ABDULLAH GÜL
ÜNİVERSİTESİ

ECG SIGNAL ACQUISITION

TEAM ELECTROSHOCK

Malik Enes ***GÜNGÖR***

Mehmet Ali ***ATILGAN***

Muratcan ***YAZICI***

Table of Contents (ctrl + left click for specified use)

1. INTRODUCTION

Electrocardiogram, ECG or EKG, is an operation that measures the electrical activities of the heart. The heart pumps blood to the body in each beat. According to American Heart Association, the upper chambers (or the right and left atria) create the first wave called *P-Wave*, and ventricles (or the right and left bottom chambers) create the *QRS complex.* The final wave is denoted as *T-wave,* and it represents the electrical return to a resting state for ventricles. The P-QRS-T complex gives hints to the doctors about the disease of the patient. It is not a harmful operation, yet may be crucial for human health. A typical ECG signal is illustrated in Figure 1.

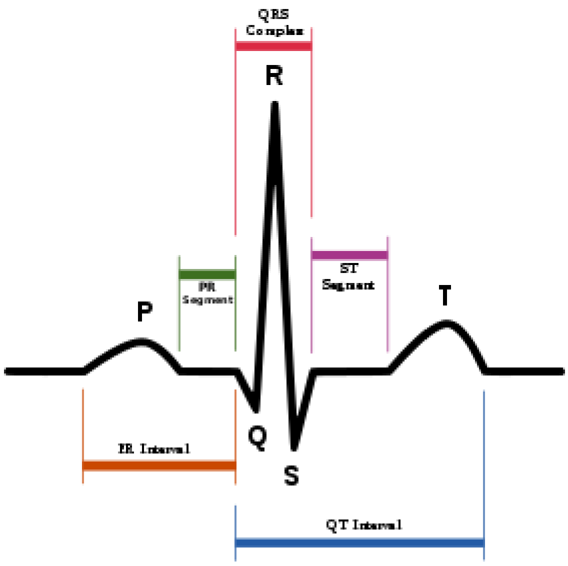

Figure 1: A typical ECG signal

In this project, an ECG is developed to acquire the electrical signals originated by the heart using a typical transducer, electrodes. The aim of this project is to obtain the heartbeats, apply several different processes, and find the beat-per-minute (BPM) rate of the patient.

This report also includes the troubles, problems, unexpected situations, and our solution approaches to them.

2. BLOCK DIAGRAM

This project consists of several different operations. In order to analyze every step and make the process clearer, a block diagram is created. The block diagram is shown in Figure 2.1.

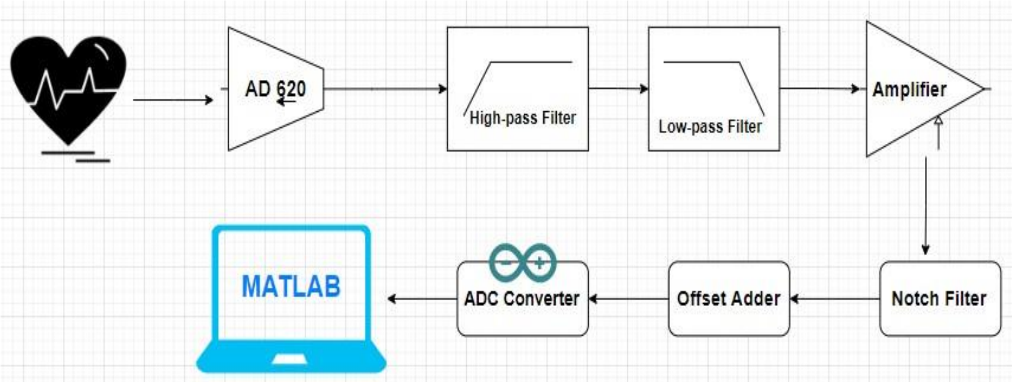

Figure 2.1: Block diagram

Each block is explained in detail in this report. The order of the blocks is important, and it was our decision to put them in the order shown in Figure 1. Since the process will be on heartbeats, the first step is to obtain heart movements from the human body. Next, these signals are sent to the special operational amplifier (AD620) to eliminate the common noises and amplify the incoming signal. Then, we decided to use the high-pass filter to get rid of the low-frequency noises so that we could process on the signal in an effective manner. Then, a low-pass filter is used to eliminate the high-frequency noises in the incoming signal. The order of the high-pass and low-pass filter would not create a difference. In all these blocks, the signal loses amplitude. We found out that the amplitude drops to around 70 mV. Thus, the amplifier section is put right after the filtering sections so that the signal could be further processed. After the amplification, a notch filter is applied to further eliminate the city-line noise. Then, an offset addition is performed due to the desired voltage level of the ADC converter. Finally, the signal is sent to the ADC, and then to the computer for further processes. The details of these blocks are explained in this report.

2.1 HEART MOVEMENT SENSING

The initial step is to take the heart signals from the heart. Using ECG electrodes and conductive cables, the heart signal is sent to the AD 620. There are several different electrode placements. Right & left chests and abdomen or right & left-hand wrist and right leg are examples of three lead electrode placements. Each electrode placement methods give slightly different results. After this initial step, the captured signal is sent to the AD 620 instrumentation- amplifier.

2.2 AD 620

AD 620 is a high accuracy instrumentation amplifier that needs only one external resistor. It is possible to create a gain up to 10,000 with this external resistor. AD-620 is useful for data acquisition systems, specifically for medical applications. It is the main reason why this instrumentation amplifier is preferred for this project. The gain is calculated by arranging the R_G value. The equation is,

$$G = \frac{49.4k\Omega}{R_G} + 1$$

The determination of R_G value is critical in this step. It should not be so low since it will cause an unstable output, and it should not be so high since the output will not be seen due to the low amplitude. When the R_G is selected as 8kΩ and the amplification factor is set to 8, the output signal of the AD-620 had a low amplitude and it would be hard to process on it. Then, we decreased the R_G to 560Ω and set the amplification gain to approximately 90. The ECG signal that came from the output of AD620 is shown in Figure 2.2.1. Note that this signal is the very first signal that has not been through the processes.

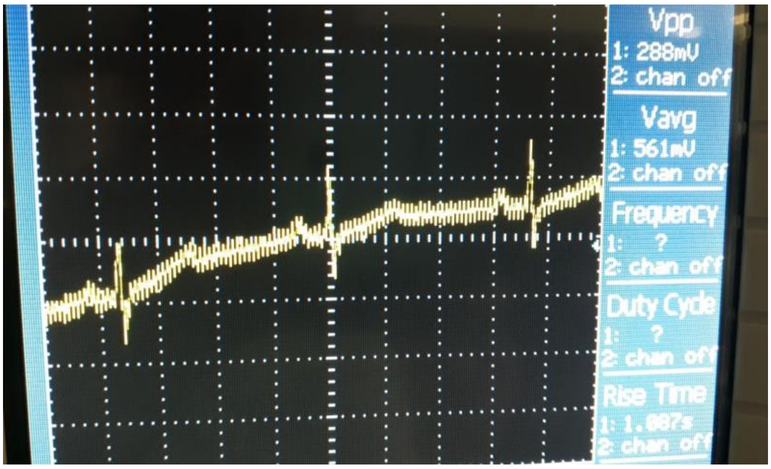

Figure 2.2.1: The output of AD620 when R_G is 560Ω

Here, it can be estimated that the amplitude of the signal without any amplification is approximately $(\frac{288mV}{90})$ 3.2mV.

2.3 FILTERING

Filtering is one of the most important applications in signal processing. In this project, the raw ECG signal should be filtered out from unwanted signals such as high frequency and low-frequency noises. To get rid of low-frequency noises, high pass filters are used. In addition, to get rid of the high-frequency noises, low-pass filters are used. In this project, we decided to perform a band-pass filtering by cascading both low-pass filter and high-pass filter. We also used notch filtering to further filter the city line noise.

In filtering applications, determining the cut-off frequencies is crucial. For this purpose, the frequency components of an ECG signal should be carefully analyzed. According to Fuentes [2] (2012), the frequency components of an ECG is as following:

- Heart rate: 0.67 – 5 Hz (i.e. 40 – 300 bpm)
- P-wave: 0.67 – 5 Hz
- QRS: 10 – 50 Hz
- T-wave: 1 – 7 Hz
- High frequency potentials: 100 – 500 Hz

The frequency characteristics of an ECG signal is shown in Figure 2.3.1.

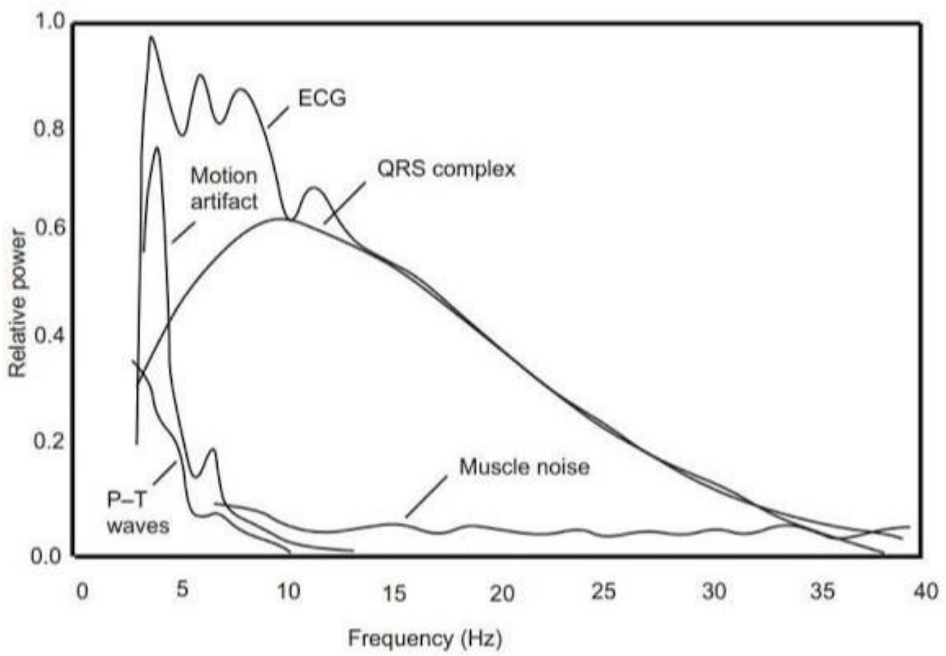

Figure 2.3.1: Frequency characteristics of an ECG [2]

The cut-off frequencies are determined based on this graph.

2.3.1 HIGH-PASS FILTERING

The low-frequency noises affect the shape of an ECG signal. The movements of the patient and respiration generally cause these noises. Fuentes states that the frequency band of this noise is generally between 0 and 0.5Hz. So, an ideal high-pass filter with a cutoff frequency of 0.5Hz would perfectly fit in this application. However, it is impossible to create an ideal filter. When 0.5Hz was set as cutoff frequency of a passive filter, frequencies up to 5Hz were effected. The experimental frequency response of a first-order high-pass filter with cutoff 0.5 Hz is shown in Figure 2.3.1.1.

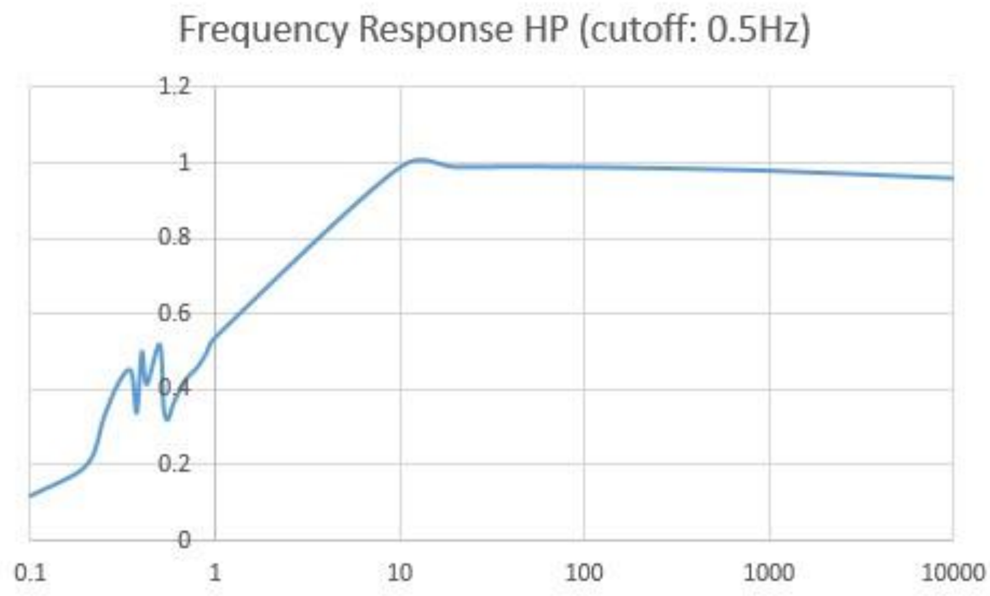

Figure 2.3.1.1: Frequency response of the high-pass filter (Amplitude (V)/ Frequency (mHz))

As can be seen in Figure 2.3.1.1, frequencies up to 10 are affected. This causes the data loss for heart rate, T wave, and P wave. Thus, we set the cutoff frequency to 0.1Hz and tested it. The type of filter was first-order passive. We also tested the higher orders passive and active designs, yet did not spot a considerable benefit. In order not to create complexity and to decrease the cost, we preferred to use a first-order passive filter. The cutoff frequency is calculated by using the following formula.

$$f_c = \frac{1}{2\pi RC}, \qquad R: 158.4k\Omega, C: 10uF,\ f_c = 0.1Hz$$

We set the circuit shown in Figure 2.3.1.2, and simulated it in PsPice. The frequency response of the simulation is shown in figure 2.3.1.3. The experimental frequency response of the filter is shown in Figure 2.3.1.4.

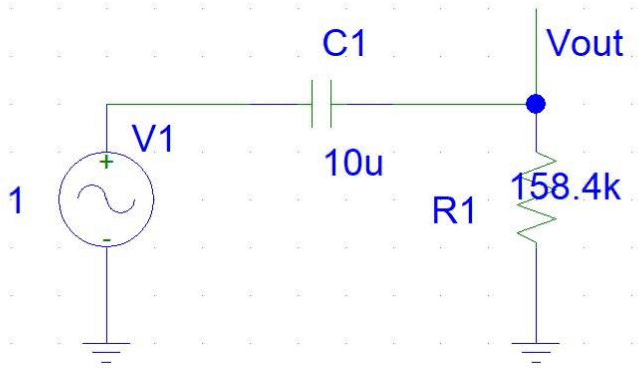

Figure 2.3.1.2: First order high-pass filter circuit

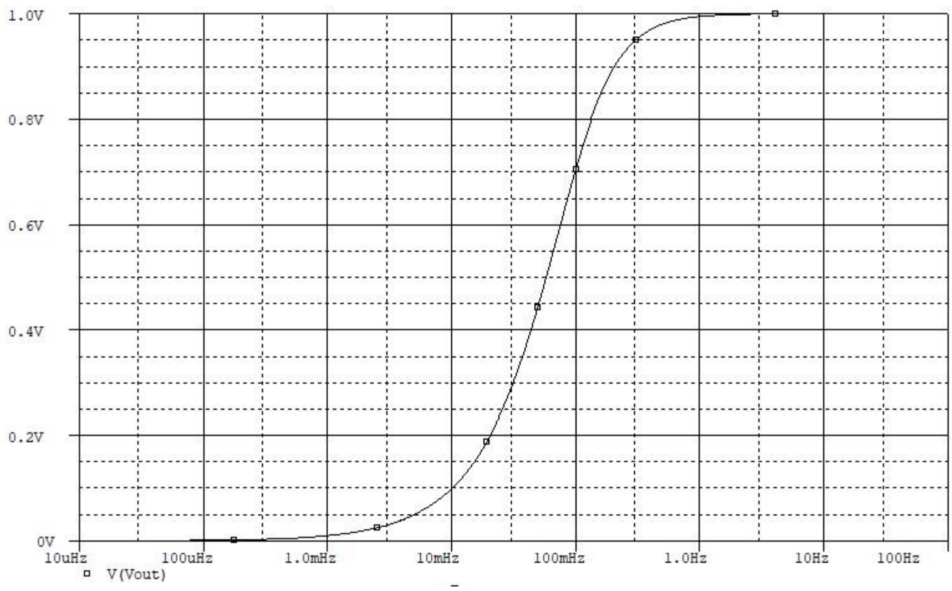

Figure 2.3.1.3: Frequency response of the simulation

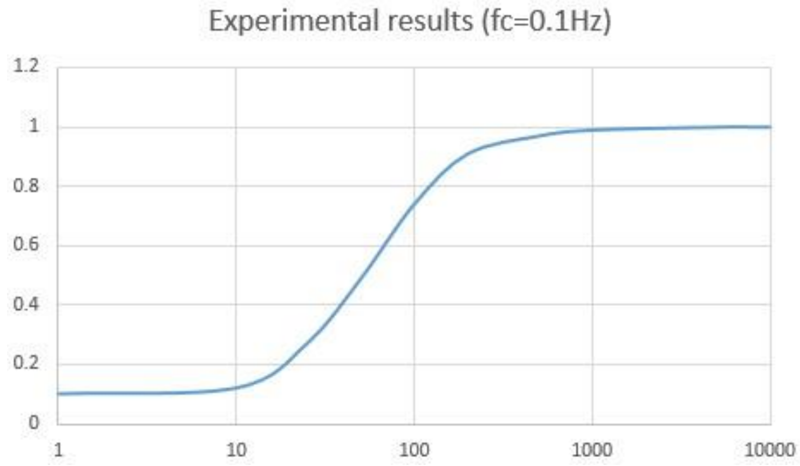

Figure 2.3.1.4: The frequency response (Amplitude (V)/ Frequency (mHz))

As can be seen in Figure 2.3.1.4, the frequencies up to 100mHz nearly eliminated, and the frequencies that carry the information (frequencies above 670mHz) passed. After obtaining this result, we directly added this to our system.

Finally, figure 2.3.1.5 illustrates the real result. The yellow signal is the output of AD620, and the blue signal is the output of the high-pass filter.

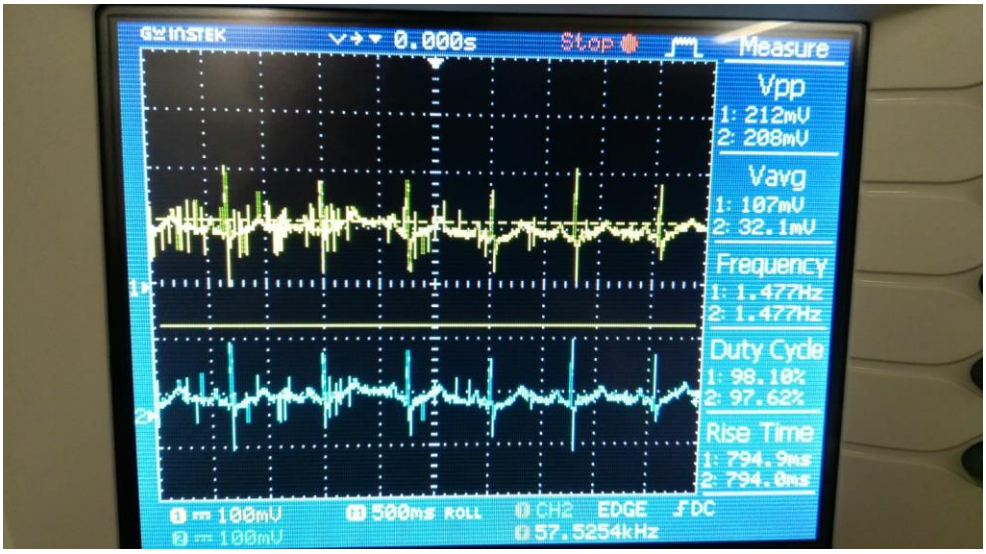

Figure 2.3.1.5: Real input (yellow) – output(blue) results

Note that the signals above still have high-frequency noise because a low-pass filter is not applied yet. Moreover, the difference is hard to detect because the frequencies that are eliminated are too low.

2.3.2 LOW-PASS FILTERING

A low-pass filter is needed to eliminate the high-frequency noise. Several external factors cause high-frequency noise such as body movements and the city line. When the frequency characteristics of an ECG is analyzed using FFT, it was clear that the most dominant high-frequency noise in the ECG is the 50Hz city-line noise. Thus, the determination of the cut-off frequency has a critical role. To select the most appropriate cut-off frequency, we did a literature survey. Guldenring (2016) states that 12-lead ECG monitoring is typically recorded using a 40Hz cut-off frequency for low pass filtering. QRS-T complex is not significantly affected if a 40Hz cut-off is used. In this way, the high-frequency noise is eliminated, and the QRST complex is preserved. Even though we know that the ECG has information up to 500HZ, we needed to filter this information as well because we are not particularly interested in the information at high frequencies. We are interested in the QRST segment, particularly in R peaks. Plus, there is no absolute gain in filtering. The noise has to be eliminated, and this elimination causes us to lose some other information as well. However, as stated above, the basis of the ECG (QRS-T segment) does not get affected by a 40Hz low-pass filter. Finally, we set a low-pass passive filter having a 40Hz cut-off frequency and tested it. It was spotted that 50Hz still passes with a considerable amplitude because of the low sharpness of the filters. It is crucial to know that setting a cut-off frequency to a certain number does not mean that the frequencies above that number will not pass. Higher the frequency, lower the amplitude, but the amplitude is never exactly zero. Thus, we decreased the cut-off to 20Hz. If Figure 1 is carefully checked, it can be seen that 20Hz low-pass filter does not cause an information loss for a QRST segment. After testing it, we determined that this fits the most in our application.

Then, we increased the order of the filter and analyzed the results. The responses of first, second and third-order low-pass passive filters (20Hz cutoff) are shown in figure 2.3.2.1, figure 2.3.2.2, and figure 2.3.2.3 respectively. Yellow signal is the input, where the blue signal is the output signal.

Figure 2.3.2.1: 1st order low passive filter response

Figure 2.3.2.2: 2nd order low pass passive filter response

Figure 2.3.2.3: 3rd order low pass passive filter response

After obtaining the responses of different order passive filters, we detected that the amplitude of the signal loses its energy more as the order increases. As can be seen, the amplitude drops to 86mV from 136 mV in the first-order filter, drops to 70mV from 140 mV in the second-order filter, and drops to 64mV from 152mV in the third-order filter. Moreover, another inference is that the signal loses its original shape after second and third-order filtering (refer to Figure 2.3.2.2 and Figure 2.3.2.3). To sum up, the advantages of the higher-order filters are less than their disadvantages in this case. Furthermore, we know that the decay in a simple first-order passive filter is theoretically -20dB/dec. The decay in an active filter is theoretically -40 dB/dec, which is considerably better than a passive filter. Thus, we set a Butterworth second-order active low-pass filter and simulated it. The circuit is shown in Figure 2.3.2.4. The cutoff frequency formula for a butterworth filter is as follows,

$$f_c = \frac{1}{2\pi RC}, \qquad R: 8040\Omega, C: 1uF,\ f_c = 19.79Hz$$

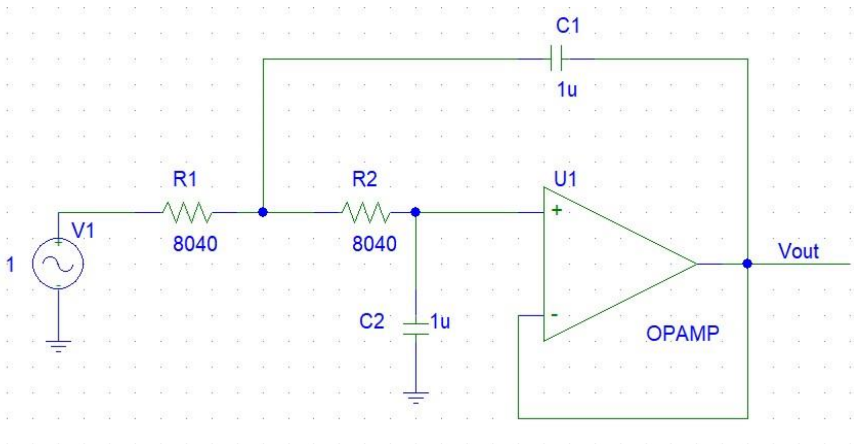

Figure 2.3.2.4: The low pass filter circuit

The theoretical and experimental frequency responses of this filter are shown in Figure 2.3.2.5 and Figure 2.3.2.6 respectively.

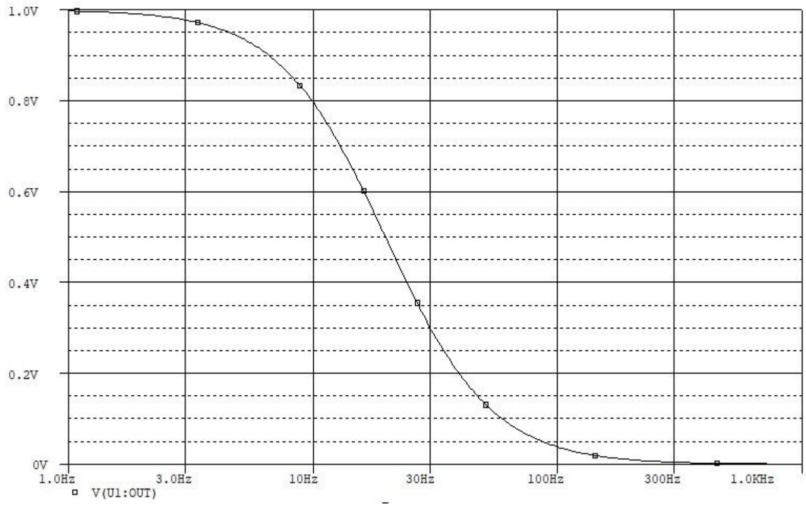

Figure 2.3.2.5: Theoretical frequency response (Amplitude (A) / Frequency (Hz))

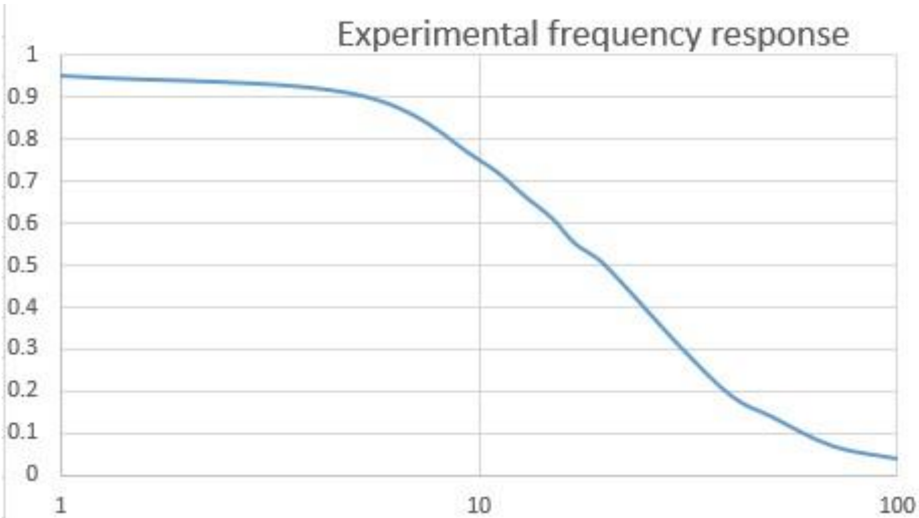

Figure 2.3.2.6: Experimental frequency response (Amplitude (V) / Frequency (Hz) graph)

To plot the experimental frequency response graph, we set the circuit and applied different inputs with different frequencies. By analyzing the transfer function ($\frac{V_{out}}{V_{in}}$) for these frequencies, the experimental result is plotted. In figure 2.3.2.6, we can see that the 50Hz noise is not totally filtered but amplified by a factor of 0.4. Because of this filtering, the amplitude of 50Hz will be so low and it will not spoil the original signal.

After the experiments, we directly added this filter to the high-pass filter to create a band-pass filter. Finally, we controlled the output of the band-pass filter. The filtered signal is displayed in figure 2.3.2.7.

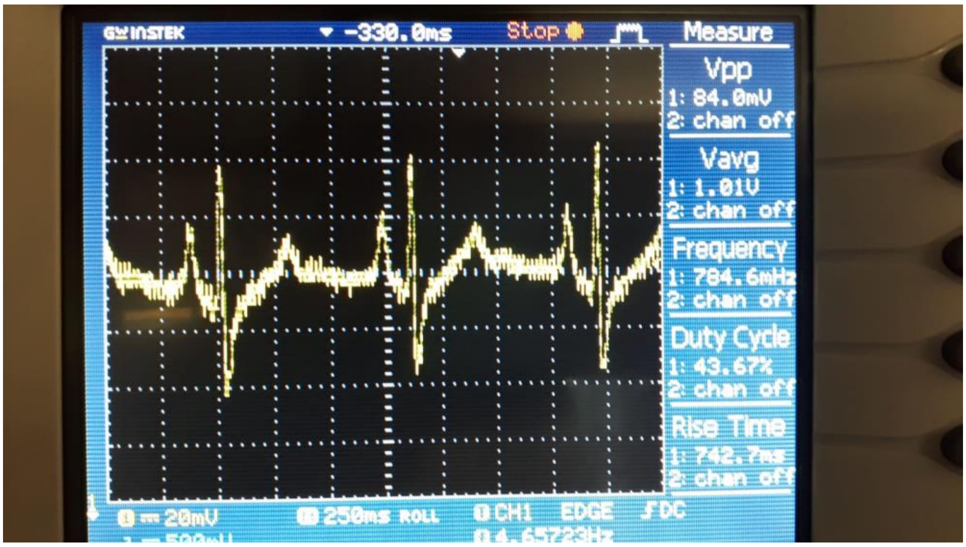

Figure 2.3.2.7: Output of the bans-pass filter

Here, the interpretation is that the signal still has the 50Hz noise in it. Other than that, it is well filtered. However, the decrease in amplitude is noticeable. In addition, the QRST segment is easily detectable.

2.4 AMPLIFIER

The signal loses its amplitude in the filtering section, and it needs to be amplified. For this reason, an amplifier circuit is set to amplify the signal. In this step, we made a selection out of several operational amplifiers. We preferred to use LF411. Its gain bandwidth does not create a limitation for our design and it has given consistent results. Consistency is crucial for a project. The other reason is that we did not get reasonable results after using the other op-amps such as LM741, LM7171, LM311P and LM386. However, it should be noted that these op-amps should have given coherent results.

The amplification factor has a crucial role for this project. To determine it, we checked the voltage limit of the analog Arduino inputs. We found out that the amplitude of the signal should be between 0V and 5V. We knew the approximate amplitude of the signal after the filtering. It was around 60mV-120mV. Thus, we set the amplification factor to 23. In this way, the amplitude of the amplified signal will be around 1.46V. This amplitude will also decrease after the notch filtering step, and at the end there will be a signal that has an amplitude of around 1V. We first planned to make the amplification in active filtering section, yet we could not manage to succeed it. After placing the resistors to the necessary terminals for amplification, we lost the signal in the active filtering section. This failure caused us to use another LF411 only for amplification. Next, we decided to use a non-inverting amplifier model. The following formulation is used to calculate the theoretical gain of the op-amp.

$$A = 1 + \frac{\mathrm{R1}}{\mathrm{R2}}$$

Here, R_1=1 kΩ & R_2=22 kΩ

We set the circuit on Pspice and simulated it. The circuitry is shown in figure 2.4.1, and the simulation response of the circuit is shown in figure 2.4.2.

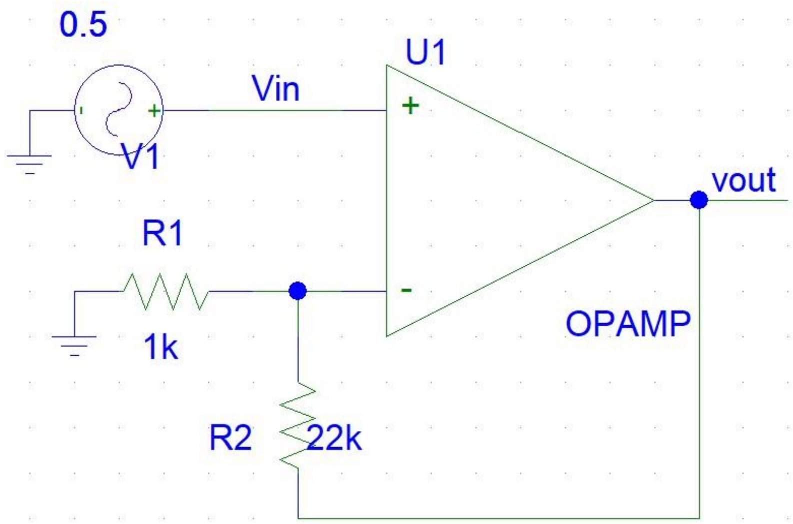

Figure 2.4.1: Amplifier circuit

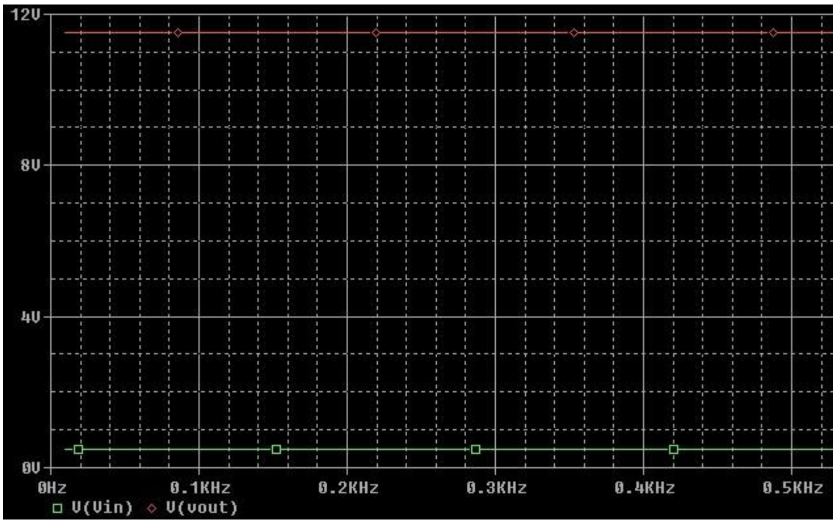

Figure 2.4.2: The response of the circuit. (Green line is for output (11.5V), blue line is for input (500mV))

In Figure 2.4.2, it can be seen that the input is 0.5V and the output is 11.5V. This simulation proves that the amplification factor is 23. Furthermore, the result of the amplifier circuit on the ECG signal is shown in Figure 2.4.3. The amplitude of the filtered signal is around 50-90mV, and the amplified signal has an amplitude of 1.46V. The experimental gain is around 18.5. It is not exactly 23, yet acceptable.

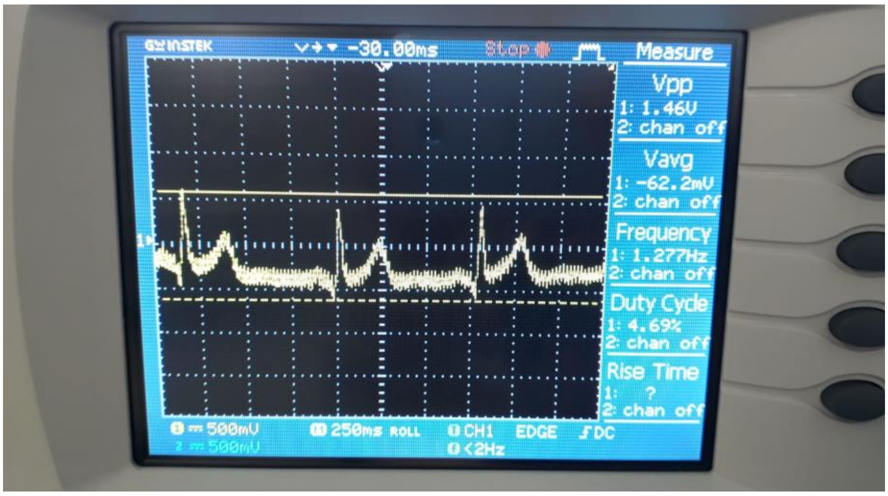

Figure 2.4.3: The amplification on the ECG signal

Furthermore, as can be seen in figure 2.4.3, even though we applied filters we could not totally get rid of 50Hz noise. We expected to face such a situation because it is practically impossible to totally get rid of a specific frequency. Hence, a notch filter is used to further filter the city line noise.

2.5 NOTCH FILTERING

Notch filtering applications are used when there is a noise at a specific frequency. For instance, the city line has a specific frequency and it needs to be filtered out. A passive notch filter is used to get rid of 50Hz. After analyzing the transfer function we detected that the stopped frequency is found by the following formula. Also the notch filter model we used is shown in Figure 2.5.1.

$$f_{stop} = \frac{1}{4\pi RC}$$

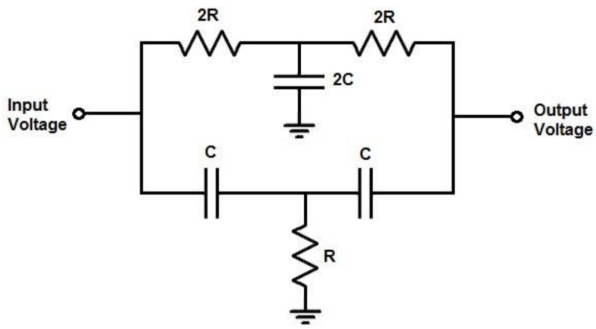

Figure 2.5.1: The notch filter circuitry

Here, we conducted several experiments and figured out that the R and C values are critical. Even though there are several R-C combinations that satisfy the f_{stop}, not all the combinations work effectively. After searching it online, we found the appropriate R-C combination for the notch filtering. The notch filter circuit is shown in Figure 2.5.2. The frequency response of the circuit is shown in Figure 2.5.3.

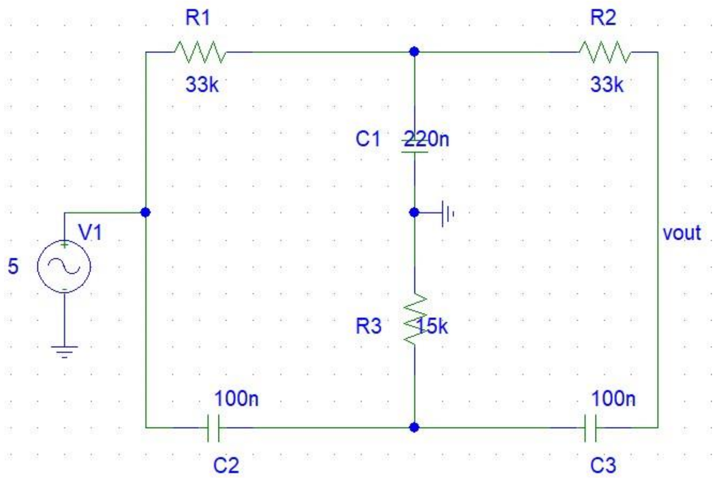

Figure 2.5.2: Notch filtering circuit

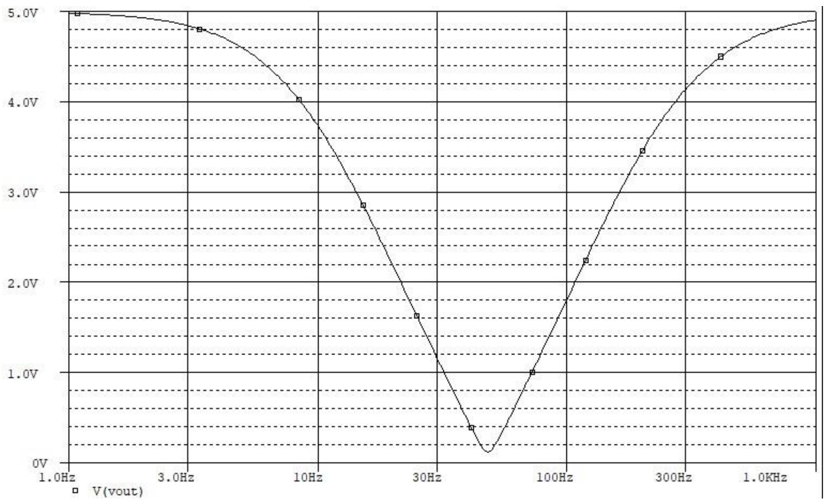

Figure 2.5.3: Theoretical frequency response of the notch filter (Amplitude (V) / Frequency (Hz))

After the simulation results, we conducted a real experiment by applying different inputs and analyzing the outputs. The real experimental results of the circuit are shown in Figure 2.5.4.

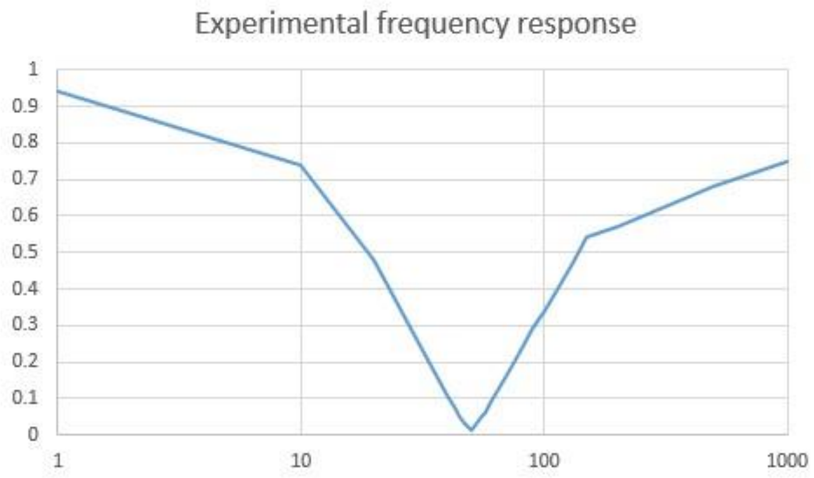

Figure 2.5.4: Experimental frequency response (Amplitude (V) / Frequency (Hz))

The amplitude is not exactly zero at 50Hz, yet it is significantly removed. Finally, we added this block to the offset addition block. The incoming ECG signal significantly lost its 50 Hz city line interference. The output of the notch filter is shown in Figure 2.5.5.

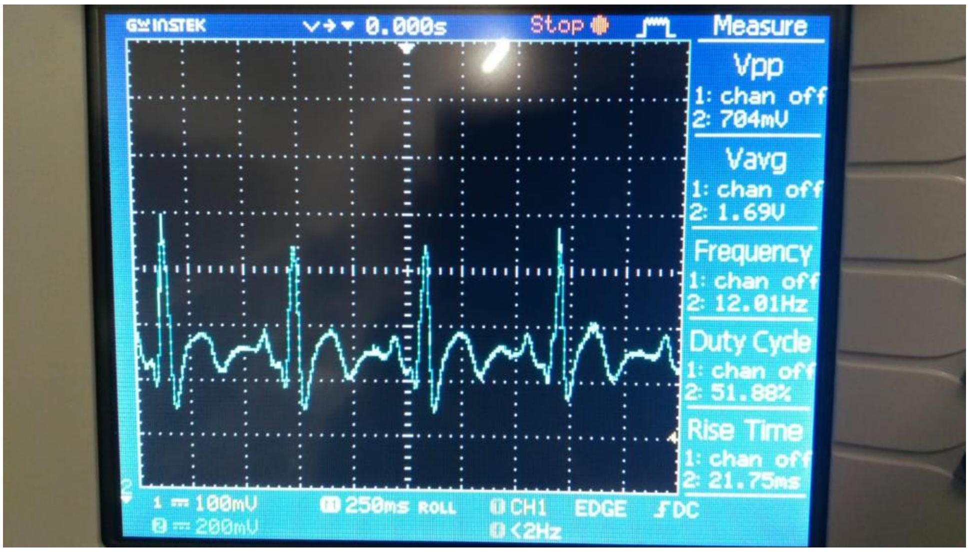

Figure 2.5.5: Output of the notch filter

This was also the very last process that we used in the hardware section. As can be seen, the ECG signal is pure and the QRST segment is easily detectable.

Important Notice: The notch filter filters out the 50Hz very effectively. However, we decided not to get the output from it so that we could see the filtering difference in MATLAB.

2.6 OFFSET ADDITION

The signal coming from the amplifier has a peak-to-peak voltage around 0.7V-1.5V. However, the amplitude is not always above zero. The analog to digital converter (Arduino) cannot sense the negative values. For this reason, an offset should be added to the signal so that all the values will be bigger than zero. In the oscilloscope, horizontal cursors are used to detect the lowest value. It was close to -0.5V. So, we decided to add a 2.2V offset. Note that the supply will be taken from 9V batteries. The method that we used to add this offset value was the voltage divider method. The formula for the voltage divider concept is as follows:

$$V_{offset} = V_1 \frac{R_2}{R_2 + R_1}$$

$$V_1 = 9V, R_1 = 680k\Omega, R_2 = 220k\Omega$$

In this configuration, the added offset voltage is 2.2V. The circuit is shown in Figure 2.6.1. Also, a capacitor is put between the ECG circuitry and the voltage divider section. The Capacitor does not allow dc voltage to go back to the circuit. It is crucial to block DC voltage coming from the voltage divider to go to the circuitry back. If a capacitor is not used, the DC voltage could go back to the circuitry and it could burn some components, spoil the signal and may even harm the patient. The input and output of the simulation are shown in Figure 2.6.2.

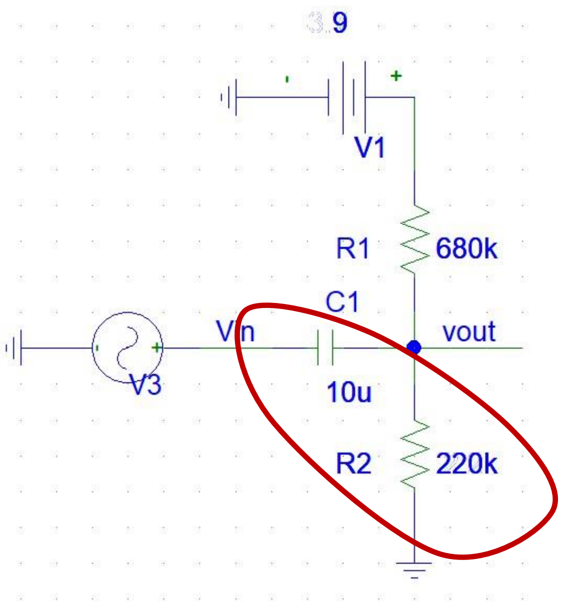

Figure 2.6.1: Voltage divider circuit

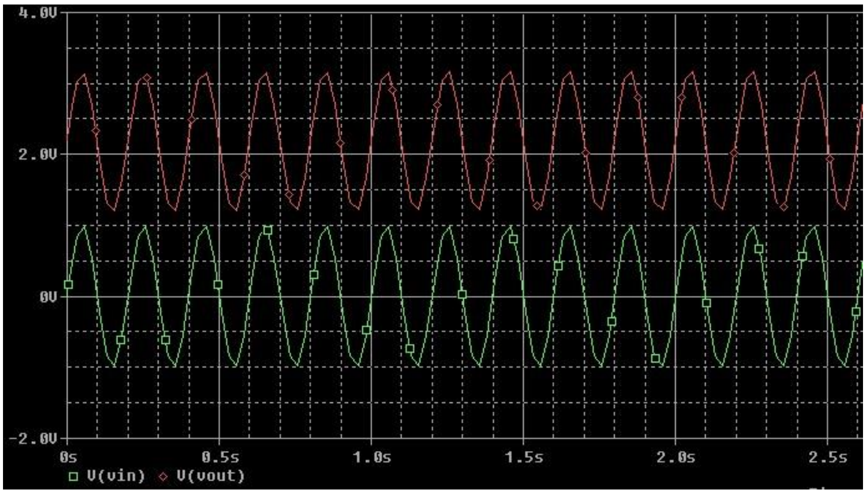

Figure 2.6.2: Input and output signals of simulation. (Green: Input, Red: Output)

Here, the voltage divider section also creates a first-order high-pass filter, circled in red line in Figure 2.6.1. The values of these passive elements are critical. If the cut off frequency of this high-pass filter is high enough, it can eliminate the information and deamplify the ECG signal. When 220kΩ and 10uF are applied, the cutoff frequency is 0.07 Hz which does not affect the original ECG signal. The 2.2V offset is added to the signal. Finally, the real offset adder result is shown in Figure 2.6.3. The horizontal cursors demonstrate the voltage levels.

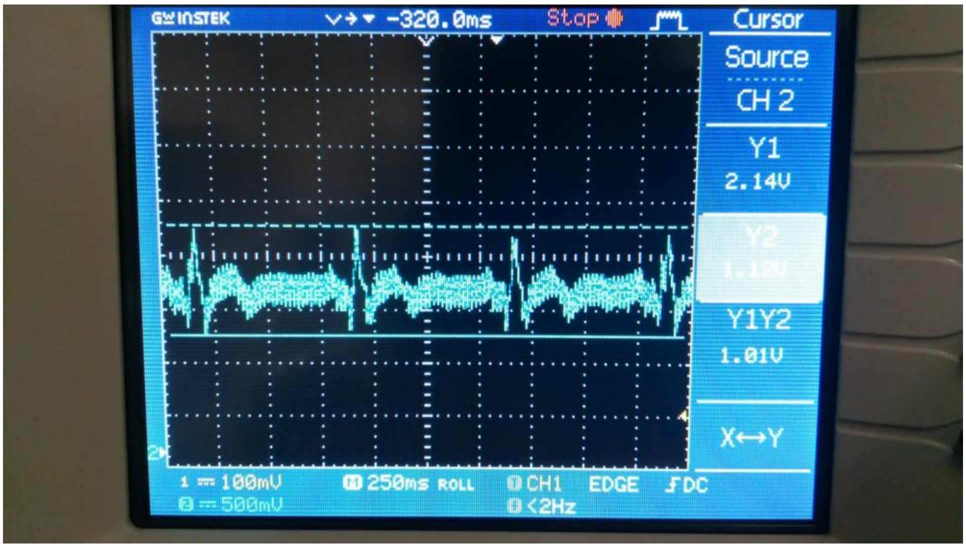

Figure 2.6.3: Real offset adder results (Y1= 1.12V, Y2=2.14V)

The signal is between 1.12V and 2.14V, which means that the Arduino can read the signal from now.

2.7 Analog to Digital Converter (ADC): Arduino UNO

After all the modifications the signal should be sent to a computer. The ADC converters are used for this purpose. We used the Arduino UNO as ADC. The signal just before the ADC went through a couple of processes. Filtering, amplification, and offset addition modified the incoming raw ECG data so that the signal can be read by the ADC and sent to the computer. Most critical detail for Arduino UNO is that it does not read the values below 0V. Its band is between 0V and 5V. All these limitations were taken into account in the filtering, amplification, and offset adder section. A sample of real-time plot of Arduino serial plotter is shown in Figure 2.7.1.

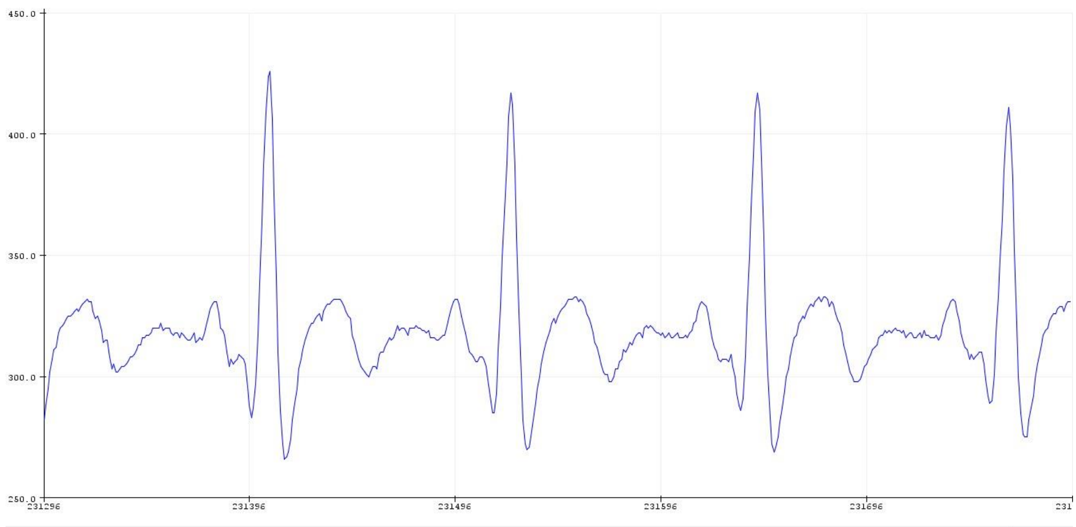

Figure 2.7.1: A sample of real time ECG graph in Arduino serial plotter

The baud rate is set to ‘38400’, and it has lead the sampling frequency to be around 770. The term *Sampling Frequency* is significantly important at this point. It simply denotes the number of data sent to the computer in one second, and it should not be less than two times the maximum frequency. Otherwise, aliasing occurs and the process accuracy decreases in future steps. For instance, 770 samples per seconds mean that the highest frequency component that can be displayed is 335Hz, which is enough for the ECG signal. By choosing the baud rate 38400, the aliasing is prevented. For further information, please refer to *Nyquist Theorem*.

3. MATLAB PROCESSING

All the hardware parts are used so that the signal could be easily processed in MATLAB. MATLAB is used to detect the R peaks and calculate the Beat-per-minute rate of a human. To do this, the raw ECG data is further filtered in MATLAB in several different methods such as Butterworth, Chebyshev and Elliptic filtering methods. Soft filtering is much easier and better than hard filters. The desired cut-off frequency, method and the order of the filter can be changed by the user. After soft filtering, the BPM is calculated by MATLAB. In this part, each step is explained in detail.

3.1 Obtaining the Raw ECG DATA

The raw ECG data is sent from Arduino UNO to MATLAB. The Raw ECG data is displayed in Figure 3.1.1.

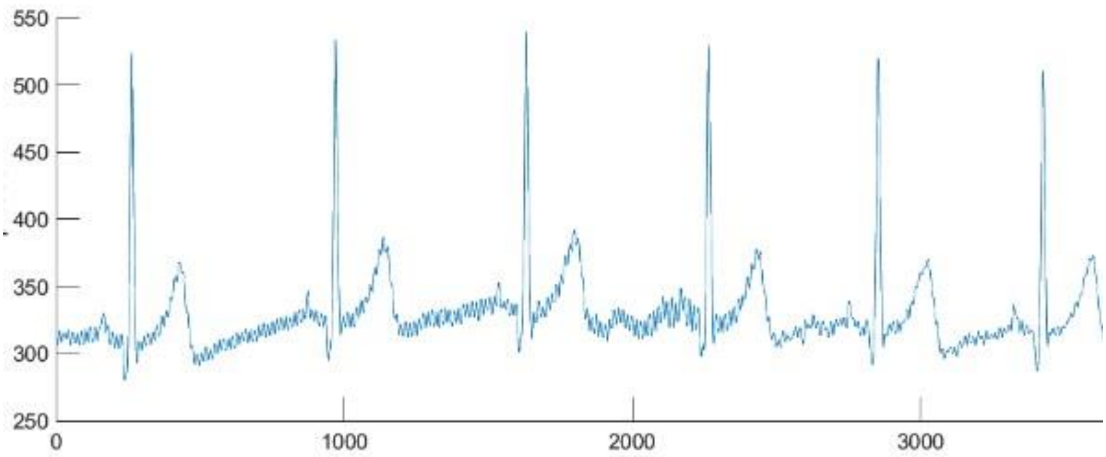

Figure 3.1.1: Raw ECG signal in MATLAB

This ECG signal slides all the time and gives the real-time results. The speed of this sliding movement depends on the baud rate. Furthermore, MATLAB is also useful for the Fourier Transformation of a given signal. By using its FFT functions, frequency components of the given ECG signal can be shown. The FFT of the Raw ECG data in Figure 3.1.1 is shown in Figure 3.1.2.

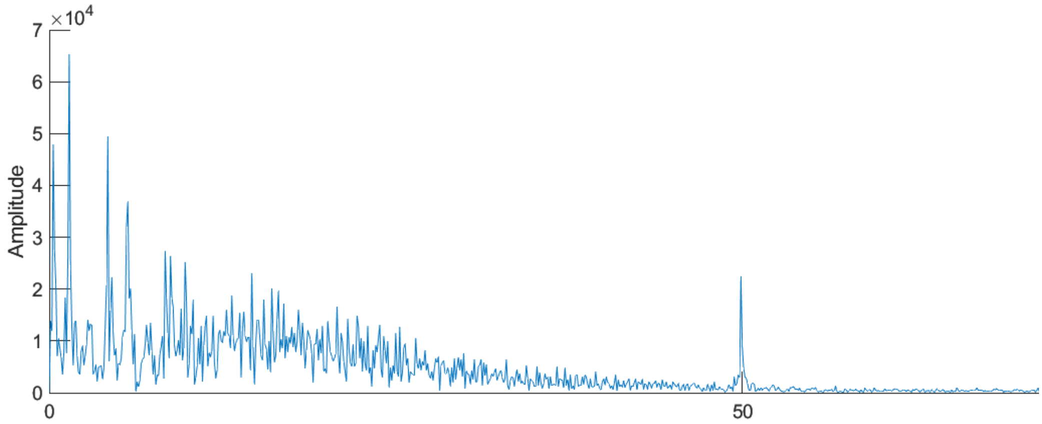

Figure 3.1.2: Fourier Transformation of the RAW ECG signal

As can be seen in Figure 3.1.2, there is a peak at exactly 50Hz due to the city line noise. This is an expected situation. The amplitude of the noise at 50Hz was even more, yet the hard filtering applications decreased its amplitude to this level. High orders of soft filters will decrease it even more.

3.2 Filtering in MATLAB

There are several filtering applications in MATLAB. We preferred to use only three of them: Butterworth, Chebyshev, Elliptic. All of these filters are applied to the signal when desired in this project.

The cut-off frequency determination is important. In MATLAB filter functions, the cut-off frequency is based on the normalized frequency, meaning that the input should be between *zero* and *one*. Here, *one* stands for the half of the sampling frequency also called as Nyquist frequency. Then, The desired cut-off frequency is calculated accordingly. For all the figures below, the cut-off frequency for the high-pass filter is 1Hz, whereas the cut-off frequency for the low-pass filter is 20Hz, and the filter orders are two. The Butterworth filtered ECG signal in time domain is displayed in Figure 3.2.1, the Fourier transformation of it is displayed in 3.2.1.

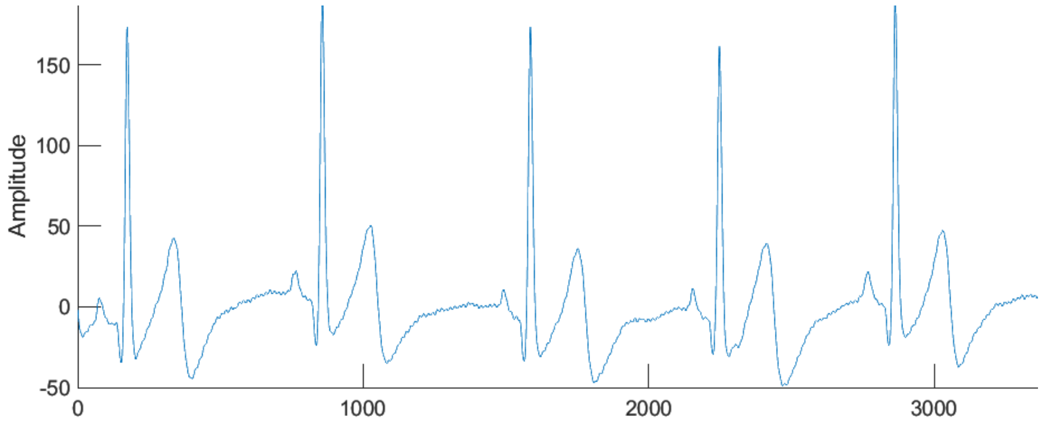

Figure 3.2.1: Butterworth filtered signal in time domain

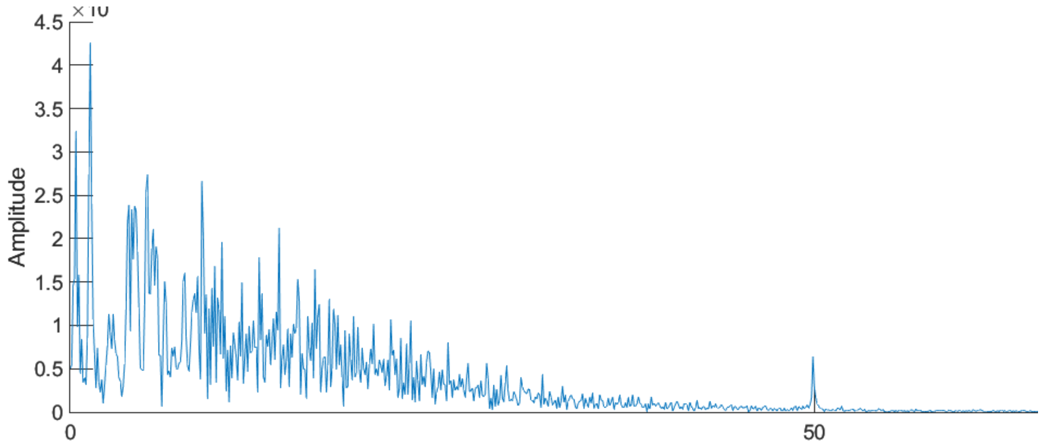

Figure 3.2.2: Butterworth filtered signal in frequency domain

Butterworth filter filtered the desired frequencies out as expected. As can be seen in Figure 3.2.2, the amplitude of the 50Hz component is lower compared to unfiltered signal (refer to Figure 3.1.2). The results of Chebyshev filter in time domain and frequency domain are displayed in Figure 3.2.3 and Figure 3.2.4 respectively.

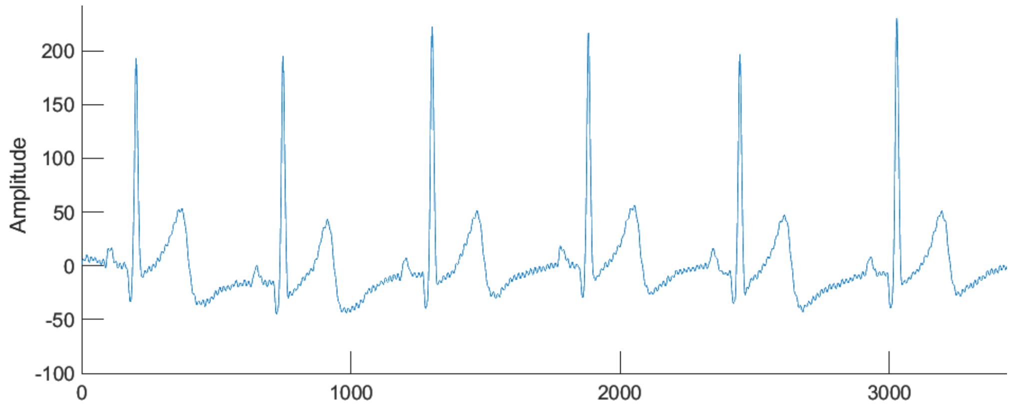

Figure 3.2.3: Chebyshev filtered signal in time domain

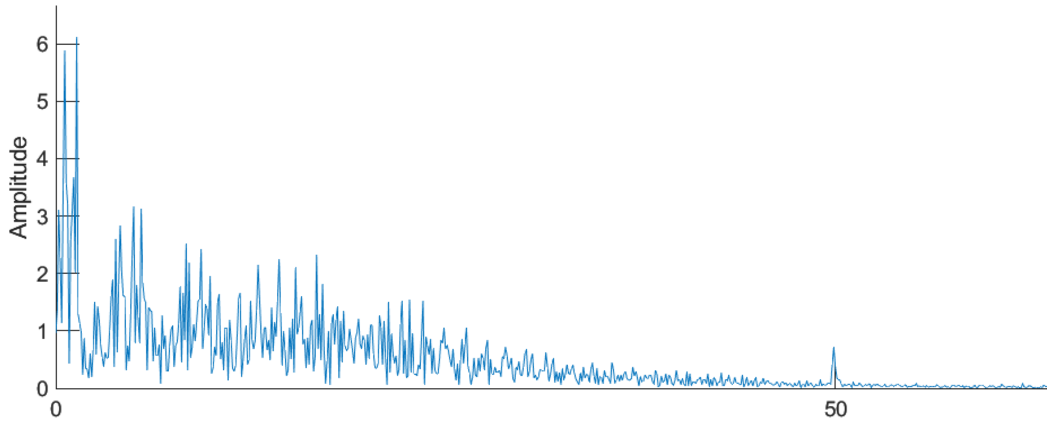

Figure 3.2.4: Chebyshev filtered signal in frequency domain

The results of Elliptic filter in time domain and frequency domain are displayed in Figure 3.2.5 and Figure 3.2.6 respectively.

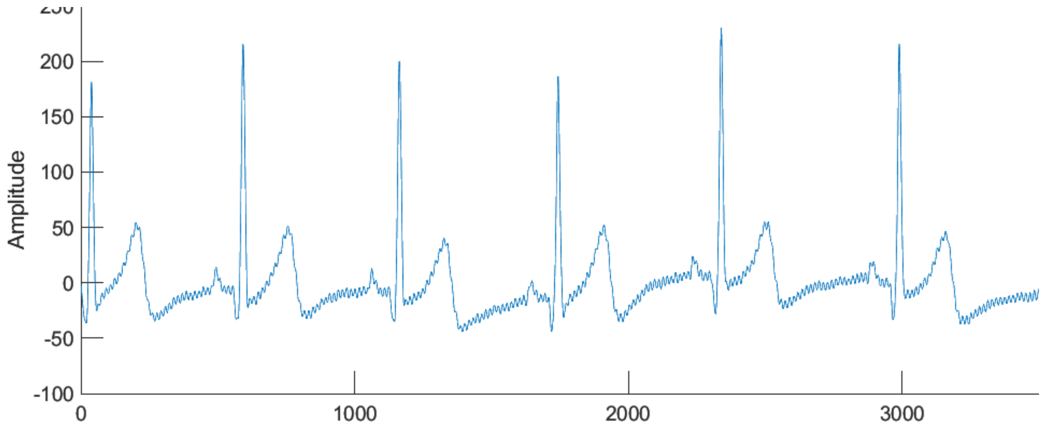

Figure 3.2.5: Elliptic filtered signal in time domain

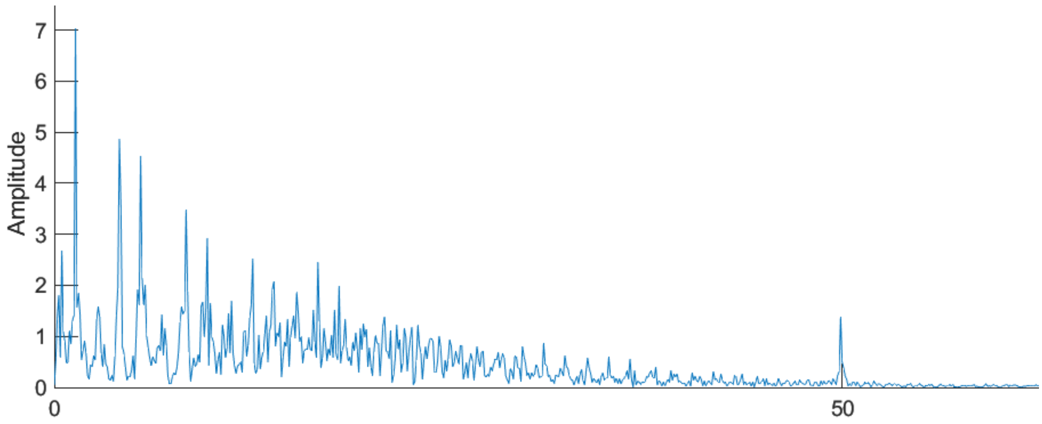

Figure 3.2.6: Elliptic filtered signal in frequency domain

All the filters worked as expected and decreased the amplitude of the unwanted frequency components. Here, note that if the order of the filters were increased, the sharpness of the filter would increase and eliminate the unwanted frequencies more effectively. However, if the order is increased to a very high number such as 10, the signal gets spoiled. Thus, the order should not be a big number if not needed.

3.3 Beat-per-minute (BPM) Calculation

BPM calculation is the last step for this project. Until now, the raw ECG signal has been through a couple of processes so that the BPM can be calculated more accurately. There are plenty of algorithms to calculate the BPM rate.

First, a new 24000 samples array is created using the filtered signal arrays, and we take the cube of these values. Next, an appropriate threshold value is set. Each value in the array is checked depending on this threshold value. There are two types in this case: the ones that are bigger than the threshold, and the ones that are less than the threshold. The values that are less than the given threshold are set to *zero*. After this step, the function '*find peaks*' in MATLAB is used to detect the peaks. In this way, the number of peaks in 24000 samples array is found. We also know the time needed for creating a 24000 array. Then the number of peaks in one minute is calculated depending on this value. Here, an important notice is that the array continuously slides and takes new values, and calculates the BPM again.

4. Final Design

4.1. Soldering

Soldering is very important because it solves most of the disconnection issues. It also minimizes the size of the design. To be able to come up with a commercially available design, the components should be on either PCB or a stripboard. Our first intention was to use a PCB, yet the PCB machine did not print it as we designed it in Altium Designer. Thus, we decided to using soldering method. The soldered version of the last product is shown in Figure 4.1.1.

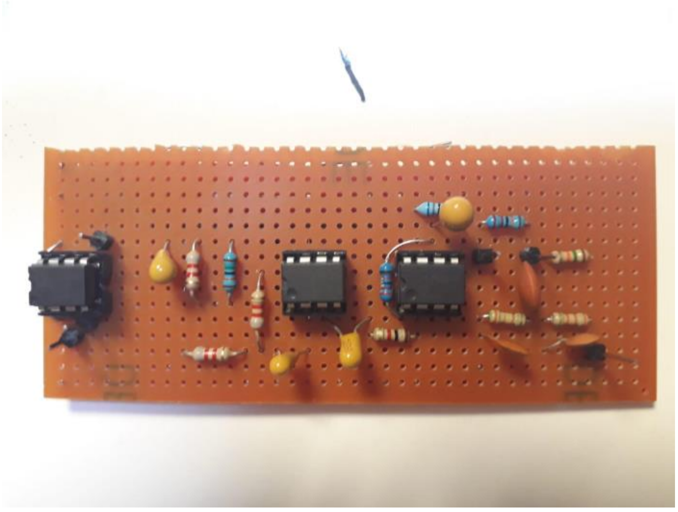

Figure 4.1.1: The soldered version of the design

During the soldering process, we soldered and tested each part one by one, and tested them separately. After making sure that the parts are correctly soldered, we added them one to other and finalized this step. There are two outputs in this design. One is before notch filter, and the other is after notch filter. As mentioned in 2.5 (refer to notch filtering), the notch filter filters the signal very well and it would not be meaningful for us to filter it in MATLAB again.

Conclusion & Discussion

In this project, we aimed to obtain the heart signal from a human body, apply several processes on it, and find the beat-per-minute rate. The processes were filtering, amplification, offset addition, analog to digital conversion, filtering in MATLAB, and using BPM calculation algorithms in MATLAB. It was very beneficial for us to understand the signal-processing concepts for both hardware and software.

To be able to successfully finalize all the steps, we first learned the theory behind each step. Then, we simulated these steps separately using Pspice or other online tools. After obtaining the expected simulation results, we started to proceed and conduct the real tests. However, during the real tests, we faced unexpected results sometimes. For instance, even though all the calculations and simulations worked as expected, we could not get an output sometimes. Nonetheless, we knew that the theory behind it was correct. We believe that the reason for those unexpected results is the connection issues. Working on a breadboard leads to connection error sometimes. Also, the jumper cables may be a cause of disconnection. To summarize, we set all the parts separately and tested them. After obtaining the expected results, we put them together and tested the whole system several times. We faced several troubles. For example, the notch filtering part did not work at the first try, and we changed the capacitors and resistors. It worked after it. We solved each and every problem we faced by applying several different approaches. The last system on the breadboard is working exactly as expected after all the corrections. However, as mentioned above, disconnection errors may occur anytime on a breadboard. Moving all the parts to the PCB would solve this issue, yet the PCB machine we use did not print it as we designed it in *Altium Designer*. Thus, we decided to use a stripboard.

References:

1) Guldenring, D., Finlay, D., Bond, R., Kennedy, A., Mclaughlin, J., & Moran, K. (2016). The Effects of 40 Hz Low:pass Filtering on the Spatial QRS:T Angle. 2016 Computing in Cardiology Conference (CinC). doi: 10.22489/cinc.2016.030-517
2) Buenda-Fuentes, F., Arnau-Vives, M., & Arnau-Vives, A (2012). High-Bandpass Filters in Electrocardiography: Source of Error in the Interpretation of the ST Segment. ISRN Cardiology.
3) https://www.researchgate.net/figure/50-Hz-twin-T-passive-notch-filter-circuit_fig8_282404009
4) Heart.org