DEPARTMENT OF ELECTRONIC AND TELECOMMUNICATION ENGINEERING
UNIVERSITY OF MORATUWA

Group Project Report

EN1093 - Laboratory Practice

Portable Heart Rate Monitor

Authors:	*Registration Number:*
K.K. Herath	170213V
H.M.U.D Herath	170215U
R.U. Hettiarachchi	170221T
M.N. Hettiaratchchi	170222X

This is submitted as a partial fulfillment for the module EN1093

January 7, 2019

1 Abstract

The proposed project measures the heart rate of a person using optical sensors. The optical sensor detects the variation of blood volume at the fingertip. In our project the sensor will be an infrared light emitting diode (IR LED) which will be on the same side of the finger. The underlying principle is that the intensity of the reflected infrared light varies on the blood volume at an instance of the fingertip, which changes proportionately to each cardiac cycle. The lighting condition in the environment is very important to avoid distortion in the signal so we have taken the following measures,

• To design a special enclosure to cover the optical sensor for accurate measurements

• To use a band filter to remove unnecessary wavelengths.

Based on the refined output signal, we used an Atmel microcontroller to calculate and display the heart rate through the LCD Display.

2 Acknowledgement

We would like to express our gratitude to our supervisor Mr.Asanka Rathnayake for giving us technical advice and guidance.
Our sincere thanks goes to Mr.Thilina Sameera Ambagahawaththa for the inspiration and advice given to us to compile this report in LaTeX.

We pay our gratitude to all the lecturers, instructors and other academic staff who intimately welcomed us to share their knowledge and experiences. We are very grateful to the personnel who are in charge of laboratories for allowing us to use the laboratories when needed and for the support given to solve technical problems.

Contents

3 Introduction

Heart rate is the number of times one's heart beats per minute.The portable heart rate monitor is a personal monitoring device that measures the heart rate in real time. For an average person this value lies between 60-100 bpm. The heart rate rises gradually during excercises and returns slowly to the rest value after excercise. The rate when the pulse returns to normal is an indication of fitness in a person. Therefore a heart rate monitor is an essential device nowadays to keep in track with your body not only for someone who is suffering with heart diseases but also for a normal person to maintain fitness. Having a portable heart rate monitor makes it easy to carry it around when travelling.
In this project reflective photoplethysmogram(PPG) is used for heart rate monitoring. These sensors uses a light based technology to sense the rate of blood flow as controlled by the heart's pumping action. This method is less sensitive to motion artifacts and therefore is user friendly.

4 Methodology

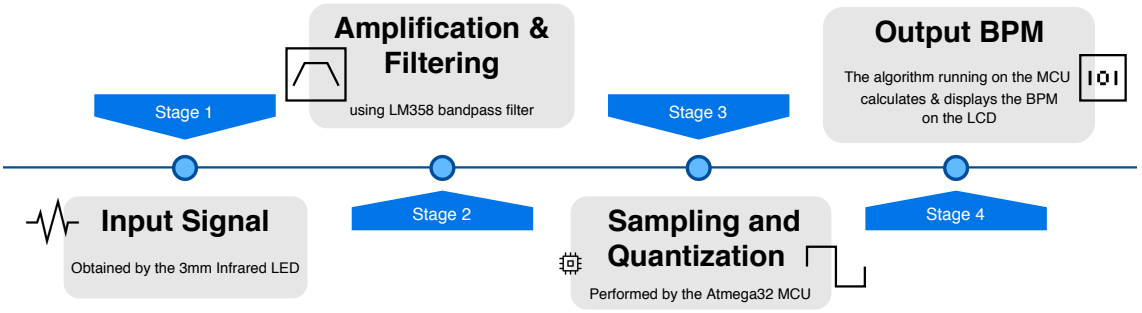


Figure 1: Methodology in a nutshell

4.1 Sensor - (principle)

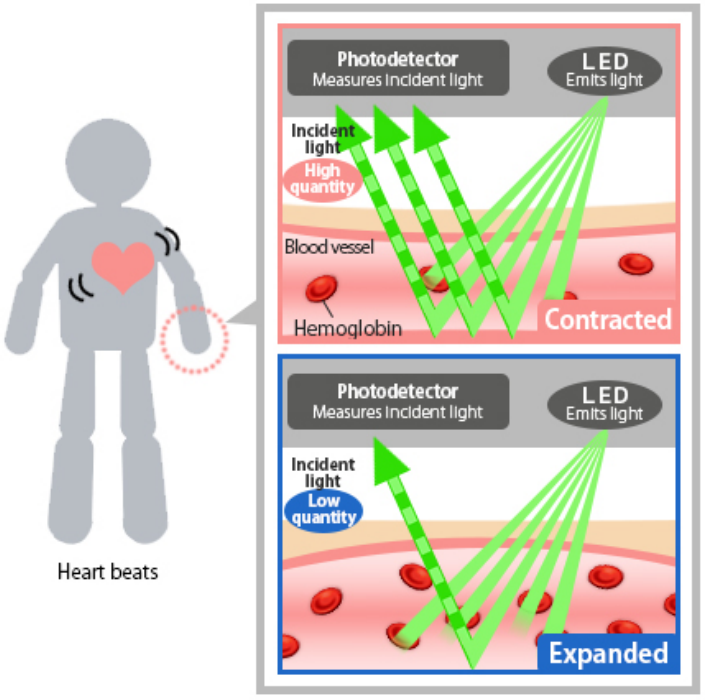


Figure 2: How IR LED light is reflected from the blood vessels[5]

4.2 Amplification and Filtering

To amplify the PPG signal we designed an amplifier circuit. However we observed that without proper filtering we cannot get a clear signal due to noise. So we had to determine the necessary values for the filter circuit.
The typical heart rate of an adult is between 60-100bpm[4].In frequency terms, these values correspond to the range 1-1.7 Hz.So an active bandpass filter circuit was designed to provide the following characteristics using LM358.

4.2.1 Calculations

$$\text{Cutoff frequency } (f) = \frac{1}{2\pi RC}$$

$$\text{Upper cutoff frequency} = \frac{1}{2\pi(680\ k\Omega\ )(100\ nF)} = 2.34051 \text{ Hz}$$

$$\text{Lower cutoff frequency} = \frac{1}{2\pi(47\ k\Omega)(4.7\ \mu F)} = 0.720484 \text{ Hz}$$

So we have a,

- Gain of 10
- Lower Cutoff Frequency of 0.72 Hz
- Upper Cutoff Frequency of 2.34 Hz

To have a low powered operational amplifier we selected LM358 integrated circuit.

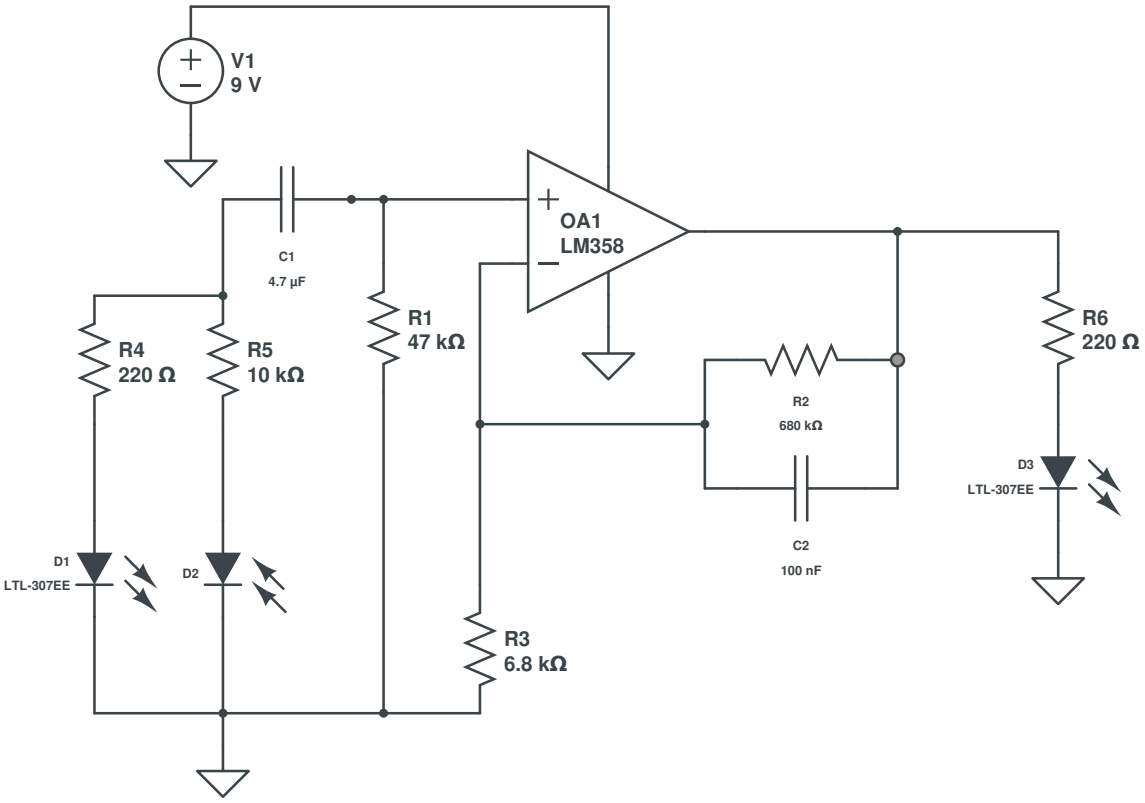


Figure 3: Active bandpass filter for the pulse sensor

4.3 Calculating BPM from the analog signal

The ATMega32 microcontroller was used to sample the analog signal to 0-1023 voltage levels. Then an algorithm was developed to identify peaks and thereby calculate the beats per minute. We used Atmel Studio to compile and flash the hex file to the microcontroller.

4.3.1 Microcontroller

The ATmega32 is a low power high performance atmel AVR 8-bit microcontroller.

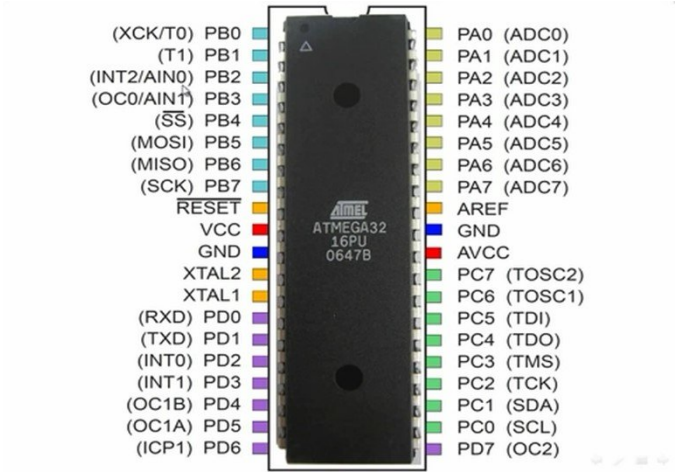
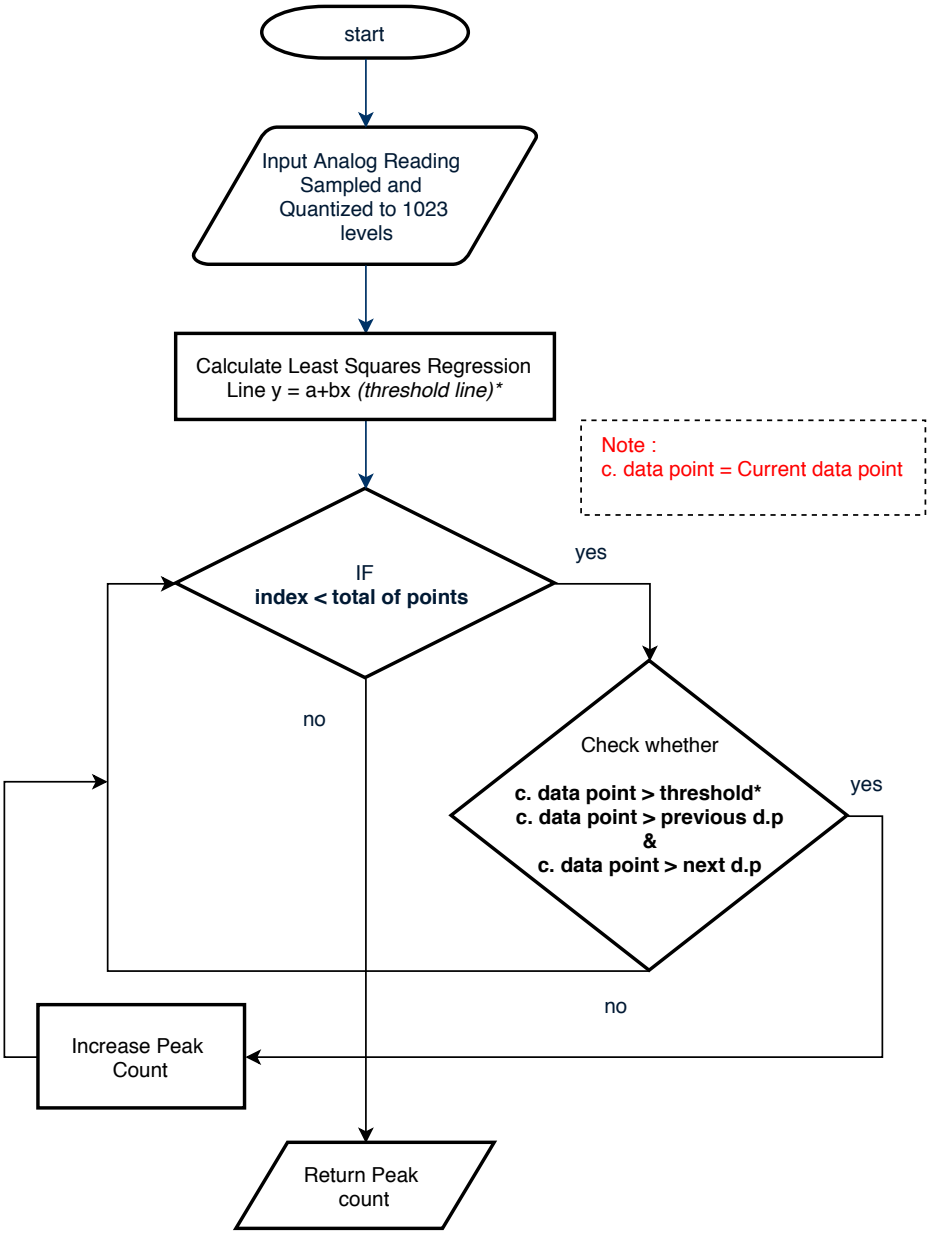


Figure 4: ATmega32 pinout

4.3.2 Obtaining a threshold value



The above flowchart explains the peak counting algorithm with automatic threshold calculation. For calculating the threshold we're calculating the least squares regression line.

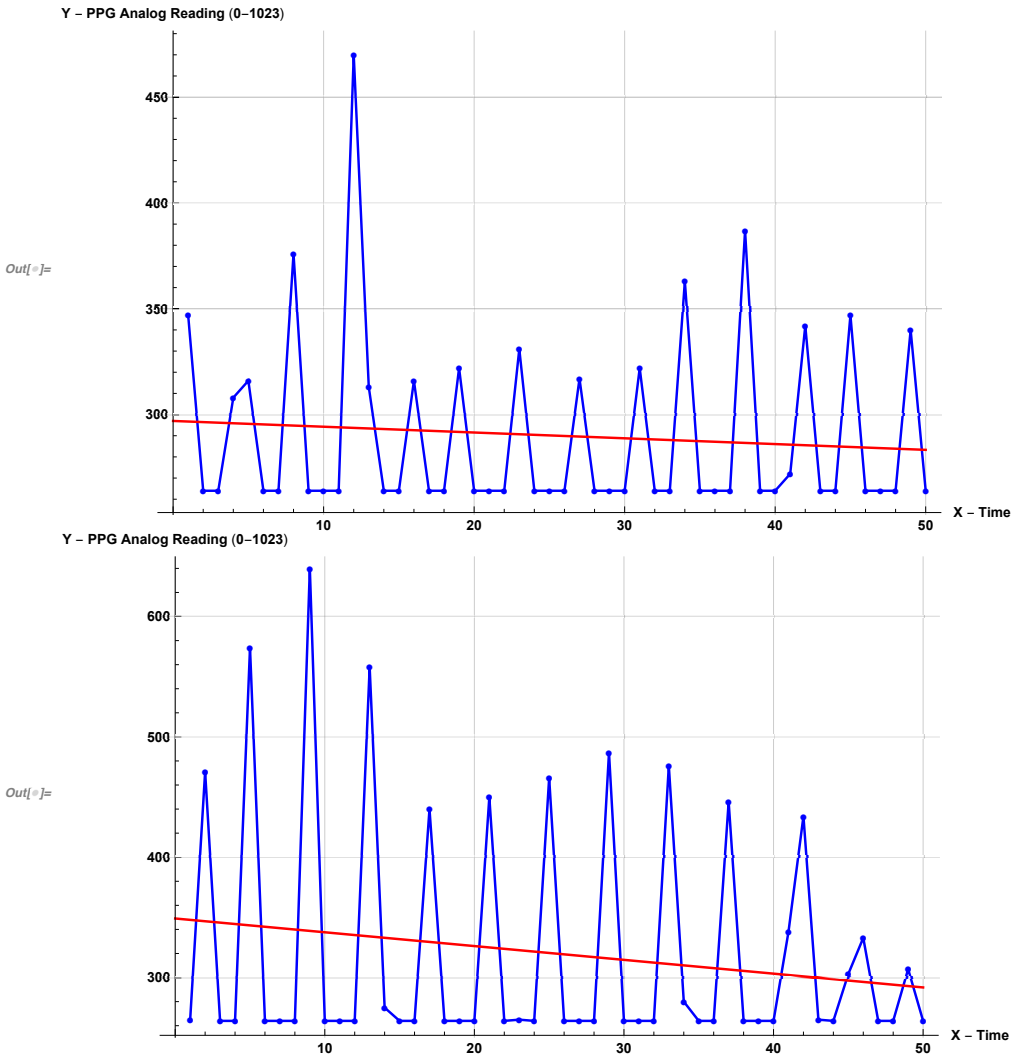


Figure 5: Determining threshold value line using simple linear regression

In the above graphs, we've presented two data sets of two people and how the threshold line(Red) was obtained.

4.3.3 C Code

```
1       /*  Group 7 | UoM | ENTC 17Batch
2       Complete Code - with debugging notes */
3       #ifndef F_CPU
4       #define F_CPU 16000000UL // 16 MHz clock speed
5       #endif
6
7       #define D0 eS_PORTD0
```

```
8       #define D1 eS_PORTD1
9       #define D2 eS_PORTD2
10      #define D3 eS_PORTD3
11      #define D4 eS_PORTD4
12      #define D5 eS_PORTD5
13      #define D6 eS_PORTD6
14      #define D7 eS_PORTD7
15      #define RS eS_PORTC6
16      #define EN eS_PORTC7
17
18      #include <avr/io.h>
19      #include <util/delay.h>
20      #include "stdlib.h"
21      #include "string.h"
22      #include "lcd.h"
23
24      char disp[16]="0000000000000001";
25      char result[8] = "00000001";
26
27
28      void lcd_disp(char data_points[],int r,int c,char w[]){
29
30         if(w=="clear")Lcd8_Clear();
31         Lcd8_Set_Cursor(r,c);
32         Lcd8_Write_String(data_points);
33      }
34
35      void ADC_Init(){
36         DDRA=0x0;        /* Make ADC port as input */
37         ADCSRA = 0x87;     /* Enable ADC, fr/128 */
38         ADMUX = 0x40;      /* Vref: Avcc, ADC channel: 0 */
39      }
40
41      int ADC_Read(char channel){
42         int Ain,AinLow;
43
44         ADMUX=ADMUX|(channel & 0x0f); /* Set input channel to read */
45
46         ADCSRA |= (1<<ADSC);  /* Start conversion */
47         while((ADCSRA&(1<<ADIF))==0); /* Monitor end of conversion
               interrupt */
48         _delay_us(10);
49         AinLow = (int)ADCL;   /* Read lower byte*/
```

```
50          Ain = (int)ADCH*256;  /* Read higher 2 bits and
51                      Multiply with weight */
52          Ain = Ain + AinLow;
53          return(Ain);       /* Return digital value*/
54       }
55
56       /*************** threshold ***************/
57       double m=0,c=0; // gradient and slope
58
59       double sumit(int data_points[],int length){
60           int i;
61           double sum=0;
62           for(int i=1;i<=length;i++)sum+=data_points[i];
63
64           return sum;
65       }
66
67       double xysum(int data_points[],int length){
68           int i;
69           double sum=0;
70           for(int i=1;i<=length;i++)sum+=(i)*data_points[i];
71
72           return sum;
73       }
74
75       void regression(int data_points[],double n){
76          double squarex= (n)*(n+1)*(2*n+1)/6.0;
77          double xbar = (n+1)/2.0;
78          double ybar = sumit(data_points,n)/n;
79
80          //printf(" xbar=%f, ybar=%f, squarex=%f, xysum
               =%f\n",xbar,ybar,squarex,xysum(data_points,n));
81
82          m=(xysum(data_points,n)- n*xbar*ybar)/( squarex - n*xbar*xbar
               );
83          c=ybar-m*xbar;
84       }
85       /*************** end threshold ***************/
86
87
88       int main(void){
89          DDRD = 0xFF; // #
90          DDRC = 0xFF; //for lcd
```

```
91          DDRA = 0x00; //Analog input
92
93
94          ADC_Init();
95          //ADMUX = 0b01100000; // Configure ADC to be left justified,
                use AVCC as reference, and select ADC0 as ADC input
96          //ADCSRA = 0b10000111; // Enable the ADC and set the
                prescaler to max value (128)
97
98
99          Lcd8_Init(); //Initializing the LCD screen
100         lcd_disp("Starting Pulse ~)",1,0,"");
101         lcd_disp("Sensor *_*",2,0,"");
102
103         _delay_ms(1000);
104
105         lcd_disp("Place your",1,0,"clear");
106         lcd_disp("fingertip",2,0,"");
107         _delay_ms(2000);
108
109         lcd_disp("Keep placing ~",1,0,"clear");
110         lcd_disp("your fingertip",2,0,"");
111
112         _delay_ms(3000);
113         int i;
114         int thresh=300;
115         int count=0;
116         int count2=0; //modified peak counting algo
117
118
119         /*  timing data */
120            double sampling_rate = 0.100 ;     // actually this is the
                   _delay_ms val
121            int time_limit = 10 ;          //in seconds
122            int se=time_limit/(sampling_rate*2);
123            int data_points[se+1];
124
125         data_points[0]=0;
126         /* /timing data */
127
128
129
130         /* for debugging purposes -> h and l records the peaks */
```

```
131         int h=0;
132         int l=1023;
133
134         char val[6]; //temporary variable for itoa
135
136
137         for(i=0;i<time_limit/(sampling_rate*2);i++){
138            data_points[i+1]=ADC_Read(0);
139
140            itoa(data_points[i+1],val,10);
141            //lcd_disp(val,1,0,"");
142            lcd_disp("Measuring.. *_*",1,0,"clear");
143
144            int k=16*(data_points[i+1]-200)/375;
145
146            char anim[16]="";
147
148            int h=0;
149            for(h=0;h<k;h++){
150               anim[h]='~';
151            }
152            itoa(k,val,10);
153            lcd_disp(anim,2,0,"");
154            _delay_ms(sampling_rate*1000);
155
156         }
157         /*
158         for(i=0;i<10;i++){
159            lcd_disp("*_*,",5,0,"clear");
160            _delay_ms(500);
161            lcd_disp("0_0,",5,0,"clear");
162            _delay_ms(500);
163            lcd_disp("*_-,",5,0,"clear");
164            _delay_ms(500);
165         }
166       */
167
168         regression(data_points,se);
169       /*    Debug code below
170         lcd_disp("regression",1,0,"clear");
171
172
173         char out[16]="";
```

```
174          itoa(m*1000,val,10);
175          strcat(out,"m= ");
176          strcat(out,val);
177          strcat(out," ; c= ");
178          itoa(c*1000,val,10);
179          strcat(out,val);
180          lcd_disp(out,2,0,"");
181          _delay_ms(5000);
182       */
183          //recorded wave form
184          for(i=0;i<time_limit/(sampling_rate*2);i++){
185             thresh=(i+2)*m+c;
186
187             int a=data_points[i+1];//current value
188             char temp[11]="";
189             char ccount[3];
190
191             if(a>h)h=a; //max peak
192             if(a<l)l=a; //min peak
193
194             if(a>thresh){
195                count+=1; //peak counting]
196                if(a>data_points[i] && a>data_points[i+2]){
197                   count2+=1;
198                }
199
200             }
201             /* debug code
202
203                itoa(count,ccount,10);
204                itoa(a,val,10);
205
206
207             strcat(temp,val);
208             strcat(temp," - ");
209             strcat(temp,ccount);
210             itoa(count2,ccount,10);
211             strcat(temp,",");
212             strcat(temp,ccount);
213             lcd_disp(temp,1,0,"clear");
214             _delay_ms(100); */
215
216          }
```

```
217           //displaying statistics of the waveform
218         /* char peaks[10]="H- "; //maximum value
219              itoa(h,val,10);
220           strcat(peaks,val);
221           strcat(peaks,"; L- "); //minimum value
222              itoa(l,val,10);
223           strcat(peaks,val);
224           lcd_disp(peaks,1,0,"");
225
226           */
227
228
229           //displaying the heart rate
230           char bpm[10]="Heart Rate : ";
231
232           /*itoa(count*(60/time_limit),val,10);
233
234           strcat(bpm,val);
235           strcat(bpm," ; N-");
236
237           */
238           lcd_disp(bpm,1,0,"clear");
239
240           itoa(count2*(60/time_limit),val,10);
241
242           strcat(val," bpm");
243           lcd_disp(val,2,4,"");
244
245
246         }
```

Based on this [1] and this data-sheet [3]

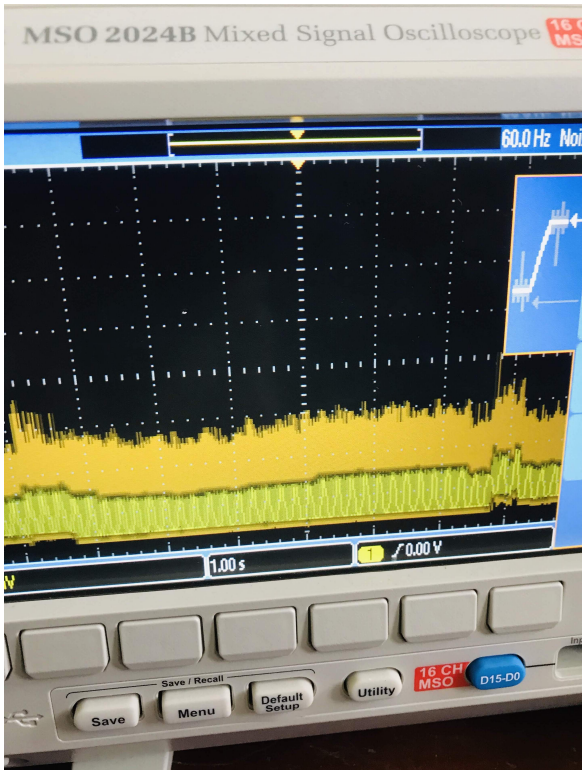
4.4 Enclosure and PCB Design

5 Implementation and Results

5.1 The effect of noise on the signal

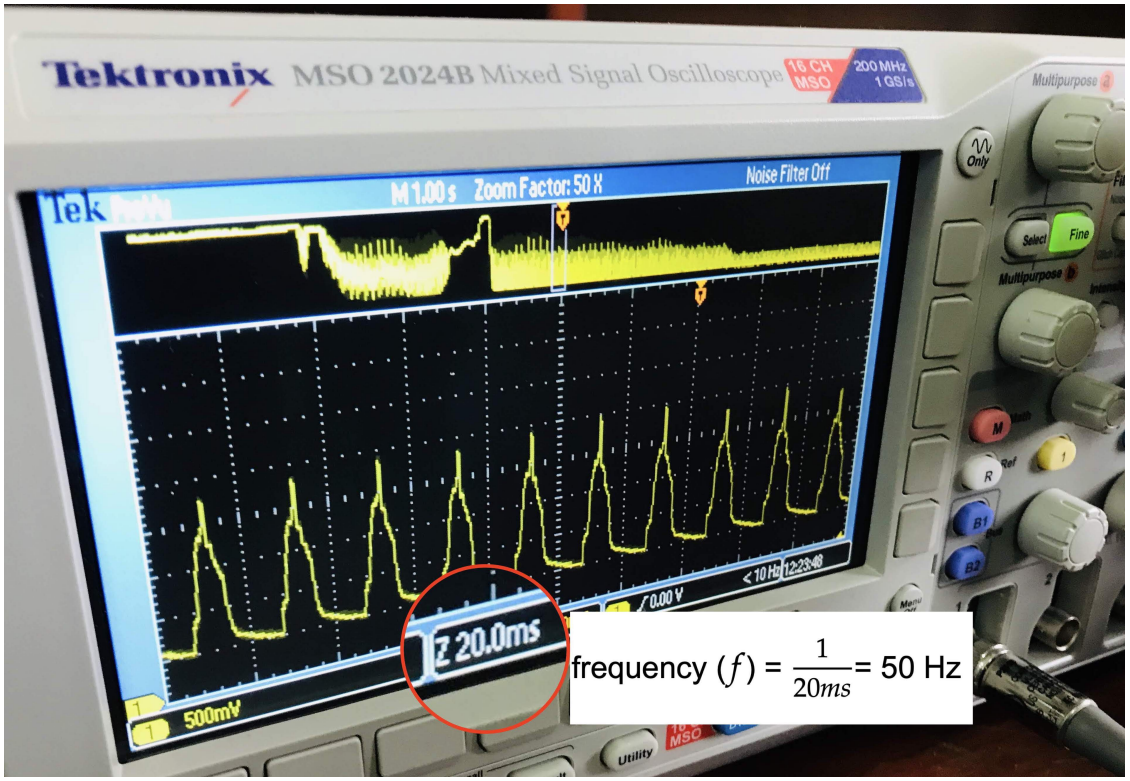
Initially our goal was to amplify the signal received from the photodiodes. Then we observed that we could not receive a clear amplified heartbeat signal as it gets suppressed by the 50Hz noise signal.

5.1.1 Without the filter - Dominant 50Hz noise



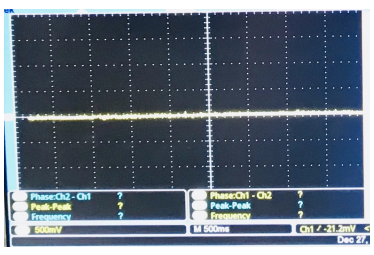
(a) oscilloscope output

(b) zoomed view

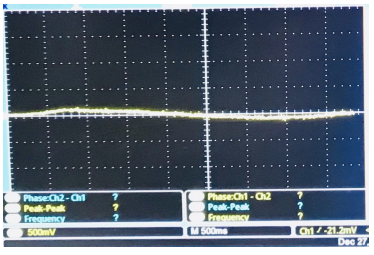


5.1.2 With the active bandpass filter

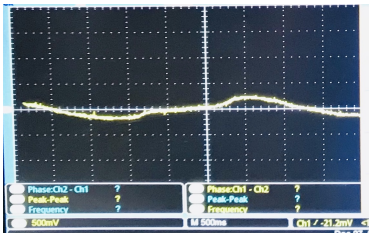
The initial circuit was modified into an active bandpass filter circuit by calculating the necessary values as shown in 4.2.1.Next, using the signal generator, the characteristics of our filter circuit from the digital oscilloscope was observed. The responses were as follows;
Observation: The amplitudes of the frequencies at the cut off frequencies were gradually reduced.[1]



(a) 0.1Hz

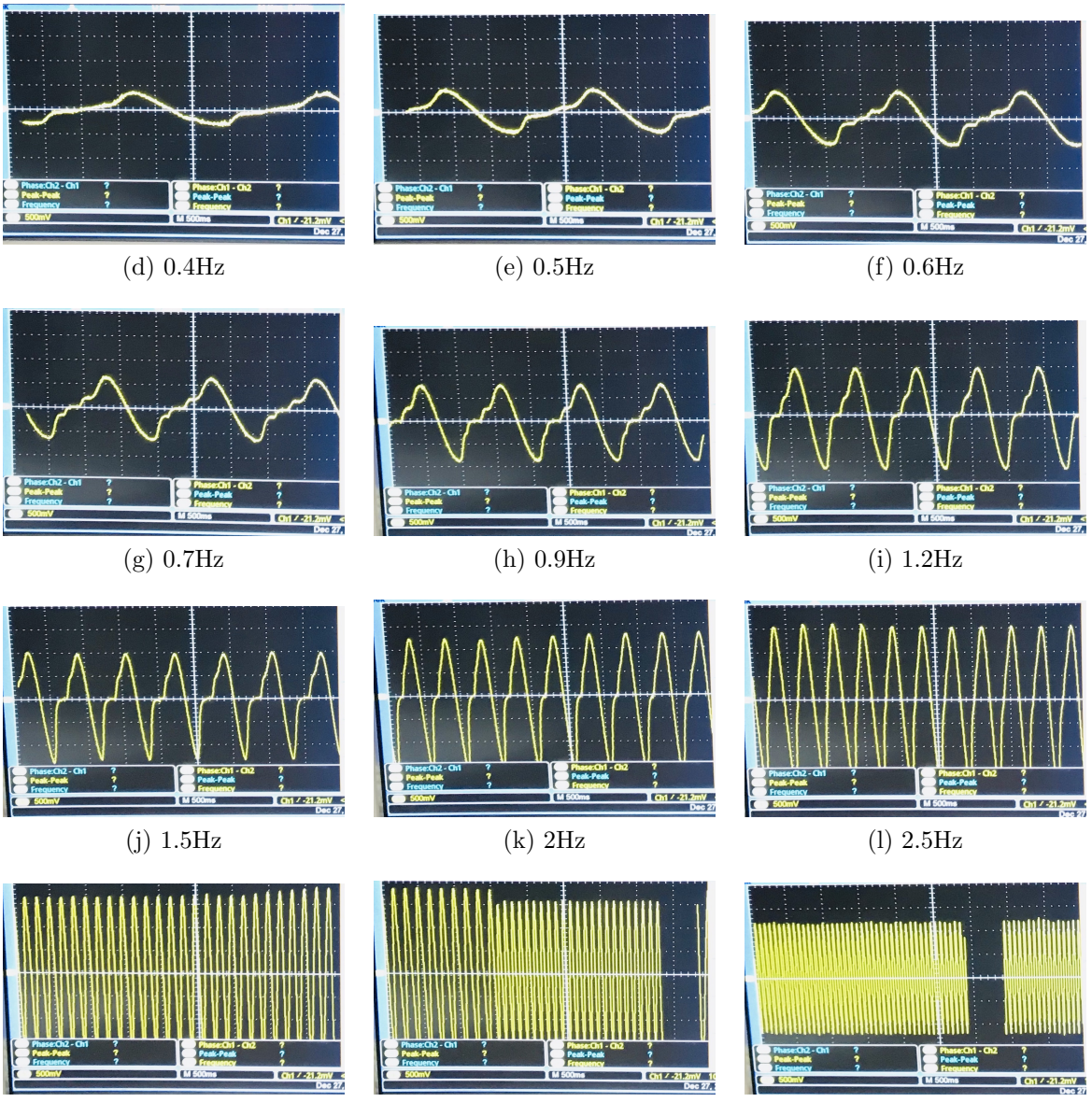


(b) 0.2Hz



(c) 0.3Hz

[1]The required range of frequencies 50-140bpm (0.83 - 2.33Hz) were passed without much attenuation



(d) 0.4Hz (e) 0.5Hz (f) 0.6Hz

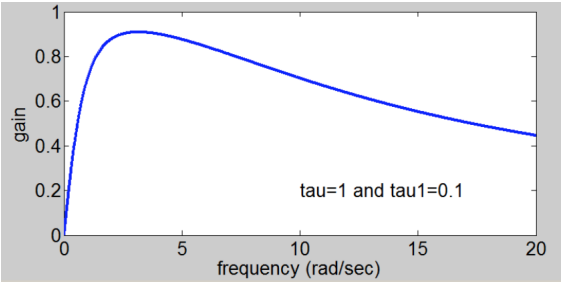
(g) 0.7Hz (h) 0.9Hz (i) 1.2Hz

(j) 1.5Hz (k) 2Hz (l) 2.5Hz

(m) 6Hz (n) 10Hz (o) 15Hz

Figure 5: Frequency response of the bandpass filter

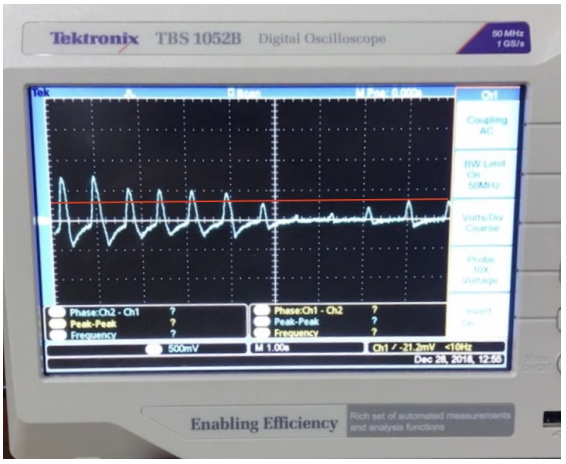
By the observations above we were able to verify that the response of this circuit can be given by the graph below:



5.1.3 Obtaining a threshold value

During testing, it was observed that due to ambient lighting conditions and movement of the patients' finger placement between the LED and photodiode sudden amplitude changes can occur.
It was evident that a hardcoded threshold value will not count the number of peaks accurately. Therefore, the calibration algorithm which was explained in 4.3.2 determines a threshold value per session.

(a) Varying amplitudes



(b) Amplitude Gradually decreasing

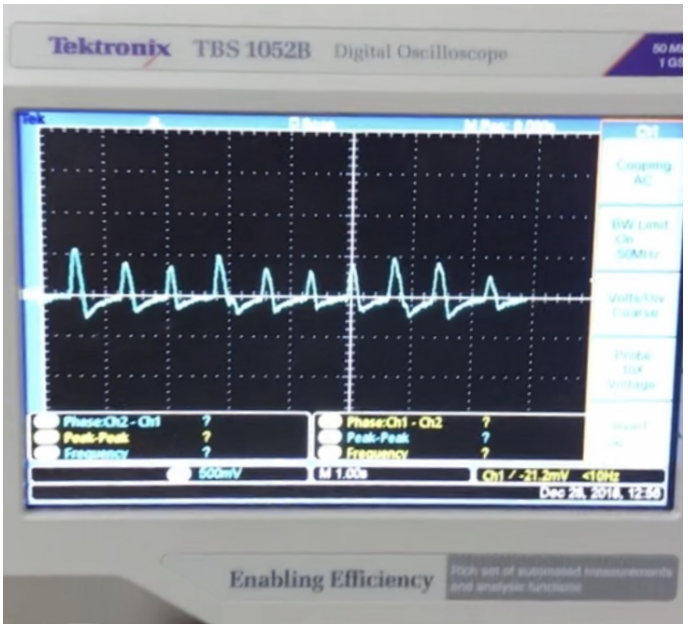


Figure 7: Filtered Waveform

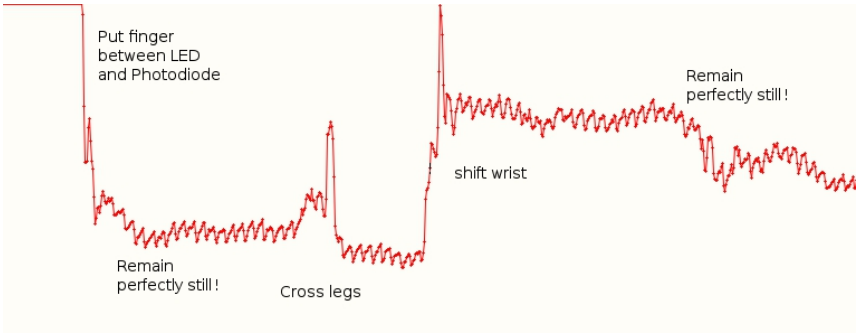


Figure 6: How PPG Varies[2]

6 Discussion

Using PPG we were able to monitor the heart rate with much accuracy and it seemed to be an easy method to be used for portable heart rate monitors. Since frequency of the analog waveform of the pulse that we receive of a person doesn't exceed 2Hz the sampling frequency of 5Hz that we have used in this project seem to give the most accurate results. Noise is the main issue that many faces when receiving analog signals and we were able to overcome that issue with the use of an appropriate active bandpass filter. The most difficult part of the project was to convert the analog signal a digital signal. Here we used a square regression line as the threshold and then counted the peaks. We used IR sensors to get the input signal from the fingertip, but it seems that using a green light PPG sensor will give a better input signal. Also, with the use of a smaller in size enclosure the heart rate monitor would be more convenient as a compatible device. As an improvement the use of a more advanced sensor would make this device output more accurate results.

7 Online Materials

The project has been open sourced on Github. All the diagrams, schematics, PCB designs can be found through our repository.

https://github.com/ramithuh/Pulse-Sensor.

References

[1] Analog readings with avr. https://maker.pro/custom/tutorial/how-to-take-analog-readings-with-an-avr-microcontroller. Accessed: 2018-12-28.

[2] Anatomy of the diy heart rate monitor. https://pulsesensor.com/blogs/news/6326816-anatomy-of-the-diy-heart-rate-monitor. Accessed: 2019-01-02.

[3] Atmega32 datasheet. http://ww1.microchip.com/downloads/en/devicedoc/doc2503.pdf. Accessed: 2018-11-02.

[4] Harvard heart letter. https://www.health.harvard.edu/heart-health/want-to-check-your-heart-rate-heres-how. Accessed: 2018-12-30.

[5] Ppg sensing illustration. https://global.epson.com/innovation/core_technology/wearable/vital_sensing.html. Accessed: 2018-12-30.