.wikipedia.org/wiki/Spectrometer
12. https://en.wikipedia.org/wiki/Collimator
13. https://ieeexplore.ieee.org/document/6111474/

# 8. APPENDIX

## 8.1 PROCESSING CODE FOR DATA

```
// Need G4P library
import processing.serial.*;
import g4p_controls.*;
import java.awt.*;
import javax.swing.JFileChooser;

import java.math.*;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.util.Date;
import static javax.swing.JOptionPane.*;

import java.io.FileReader;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

//import signal.library.*;

/************** Packet Validation  **********************/
private static final int CESState_Init = 0;
private static final int CESState_SOF1_Found = 1;
private static final int CESState_SOF2_Found = 2;
private static final int CESState_PktLen_Found = 3;

private static final int CESState_EOF_Wait = 98;
private static final int CESState_EOF_Found = 99;

/*CES CMD IF Packet Format*/
private static final int CES_CMDIF_PKT_START_1 = 0x0A;
private static final int CES_CMDIF_PKT_START_2 = 0xFA;
private static final int CES_CMDIF_PKT_STOP = 0x0B;

/*CES CMD IF Packet Indices*/
private static final int CES_CMDIF_IND_START_1 = 0;
private static final int CES_CMDIF_IND_START_2 = 1;
private static final int CES_CMDIF_IND_LEN = 2;
private static final int CES_CMDIF_IND_LEN_MSB = 3;
private static final int CES_CMDIF_IND_PKTTYPE = 4;
private static final int CES_CMDIF_IND_DATA0 = 5;
private static final int CES_CMDIF_IND_DATA1 = 6;
private static final int CES_CMDIF_IND_DATA2 = 7;

/* CES OTA Data Packet Positions */
private static int CES_OTA_DATA_PKT_POS_LENGTH = 0;
private static int CES_OTA_DATA_PKT_POS_CMD_CAT = 1;
private static int CES_OTA_DATA_PKT_POS_DATA_TYPE = 2;
private static int CES_OTA_DATA_PKT_POS_SENS_TYPE = 3;
private static int CES_OTA_DATA_PKT_POS_RSVD = 4;
private static int CES_OTA_DATA_PKT_POS_SENS_ID = 5;
private static int CES_OTA_DATA_PKT_POS_DATA = 6;

private static int CES_OTA_DATA_PKT_OVERHEAD = 6;
private static int CES_CMDIF_PKT_OVERHEAD = 5;

/************** Packet Related Variables **********************/

int ecs_rx_state = 0;
int CES_Pkt_Len;
```

```
int CES_Pkt_Pos_Counter, CES_Pkt_Data_Counter1, CES_Pkt_Data_Counter2;
int CES_Pkt_PktType;
char DataRcvPacket1[] = new char[1000];
char DataRcvPacket2[] = new char[1000];

/********** ControlP5 Related Variables **********/

int colorValue;
HelpWidget helpWidget;
HeaderButton headerButton;
MessageBox msgBox;
SPO2_cal s;
boolean visibility=false;

/********** Graph Related Variables **********/

double maxAxis_ir, minAxis_ir, maxAxis_red, minAxis_red;
double receivedVoltage_RED, receivedVoltage_IR;
BigDecimal min, max;

/********** File Related Variables **********/

boolean logging = false;
FileWriter output;
JFileChooser jFileChooser;
Date date;
FileReader readOutput;
String line;
BufferedWriter bufferedWriter;
DateFormat dateFormat;

/********** Port Related Variables **********/

Serial port = null;
int Ss = -1;
String[] comList;
boolean serialSet;
boolean portSelected = false;
String portName;
char inString = '\0';
String selectedPort;

/********** Logo Related Variables **********/

PImage logo;

/********** General Variables **********/

boolean startPlot, Serialevent = false;
String msgs;
int startTime = 0;

int pSize = 400;
float[] xdata = new float[pSize];
float[] ydata = new float[pSize];
float[] AvgYdata = new float[pSize];
float[] zdata = new float[pSize];
float[] AvgZdata = new float[pSize];
int arrayIndex = 1;
Graph g, g1;
float time =0;
BigDecimal avg, rms, a;
double additionFactor_red, additionFactor_ir;
```

```
float value1, value2;
float RedAC = 0, RedDC = 0, IrAC = 0, IrDC = 0;
//SignalFilter myFilter;

public void setup() {
  size(1000, 700, JAVA2D);
  //fullScreen();
  createGUI();
  customGUI();
  // Place your setup code here

  date = new Date();
  logo = loadImage("logo.png");

  headerButton = new HeaderButton(0, 0, width, 60);
  helpWidget = new HelpWidget(0, height - 30, width, 40);
  msgBox = new MessageBox();
  s = new SPO2_cal();
  g = new Graph(100, 100, width-120, 200);
  g1 = new Graph(100, 350, width-120, 200);
  setChartSettings();
  for (int i=0; i<pSize; i++)
  {
    time = time + 2;
    xdata[i]=time;
    ydata[i] = 0;
    zdata[i] = 0;
  }
  time = 0;
  g.GraphColor = color(0, 255, 0);
  g.Title = "RED";
  g1.GraphColor = color( 0, 255, 0);
  g1.Title = "IR";
 // myFilter = new SignalFilter(this);
}

/************************************************ Draw Function
************************************************************/

public void draw() {
  background(0);
  while (portSelected == true && serialSet == false)
  {
    startSerial(comList);
  }
  background(0);
  if (startPlot)
  {
    g.LineGraph(xdata, ydata);
    g1.LineGraph(xdata, zdata);
  }

  g.DrawAxis();
  g1.DrawAxis();

 // msgBox.MessageBoxAxis(0, height - 100, width, 70);
 // msgBox.draw();
  headerButton.draw();
  helpWidget.draw();
}

/************************************************ Opening Port Function ****************************************
**************/
```

```
void startSerial(String[] theport)
{
  try
  {
    port = new Serial(this, selectedPort, 57600);
    port.clear();
    serialSet = true;
    msgs = "Port "+selectedPort+" is opened Click Start button";
    portName = "\\"+selectedPort+".txt";
  }
  catch(Exception e)
  {
    msgs = "Port "+selectedPort+" is busy";
    showMessageDialog(null, "Port is busy", "Alert", ERROR_MESSAGE);
    System.exit (0);
  }
}

/************************************************  Serial  Port  Event  Function
************************************************************/

void serialEvent (Serial blePort)
{
  Serialevent = true;
  inString = blePort.readChar();
  ecsProcessData(inString);
}

/************************************************  Getting  Packet  Data  Function
************************************************************/

void ecsProcessData(char rxch)
{
  switch(ecs_rx_state)
  {
  case CESState_Init:
    if (rxch==CES_CMDIF_PKT_START_1)
      ecs_rx_state=CESState_SOF1_Found;
    break;

  case CESState_SOF1_Found:
    if (rxch==CES_CMDIF_PKT_START_2)
      ecs_rx_state=CESState_SOF2_Found;
    else
      ecs_rx_state=CESState_Init;
    break;

  case CESState_SOF2_Found:
    ecs_rx_state = CESState_PktLen_Found;
    CES_Pkt_Len = (int) rxch;
    CES_Pkt_Pos_Counter = CES_CMDIF_IND_LEN;
    CES_Pkt_Data_Counter1 = 0;
    CES_Pkt_Data_Counter2 = 0;
    break;

  case CESState_PktLen_Found:
    CES_Pkt_Pos_Counter++;
    if (CES_Pkt_Pos_Counter < CES_CMDIF_PKT_OVERHEAD)  //Read Header
    {
      if (CES_Pkt_Pos_Counter==CES_CMDIF_IND_LEN_MSB)
        CES_Pkt_Len = (int) ((rxch<<8)|CES_Pkt_Len);
      else if (CES_Pkt_Pos_Counter==CES_CMDIF_IND_PKTTYPE)
```

```
      CES_Pkt_PktType = (int) rxch;
    } else if ( (CES_Pkt_Pos_Counter >= CES_CMDIF_PKT_OVERHEAD) && (CES_Pkt_Pos_Counter <
CES_CMDIF_PKT_OVERHEAD+CES_Pkt_Len+1) )  //Read Data
   {
     if (CES_Pkt_PktType == 2)
     {
       if (CES_Pkt_Data_Counter1 < 4)
       {
         DataRcvPacket1[CES_Pkt_Data_Counter1]= (char) (rxch);
         CES_Pkt_Data_Counter1++;
       } else
       {
         DataRcvPacket2[CES_Pkt_Data_Counter2]= (char) (rxch);
         CES_Pkt_Data_Counter2++;
       }
     }
   } else  //All header and data received
   {
     if (rxch==CES_CMDIF_PKT_STOP)
     {
       int data1 = ecsParsePacket(DataRcvPacket1, DataRcvPacket1.length-1);
       int data2 = ecsParsePacket(DataRcvPacket2, DataRcvPacket2.length-1);
       receivedVoltage_RED = data1 * (0.00000057220458984375) ;
       receivedVoltage_IR = data2 * (0.00000057220458984375) ;

       time = time+0.1;
       xdata[arrayIndex] = time;

       //receivedVoltage_RED = myFilter.filterUnitFloat((float)receivedVoltage_RED);
       //receivedVoltage_IR = myFilter.filterUnitFloat((float)receivedVoltage_IR);


       AvgYdata[arrayIndex] = (float)receivedVoltage_RED;
       AvgZdata[arrayIndex] = (float)receivedVoltage_IR;
       value1 = (float)( AvgYdata[arrayIndex] - averageValue(AvgYdata));
       value2 = (float)( AvgZdata[arrayIndex] - averageValue(AvgZdata));
       ydata[arrayIndex] = value1;
       zdata[arrayIndex] = value2;

       float RedDC = (float) averageValue(AvgYdata);
       float IrDC = (float) averageValue(AvgZdata);

       arrayIndex++;
       if (arrayIndex == pSize)
       {
         arrayIndex = 0;
         time = 0;
         RedAC = s.SPO2_Value(ydata);
         IrAC = s.SPO2_Value(zdata);
         float value = (RedAC/abs(RedDC))/(IrAC/abs(IrDC));

         /********  Emprical Formalae  *********/
         //float SpO2 = 10.0002*(value)-52.887*(value) + 26.817*(value) + 98.293;
         //  float SpO2 =((0.81-0.18*(value))/(0.73+0.11*(value)));
         float SpO2=110-25*(value);

         SpO2 = (int)(SpO2 * 100);
         SpO2 = SpO2/100;
         oxygenSaturation.setText(SpO2+"");
       }
       if (startPlot) {
       }
       a = new BigDecimal(averageValue(ydata));
```

```
        avg = a.setScale(5, BigDecimal.ROUND_HALF_EVEN);
        a = new BigDecimal(RMSValue(ydata));
        rms = a.setScale(5, BigDecimal.ROUND_HALF_EVEN);
        a = new BigDecimal(max(ydata));
        max = a.setScale(5, BigDecimal.ROUND_HALF_EVEN);
        a = new BigDecimal(min(ydata));
        min = a.setScale(5, BigDecimal.ROUND_HALF_EVEN);
        msgBox.msg(min, max, avg, rms);

        if (logging == true)
        {
          try {
            date = new Date();
            dateFormat = new SimpleDateFormat("HH:mm:ss");
            output = new FileWriter(jFileChooser.getSelectedFile(), true);
            bufferedWriter = new BufferedWriter(output);
                        //bufferedWriter.write(dateFormat.format(date)+"    :    "    +receivedVoltage_RED+"    ,
"+receivedVoltage_IR);
            bufferedWriter.write(arrayIndex-1+" , "+value1 +" , "+value2);
            bufferedWriter.newLine();
            bufferedWriter.flush();
            bufferedWriter.close();
          }
          catch(IOException e) {
            println("It broke!!!");
            e.printStackTrace();
          }
        }

        maxAxis_red = max(ydata);
        minAxis_red = min(ydata);
        // println(maxAxis_red,minAxis_red);
        maxAxis_ir = max(zdata);
        minAxis_ir = min(zdata);

        if (g.yMax != maxAxis_red)
        {
          g.yMax = (float)(maxAxis_red);
        }
        if (g.yMin != minAxis_red)
        {
          g.yMin = (float)(minAxis_red);
        }

        if (g1.yMax != maxAxis_ir)
        {
          g1.yMax = (float)(maxAxis_ir);
        }
        if (g1.yMin != minAxis_ir)
        {
          g1.yMin = (float)(minAxis_ir);
        }

        ecs_rx_state=CESState_Init;
      } else
      {
        ecs_rx_state=CESState_Init;
      }
    }
    break;

  default:
    break;
```

```
  }
}

/********************************************** Recursion Function To Reverse The data
**********************************************************/

public int ecsParsePacket(char DataRcvPacket[], int n)
{
  if (n == 0)
    return (int) DataRcvPacket[n]<<(n*8);
  else
    return (DataRcvPacket[n]<<(n*8))| ecsParsePacket(DataRcvPacket, n-1);
}

// Use this method to add additional statements
// to customise the GUI controls
public void customGUI() {
  comList = port.list();
  String comList1[] = new String[comList.length+1];
  comList1[0] = "SELECT THE PORT";
  for (int i = 1; i <= comList.length; i++)
  {
    comList1[i] = comList[i-1];
  }
  start.setEnabled(false);
  oxygenSaturation.setVisible(false);
  comList = comList1;
  portList.setItems(comList1, 0);

  oxygenSaturation.setFont(new Font("Arial", Font.PLAIN, 55));
  oxygenSaturation.setLocalColor(2, color(255, 255, 255));

  start.setLocalColorScheme(GCScheme.CYAN_SCHEME);
}

void setChartSettings() {
  g.xDiv=10;
  g.xMax=pSize;
  g.xMin=0;
  g.yMax=0.001;
  g.yMin=0.005;

  g1.xDiv=10;
  g1.xMax=pSize;
  g1.xMin=0;
  g1.yMax=0.001;
  g1.yMin=0.005;
}

double averageValue(float dataArray[])
{

  float total = 0;
  for (int i=0; i<dataArray.length; i++)
  {
    total = total + dataArray[i];
  }

  return total/dataArray.length;
}

double RMSValue(float dataArray[])
{
```

```
  float total = 0;
  for (int i=0; i<dataArray.length; i++)
  {
    total = (float)(total + Math.pow(dataArray[i], 2));
  }
  total /= dataArray.length;
  return Math.sqrt(total);
}
```

## 8.2 PROCESSING CODE FOR SpO2 CALCULATION

```
public class SPO2_cal
{
  float Vdc = 0;
  float Vac = 0;
  float spo2_cal_array[] = new float[pSize];

  float SPO2 = 0;


  public float SPO2_Value(float spo2_array[])
  {
    SPO2 = 0;
    int k = 0;
    for(int i = 50; i < spo2_array.length; i++)
    {
     // float roundoff = Math.round((spo2_array[i] * spo2_array[i])*100)/100;
      spo2_cal_array[k++] = spo2_array[i] * spo2_array[i];
    }
    SPO2 = sum(spo2_cal_array, k);

    return (SPO2);
    //int  p = 0;
    //try
    //{
    //  int j = 0;
    //  while (j < spo2_array.length)
    //  {
    //    int i = 0;
    //    if (spo2_array[j] >= 0)
    //    {
    //      while (spo2_array[i] >= 0)
    //      {
    //        spo2_cal_array[i] = spo2_array[i]*spo2_array[i];
    //        i++;
    //      }
    //      while (spo2_array[i] < 0)
    //      {
    //        spo2_cal_array[i] = spo2_array[i]*spo2_array[i];
    //        i++;
    //      }

    //      j = j+i;
    //      //   println("po :"+i+" "+j);
    //    } else
    //    {
    //      while (spo2_array[i] < 0)
    //      {
    //        spo2_cal_array[i] = spo2_array[i]*spo2_array[i];
    //        i++;
    //      }
    //      while (spo2_array[i] >= 0)
    //      {
    //        spo2_cal_array[i] = spo2_array[i]*spo2_array[i];
```

```
  //        i++;
  //      }
  //      while (spo2_array[i] < 0)
  //      {
  //        rem_array[p] = spo2_array[i];
  //        i++;
  //        p++;
  //      }
  //      float minValue = min(rem_array);
  //      int count = 0;
  //      for (int k = i-p; k < i; k++)
  //      {
  //        if (rem_array[k] == minValue)
  //        {
  //          spo2_cal_array[k] = rem_array[k]*rem_array[k];
  //          count++;
  //          break;
  //        } else
  //        {
  //          spo2_cal_array[k] = rem_array[k]*rem_array[k];
  //          count++;
  //        }
  //      }
  //      j = j+count;
  //      // println("ne :"+i+" "+j);
  //    }
  }


  public float sum(float array[], int len)
  {
    float spo2 = 0;
    for (int p = 0; p < len; p++)
    {
      spo2 = spo2 + array[p];
    }
    Vac = (float)Math.sqrt(spo2/len);

    //println(Vac);
    return Vac;
  }
}
```

## 8.3 PROCESSING CODE FOR GRAPHING

```
class Graph
{

  boolean Dot=true;            // Draw dots at each data point if true
  boolean RightAxis;            // Draw the next graph using the right axis if true
  boolean ErrorFlag=false;      // If the time array isn't in ascending order, make true
  boolean ShowMouseLines=true;  // Draw lines and give values of the mouse position

  int     xDiv=5, yDiv=5;           // Number of sub divisions
  int     xPos, yPos;           // location of the top left corner of the graph
  int     Width, Height;        // Width and height of the graph


  color   GraphColor;
  color   BackgroundColor=color(255);
  color   StrokeColor=color(180);

  String  Title="Title";        // Default titles
  String  xLabel="x - Label";
```

```
String  yLabel="y - Label";

float   yMax, yMin;      // Default axis dimensions
float   xMax=10, xMin=0;
float   yMaxRight=1024, yMinRight=0;

PFont   Font;                  // Selected font used for text

Graph(int x, int y, int w, int h) {  // The main declaration function
  xPos = x;
  yPos = y;
  Width = w;
  Height = h;
}

void DrawAxis() {



  fill(0);
  color(0);
  stroke(0);
  strokeWeight(1);
  int t=60;

  // rect(xPos-t*1.6,yPos-t,Width+t*2.5,Height+t*2);            // outline
  textAlign(CENTER);
  textSize(25);

  fill(255);
  // text(Title,xPos+Width/5,yPos+20);                          // Heading Title
  textAlign(CENTER);
  textSize(14);
  text("No.Of Samples  -  "+Title, xPos+Width/2, yPos+Height+t/1.5);                  // x-axis Label

  rotate(-PI/2);                                    // rotate -90 degrees
  text("", -yPos-Height/2, xPos-t*1.6+40);                  // y-axis Label
  rotate(PI/2);                                     // rotate back
  fill(255);
  textSize(10);
  noFill();
  stroke(0);
  smooth();
  strokeWeight(1);
  //Edges
  stroke(255);
  line(xPos-3, yPos+Height, xPos-3, yPos);                       // y-axis line
  line(xPos-3, yPos+Height, xPos+Width+5, yPos+Height);          // x-axis line

  stroke(200);

  if (yMin<0) {
    line(xPos-7, // zero line
      yPos+Height-(abs(yMin)/(yMax-yMin))*Height, //
      xPos+Width,
      yPos+Height-(abs(yMin)/(yMax-yMin))*Height
      );
  }

  stroke(255);

  for (int x=0; x<=xDiv; x++) {
```

```
    line(float(x)/xDiv*Width+xPos-3, yPos+Height, //  x-axis Sub devisions
      float(x)/xDiv*Width+xPos-3, yPos+Height+5);

    textSize(10);                              // x-axis Labels

    String xAxis=str((xMin+float(x)/xDiv*(xMax-xMin))); // the only way to get a specific number of decimals
    String[] xAxisMS=split(xAxis, '.');                 // is to split the float into strings
    text(xAxisMS[0]+"."+xAxisMS[1].charAt(0), // ...
      float(x)/xDiv*Width+xPos-3, yPos+Height+15);   // x-axis Labels
   }

   for (int y=0; y<=yDiv; y++) {
    line(xPos-3, float(y)/yDiv*Height+yPos, // ...
      xPos-7, float(y)/yDiv*Height+yPos);              // y-axis lines

    textAlign(RIGHT);
    fill(255);

    String yAxis=str(yMin+float(y)/yDiv*(yMax-yMin));     // Make y Label a string
    String[] yAxisMS=split(yAxis, '.');                   // Split string
    //println(yAxisMS);
    text(yAxisMS[0]+"."+yAxisMS[1], // ...
      xPos-15, float(yDiv-y)/yDiv*Height+yPos+3);       // y-axis Labels
   }
   stroke(0);
  }
  void LineGraph(float[] x, float[] y) {

   for (int i=0; i<(x.length-1); i++)
   {
    smooth();
    strokeWeight(3);
    stroke(GraphColor);
    line(xPos+(x[i]-x[0])/(x[x.length-1]-x[0])*Width,
      yPos+(Height)-(y[i]/(yMax-yMin)*Height)+(yMin)/(yMax-yMin)*Height,
      xPos+(x[i+1]-x[0])/(x[x.length-1]-x[0])*Width,
      yPos+(Height)-(y[i+1]/(yMax-yMin)*Height)+(yMin)/(yMax-yMin)*Height);

   }

   stroke(0);
   fill(0);
   rect(xPos-22+((time-2)-x[0])/(x[x.length-1]-x[0])*Width,0,50,height);
  }
}
```

**[9]**