Department of Electronic and Telecommunications Engineering

EN2091 Laboratory Practice and Project

Final Report

ECG Heart Monitor

Charles J.	210079K
Madugalle M.R.K.J.M.M.S.B	210351M
Kavishan G. T.	210285M
Wijesinghe C. D.	210720U

Contents

Abstract

Our project is regarding the development of a analog ECG heart monitor for real-time cardiac activity monitoring. Here we utilize only analog components to amplify heart-generated electrical signals and for signal conditioning to reduce noise. It is then followed by analog-to-digital conversion and subsequent signal processing stages, implemented on a microcontroller. Additionally, our project explores the integration of wireless communication for remote monitoring.

1 Introduction

An electrocardiogram (ECG) is a simple diagnosis technique employed to assess the heart's electrical activity by detecting electrical signals generated during each heartbeat through electrodes fixed to the skin. The voltage difference between the right arm and left arm is magnified, incorporating a feedback mechanism through the right leg. ECG leads measure these voltages from the right arm, left arm, and right leg. The voltage signal's amplitude ranges from 0.001 mV to 100 mV (typical value is 1 mV), with a frequency spanning from 0.01 Hz to 250 Hz. Amplifying this signal proves challenging due to the small amplitude of the raw ECG signals and subjection to corruption from various sources such as noise, power line interference, RF inteference, noise from electrode contact, stray capacitance, and bio signal artifacts induced by movements of the subject.

In our project we propse and implement a analog circuit to amplify the above eletrical signal and to perform filtering to reduce noise. We make use of only basic analog electrical devices such as resistors, capacitors and OpAmps in our design.

The below block diagram illustrates the stages of signal conditioning employed in our design.

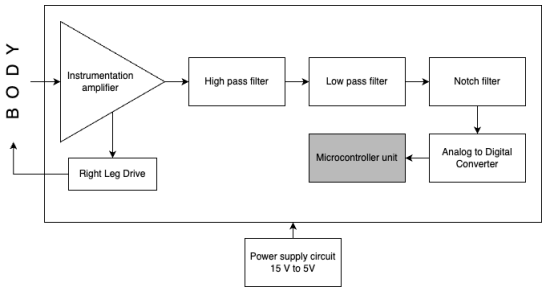


Figure 1: Block diagram of the circuit design

We implement the design of our circuit on a single PCB using the Altium PCB design software. All circuits were simulated using LTSpice and MultiSim prior to implementation.

2 Functionality

2.1 Instrumentation amplifier

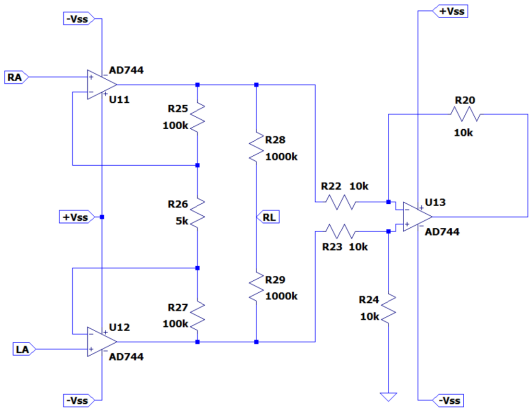


Figure 2: Instrumentation amplifier

The primary objective of the Instrumentation amplifier is amplifying the small electrical signals generated by the heart. These ECG signals are typically in the milli-volt range. The instrumentation amplifier amplifies this by amplifying the differential voltage between two electrodes while also rejecting common-mode inputs like interference from nearby power lines or

biosignal artifacts.

The proposed design of the instrumentation amplifier in our circuit operates using three TL071CP OpAmp ICs. We selected this IC because of its high Common Mode Rejection Ratio (CMRR). It has a minimum CMRR of 95 dB which was suitable enough for the amplification process. The input to the circuit is fed through two ECG leads. The gain can be adjusted using the variable resistor in the middle. We found through fine-tuning that preferred gain is 35.6.

2.2 Right Leg Drive

The Right Leg Drive (RLD) circuit is used to minimize the common-mode signal between the two input electrodes. This is done by taking the original signal, inverting and amplifying it and sending back to the body through the right leg. In the RLD configuration we use two TL072CP OpAmps; one acting as a unity gain buffer and the other as an inverting amplifier. We also include a high impedance resistor ($0.5M\Omega$) before injecting the signal to the body for patient protection purposes. The gain of the RLD stage is 67.

2.3 Filter specifications

The selection of proper cutoff frequencies for filtering was a pivotal decision. The American Heart Association (AHA) suggests a minimum bandwidth of 150Hz for individuals aged 12 to 16 years and a minimum bandwidth of 125Hz for adults[1]. Based on this recommendation and analyzing several other designs, we chose the range of 0.01 Hz - 170 Hz to preserve accuracy while maintaining minimal noise.

We decided on implementing filter circuits using active filters to prevent loading effect. Furthermore, we used TL072CP OpAmp ICs when implementing the filter circuits.

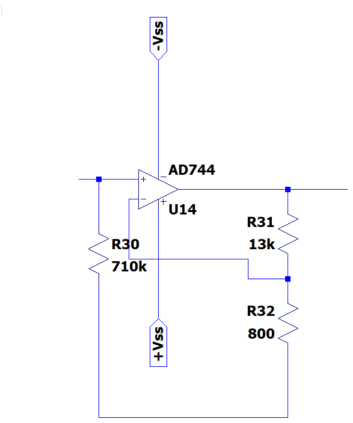


Figure 3: High-pass filter

2.4 High-pass filter

A first-order active high-pass filter is a circuit was proposed to allow signals with frequencies higher than 0.01 Hz to pass through while attenuating lower-frequency signals. The main objective was this to remove the DC shift of the original electrical signal.

For a high-pass filter, we used a first order active filter since the precision and the rapid change of the transition band was not significant. The cutoff frequency (f_c) of a first-order high-pass filter is given by the formula:

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

2.5 Low-pass Filter

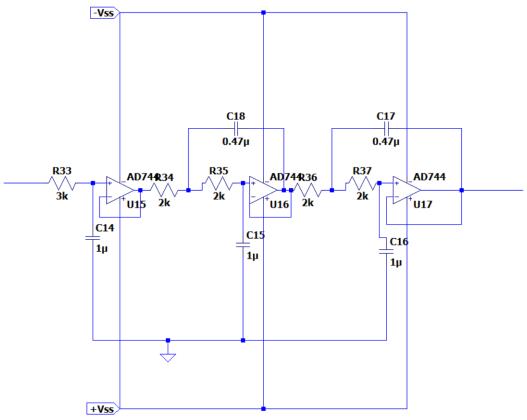


Figure 4: Low-pass filter

We chose our cutoff frequency of the low-pass filter as 170 Hz. We also decided that the transition needs to be as steep as possible therefore adapted a 5th order configuration. The Bessel filter yields stable transient and constant group delay [2].

We realized the 5th order filter by cascading a first order filter with two 2nd order filters with Sallen-Key topology. The gain of the filter is unity. The following formula was used to determine the capacitor and resistances needed.

$$f_c = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$$

2.6 Notch filter

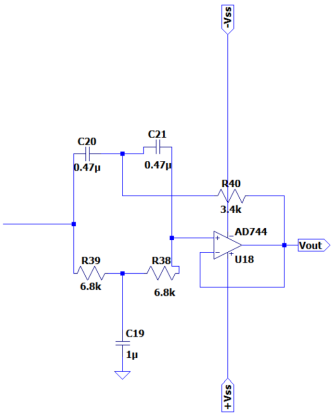


Figure 5: Notch filter

The purpose of the notch filter (band reject filter) is to remove the spike at 50-60 Hz in the spectrum of the signal. This spike is resulted due to the power line interference. It is realized by a twin-T RC circuit including a TL072CP OpAmp. This filter configuration is an active filter. Also a quality control circuit is appended to the filter to optimize the steepness of the transition band by adjusting the Q-factor. The resistances are selected to be adjustable using variable resistors so that the exact power line interference frequency can be cut off. The gain from the notch filer is unity while a non-unity gain is added by the quality circuit.

The cutoff frequency of the notch filter can be approximated as using the following.

$$f_n = \frac{1}{2\pi RC}$$

Additionally, the Q-factor can be approximated using the below formula.

$$Q = \frac{R_1}{2(R_1 - R_2)}$$

2.7 Power circuit

The OpAmp ICs in all circuits require +5v and -5V supply voltage for preferred operation. To synthesize these voltages we propose the power circuit shown in **figure 6**. The input to the power circuit is taken through a 15V DC power adapter bought from the market. This 15V input voltage is then broken down to into two 7.5V potentials of opposite polarity. The LM7805 and LM7905 regulators are used to obtain stabilized voltages of +5V and -5V. Push-pull transistor configuration is used to balance the load. Capacitors are used for stabilization.

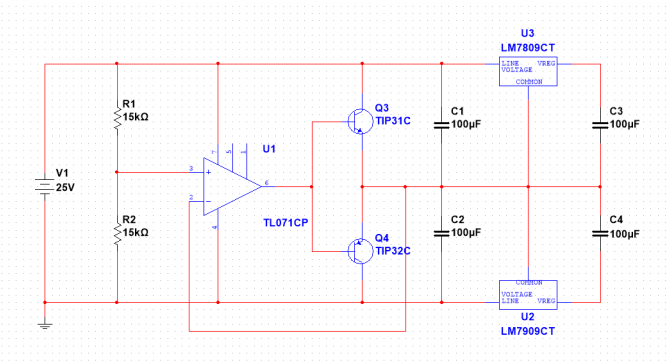


Figure 6: Power circuit

2.8 Analog to Digital Conversion Pre-conditioning circuit

The signal is taken into the ESP32 Micro-controller through one of the chip's inbuilt Analog to Digital conversion pins. The constraint for ADC conversion is to provide signals only between 0-3.3V to the input pin. Since the notch filter output clearly exceeds this range we implemented a level shifting

circuit to shift and scale down the output signal to the 0-3.3V range. The proposed circuit is shown in **figure 7**.

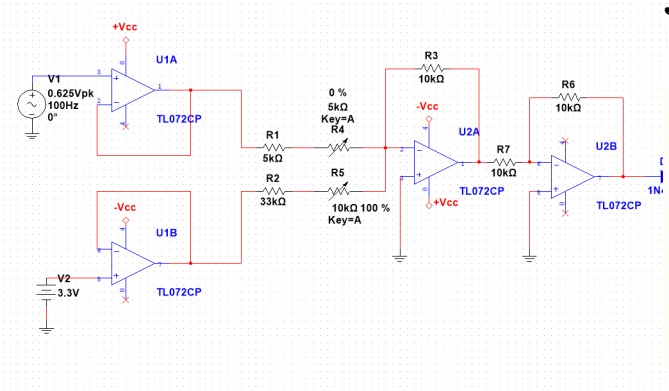


Figure 7: Shifting and scaling circuit

3 Design Parameters

The following are the gain and frequency response calculations for individual stages using the PCB schematics.

3.1 Instrumentation Amplifier

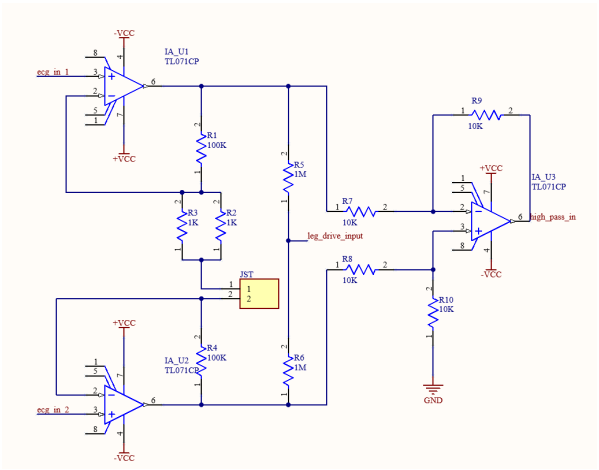


Figure 8: Instrumentation amplifier PCB Schematic

Gain

$$Gain = \left(\frac{R_1 + R_4 + R_A}{R_A}\right) = 1 + \left(\frac{200}{R_A}\right)$$

Here the overall impedance of R_2, R_3 and variable resistor is taken as R_A. By changing the resistance R_A we can obtain a gain within 20-400.

3.2 Right Leg Drive

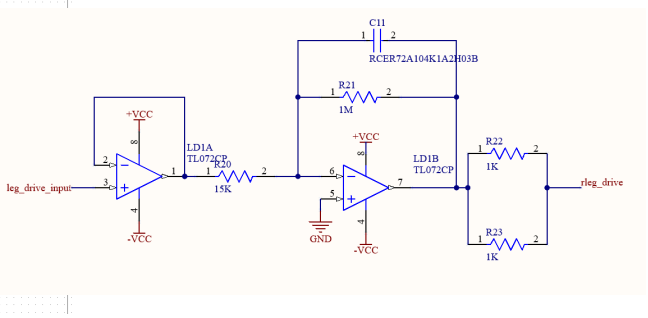


Figure 9: Right leg drive PCB schematic

Gain

$$Gain = \frac{R_{21}}{R_{20}} = \frac{1M\Omega}{15k\Omega} = 67$$

The right leg drive circuit outputs a signal with unity gain and 180° phase delay.

3.3 High-pass filter

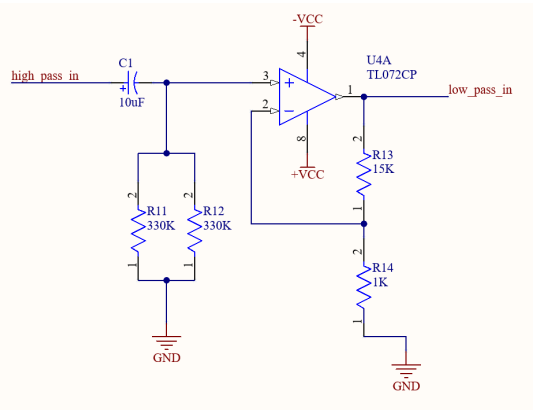


Figure 10: High-pass filter circuit PCB Schematic

Gain

$$Gain = 1 + \left(\frac{R_{13}}{R_{14}}\right) = 1 + \left(\frac{15k\Omega}{1k\Omega}\right) = 16$$

Cut-off frequency

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \times 10\mu F \times 165k\Omega}$$

$$f_c = 0.096Hz$$

3.4 Low-pass filter

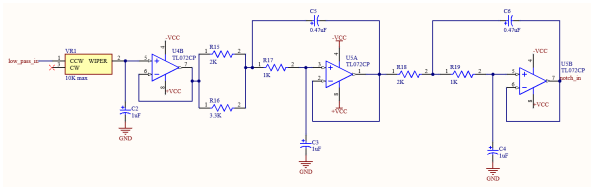


Figure 11: Low-pass filter PCB Schematic

The gain if the low-pass filter is unity.
The cut-off frequency of the low-pass filter can be calculated from the RC values of the last OpAmp's configuration.

Cut-off frequency

$$f_c = \frac{1}{2\pi\sqrt{2k\Omega \times 1k\Omega \times 0.47\mu F \times 1\mu F}}$$
$$f_c = 164Hz$$

This is close the required cut-off value of 170 Hz.

3.5 Notch filter

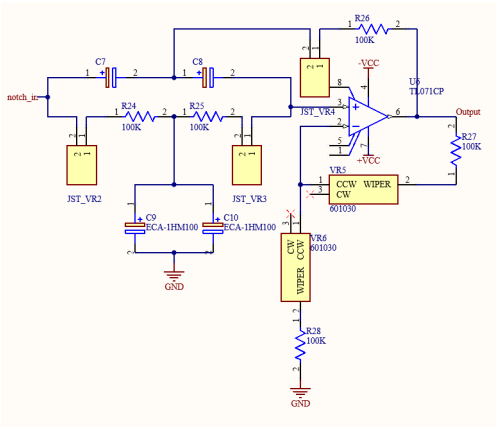


Figure 12: Notch filter PCB Schematic

Gain
The gain of the notch filter is only due to the quality circuit (Due to the impedences VR5 and VR6). We fine-tuned the values of these variable resistors to obtain the preferable gain as follows.

$$Gain = 1 + \left(\frac{2.5k\Omega}{4k\Omega}\right) = 1.625$$

Cut-off frequency

The cut-off frequency of the circuit can be varied according to the power-line noise existing. We therefore do not calculate the exact cut-off frequency. But the resistor values we used are shown in the PCB schematic.

4 PCB design

All circuits mentioned above were realized and fabricated on a single PCB. The PCB was deisgned using Altium Designer PCB Software. In our design approach, we employed 4 layers to minimize interference. This consists of two signal layers, the ground layer and the power layer. The PCB was designed in a way that the processor (ESP32 Dev Module) is also mounted on the same PCB. The PCB and display wiring is also included within the PCB routing.

After the fabrication, the relevant components are soldered by hand to the PCB. For convenience of replacement, we soldered IC bases and mounted the ICs onto them. Additionally, we used JST cables and pin headers for easy replacement. The final appearance of the PCB after soldering is shown in **Figure 11**.

PCB layout diagrams for individual layers are shown in the **Appendix I**.

5 Enclosure design

The SolidWorks CAD software was utilized to design the enclosure. This enclosure has a removable top lid, facilitating the fixation and replacement of components as needed. To prevent damage to the components, there are strategically placed ventilation and airflow, allowing heat to dissipate to the outside. The top lid houses the display and knobs for filter

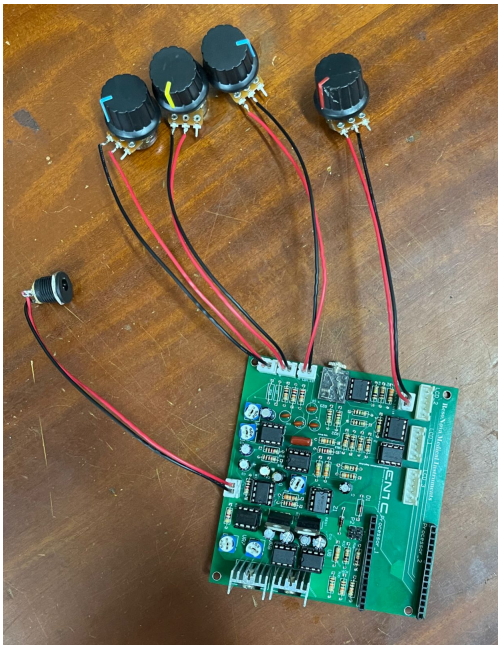


Figure 13: PCB after soldering all the components

tuning. A power on/off switch is positioned on the side. The enclosure is equipped with holes for power adapter input and 3.5mm jack connections.
3D views and drawings of the enclosure design are shown in the **Appendix I**.

6 Micro-controller programming

The input taken into the ESP32 Dev Module is then processed to be drawn on to a TFT LCD dislpay. The micro-controller is programmed using the Arduino IDE with C++ programming language. We used the **MCUFRIEND kbv** Arduino library to operate the TFT display. The full code for the program is shown in the **Appendix III** and is also uploaded to Github.

7 Simulating circuits and testing

All circuits have been simulated on LTSpice and MultiSim simulation software before implementing physically. Thereafter we realized the circuits on breadboards and tested their functionalities individually as well as cascaded. The simulation results and breadboard implementation is shown in the **Appendix II**.

8 Final prototype

Shown below is the working finished prototype with the PCB mounted and components assembled.

Figure 14: Final device prototype

9 Task allocation

Charles J. - PCB soldering, Notch circuit optimization
Kavishan G.T. - PCB Design, Filter circuits optimization
Madugalle M.R.K.J.M.M.S.B - Enclosure design, Intrumentation amplifier optimization
Wijesinghe C.D. - ESP32 programming, RLD circuit optimization

10 Resources

GitHub repository including all the schematic files, PCB and enclosure design files and code can be found here: github.com/Sajitha-Madugalle/Heart_Rate_Monitor.

11 References

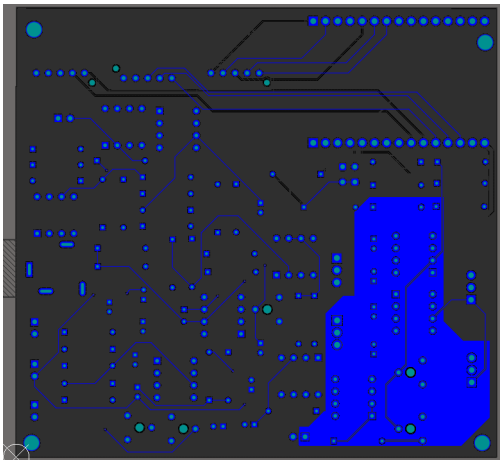
[1] P. R. Rijnbeek, J. A. Kors, and M. Witsenburg, "Minimum Bandwidth Requirements for Recording of Pediatric Electrocardiograms," Circulation, vol. 104, no. 25, pp. 3087–3090, Dec. 2001, doi: https://doi.org/10.1161/hc5001.101063.

[2] J. Karki, "Active Low-Pass Filter Design." Accessed: Dec. 05, 2023. [Online]. Available: www.ti.com/lit/an/sloa049d/sloa049d.pdf

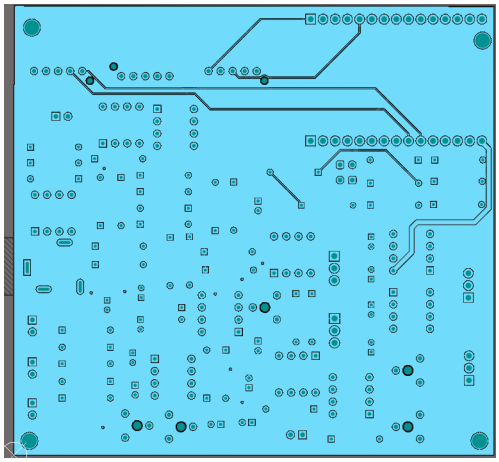
[3] "Design of an ECG sensor circuitry for cardiovascular disease diagnosis," International Journal of Biosensors & Bioelectronics, vol. Volume 2, no. Issue 4, May 2017, doi: https://doi.org/10.15406/ijbsbe.2017.02.00032.

Appendix I - PCB Layout and Enclosure design

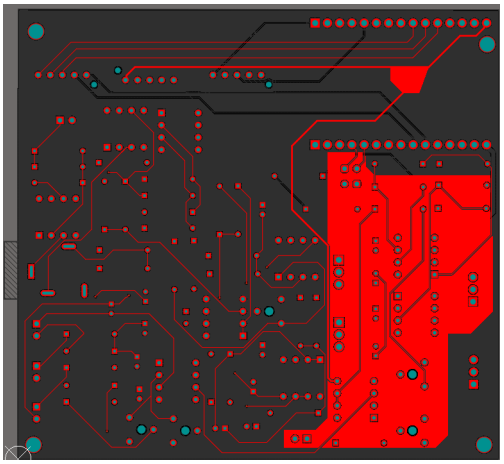
PCB Layout from Altium Designer



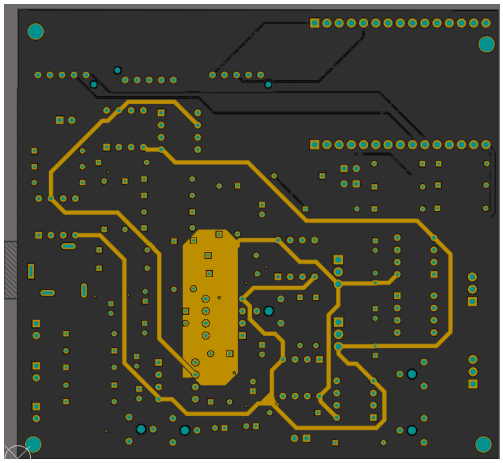
(a) Bottom layer



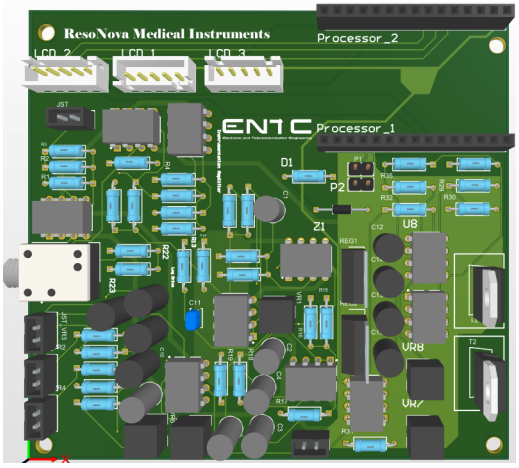
(b) Ground layer



(c) Top layer



(d) Power layer



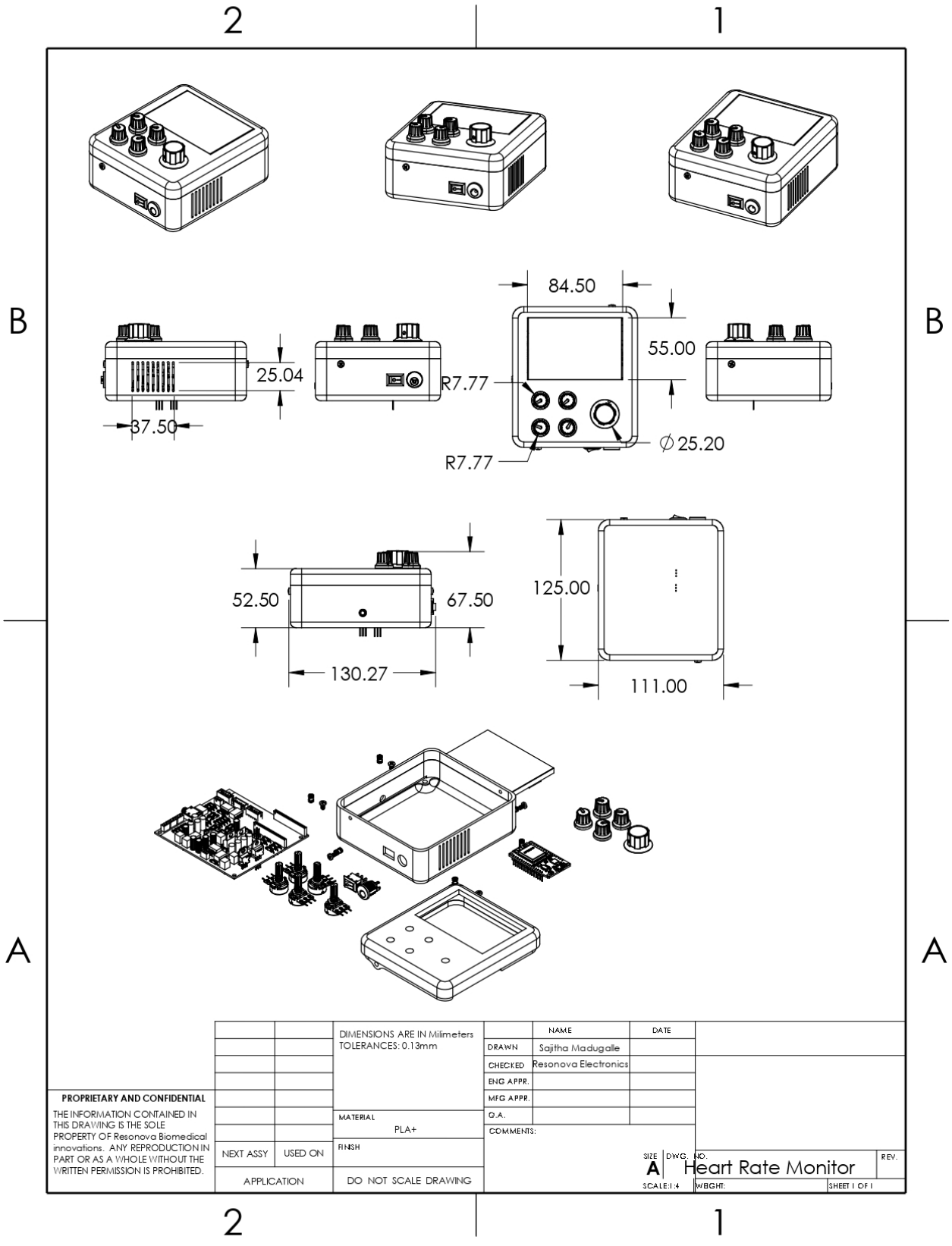
(e) 3D view

Figure 15: PCB layout for different layers

Enclosure design using Solidworks

Figure 16: Enclosure design using Solidworks

Enclosure design drawings



Appendix II - Simulation and Testing

LTSpice simulation of the circuit

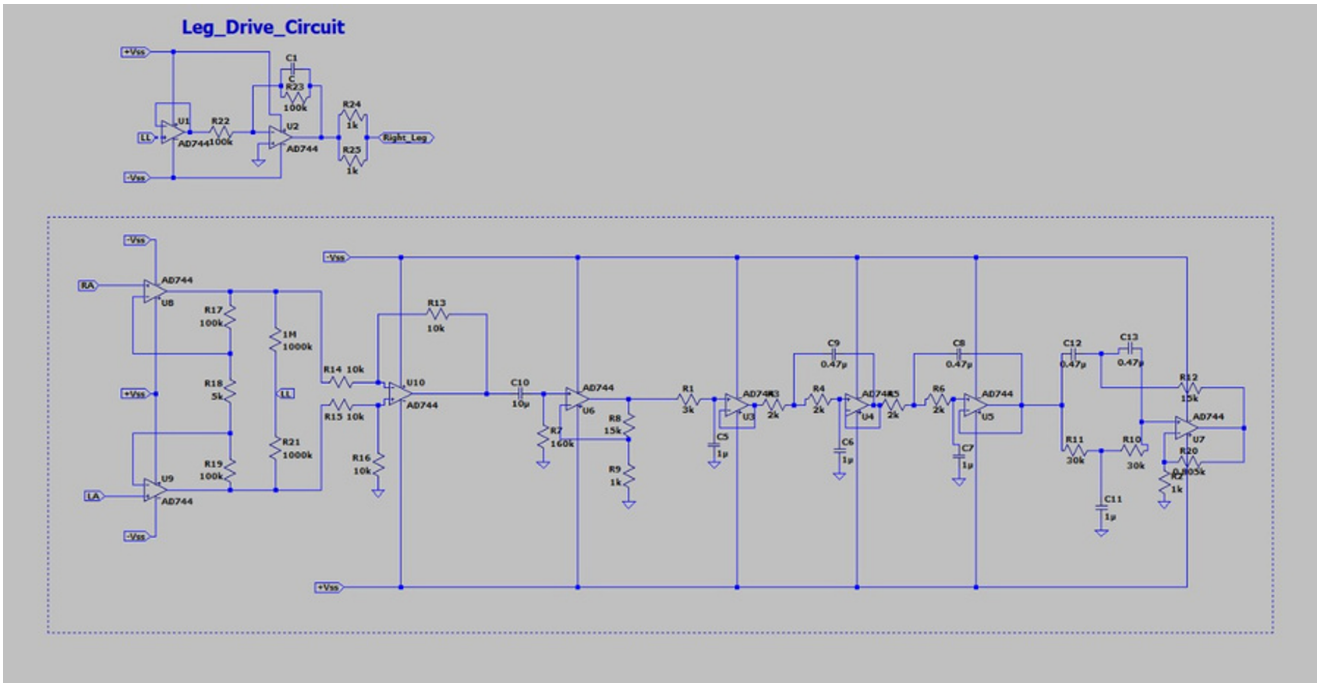


Figure 17: LTSpice schematic of the initial design

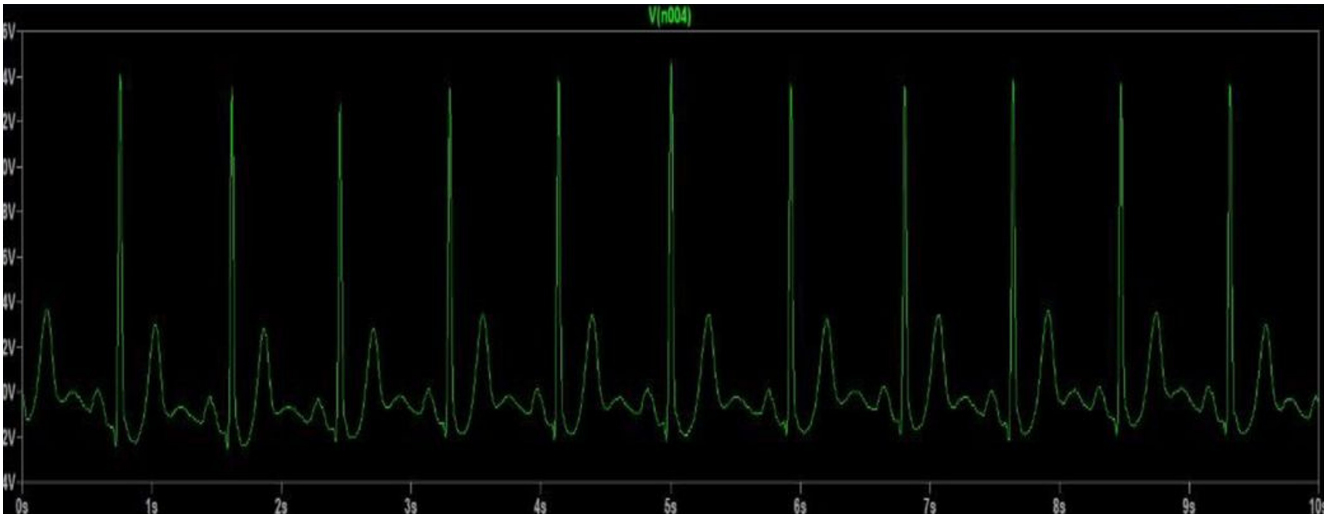


Figure 18: Simulation output from the LTSpice schematic

Breadboard implementation

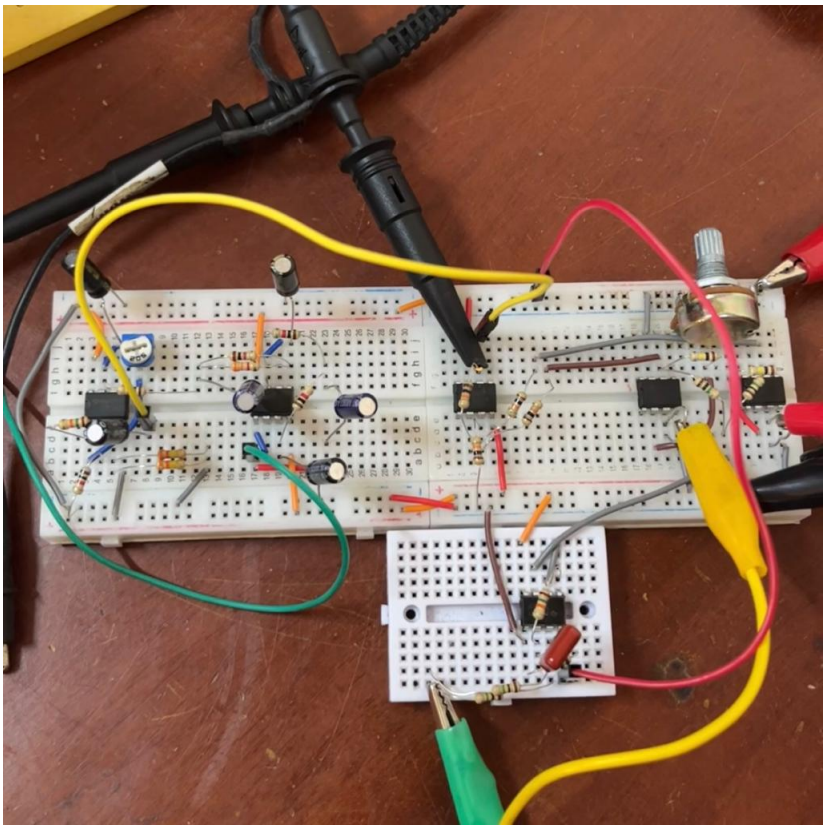


Figure 19: Breadboard implementation of Instrumentation amplifier, RLD and filter circuits

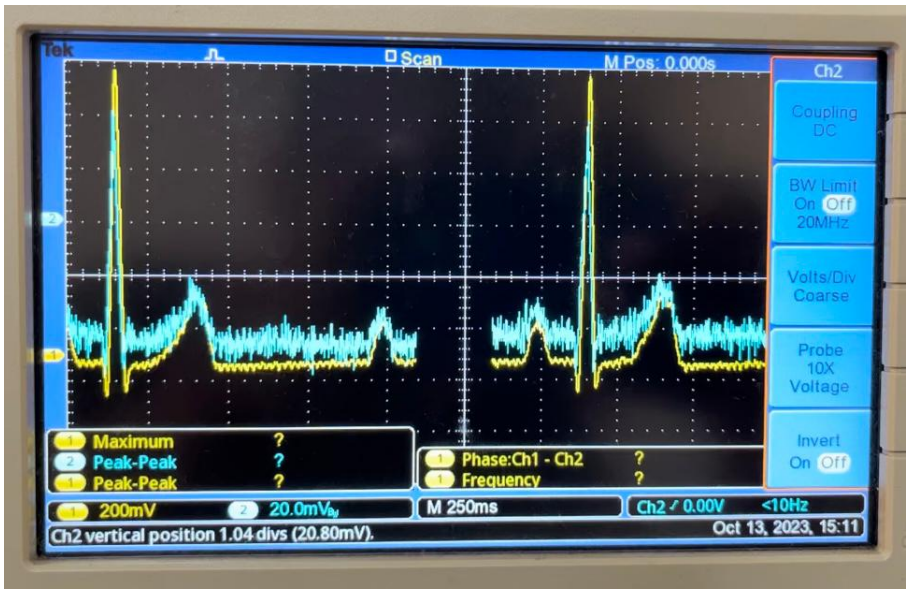


Figure 20: Output from the breadboard circuit observed through an oscilloscope. Cyan waveform shows output after low-pass filtering. Yellow waveform shows output after notch filter.

Appendix III - Microcontroller code

ESP32 Dev Module C++ code

```
#include <Adafruit_GFX.h>
#include <MCUFRIEND_kbv.h>
MCUFRIEND_kbv tft;


// Define color codes
#define BLACK 0x0000
#define BLUE 0x001F
#define GREEN 0x07E0
#define RED 0xF800
#define WHITE 0xFFFF

// Define variables for time intervals
unsigned long previousMillis1, previousMillis2 = 0;
const long interval1 = 2;
const long interval2 = 5;

void setup(void)
{
  Serial.begin(9600);

  // Initiate the display
  uint16_t ID = tft.readID();
  Serial.println("Example: Font_simple");
  Serial.print("found ID = 0x");
  Serial.println(ID, HEX);
  if (ID == 0xD3D3) ID = 0x9481; // force ID if write-only display
  tft.begin(ID);

  // Set screen orientation
  tft.setRotation(1);
  tft.fillScreen(BLACK);

  // Draw the grid
  drawGrid();
}

void loop() {
  unsigned long currentMillis = millis();

  // Task1 - Draw ECG graph
  if (currentMillis - previousMillis1 >= interval1) {
    previousMillis1 = currentMillis;
    runECG();
  }
}

void drawGrid(){
  // Clear screen
  tft.fillScreen(BLACK);
  // Set text
  tft.setCursor(20, 20);
  tft.setTextColor(WHITE);
  tft.setTextColor(WHITE, BLACK);
```

```
  tft.setTextSize(3);
  tft.println("ECG");

  // Draw vertical lines
  for (int i = 1; i <= 6; i++){
    tft.drawLine(80 * i,0,80*i, 320,BLUE);
  }

  // Draw horizontal lines
  for (int j = 1; j <= 4; j++){
    tft.drawLine(0,80 *j,480, 80*j,BLUE);
  }
}


void runECG(){
  // Read analog value from ADC pin
  long ecgValue = analogRead(34);

  // Normalize the value read to it screen size
  int normalizedVal = map(ecgValue, 0, 4095, 0, 320);

  static int lastX = 0;
  static int lastY = tft.height()/2 + 50;
  int x = lastX + 2;
  int y = tft.height() - normalizedVal;

  // Reset cursor if reached the right end of scren
  if (x >= tft.width()){
    x = 0;
    drawGrid();
    lastX = 0;
    lastY = tft.height()/2 + 50;

  }

  // Draw ECG line
  tft.drawLine(lastX, lastY, x, y, GREEN);
  lastX = x;
  lastY = y;
}
```