MICROPROCESSOR BASED BEAT RATE ANALYSER USING MATLAB

GRADUATION PROJECT SUBMITTED TO THE FACULTY OF ENGINEERING

OF

NEAR EAST UNIVERSITY

By

Murat GÜNDÜZ

Süleyman DİZEN

In Fulfillment of the Requirement for

The Degree of Bachelor of Science

In Biomedical Engineering

NICOSIA - 2014

ABSTRACT

Purpose of this project is reserching and applying about Beat Rate Calculation and Analysis using MATLAB by obtained microprocessor based signal from fingertip. The beat rate signal that obtained from finger is processed with the microprocessor then the processed clean signal is sent to computer into the Matlab. Finally, the signal is corolated and also processed using matlab to see result and stuation of measurant.

ACKNOWLEDGEMENT

I would like to thank our project advisor and lecturer of many courses, Ali IŞIN, and director of Biomedical Department Assosciate Prof. Dr. Terin ADALI that is why she make us to do better future and her knowledge to overcome hardness that will occure in the future.

Contents

Chapter-1

1. Structure of Heart

Introduction

The heart is one of the most important organs in the entire human body. It is really nothing more than a pump, composed of muscle which pumps blood throughout the body, beating approximately 72 times per minute of our lives. The heart pumps the blood, which carries all the vital materials which help our bodies function and removes the wast product that we do not need.

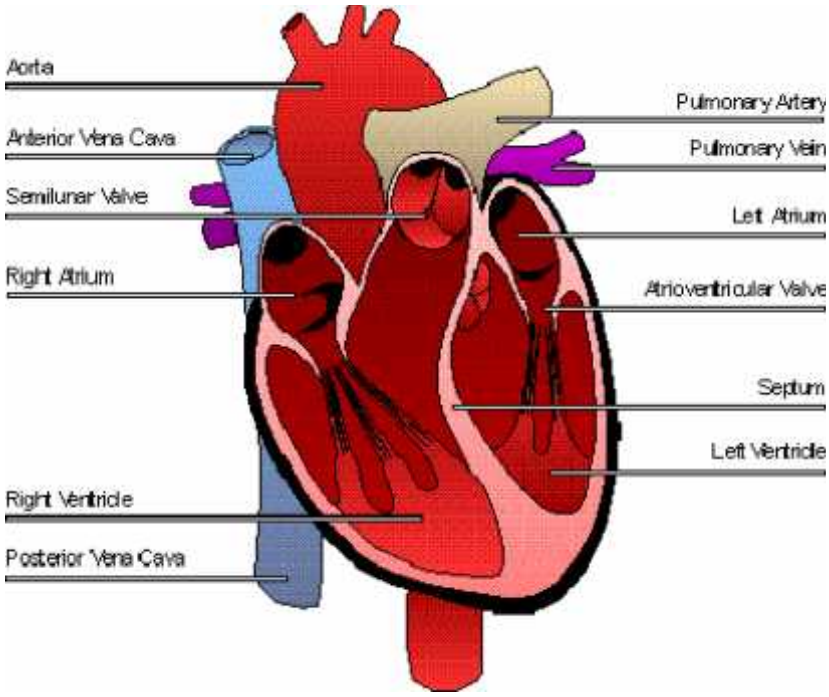

Figure - 1

The heart is essentially a muscle(a little larger than the first). Like any other muscle in the human body, it contracts and expands. Unlike skeletal muscles, however, the heart works on the "All -or-Nothing Law". That is, each time the heart contracts it does so with all its force. In skeletel muscles, the principle of "gradation" is present. The pumping of the heart is called the cardiac cycle, which occurs about 72 times per minute. This means that each cycle lasts about eight-tenths of a second. During this cycle the entire heart actually rests for about four-tenths of a second.

The walls of the heart are made up of three layers, while the cavity is divided into four parts. There are two upper chambers, called the right and left*atria*, and two lower chambers, called the right and left *ventricles*. The Right Atrium, as it is called, receives blood from the upper and lower body through the *superior vena cava* and the inferior vena cava, respectively, and from the heart muscle itself through the *coronary sinus*. The right atrium is the larger of the two atria, having very thin walls. The right atrium opens into the right ventricle through the *right atrioventicular valve*(tricuspit), which only allows the blood to flow from the atria into the ventricle, but not in the reverse direction. The right ventricle pumps the blood to the lungs to be reoxygenated. The left atrium receives blood from the lungs via the four pulmonory *veins*. It is smaller than the right atrium, but has thicker walls. The valve between the left atrium and the left ventricle, the *left atrioventicular valve*(bicuspid), is smaller than the tricuspid. It opens into the left ventricle and again is a one way valve. The left ventricle pumps the blood throughout the body. It is the *Aorta*, the largest artery in the body, which originates from the left ventricle.

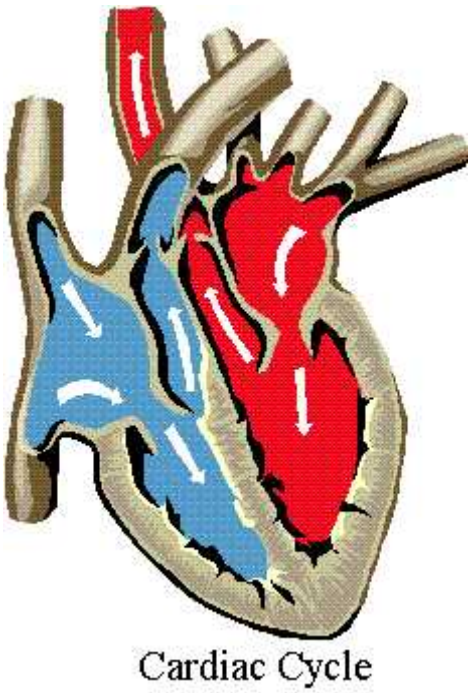

Figure - 2

Chapter -2

2. Heart Beat Rate

Heart rate is a term used to describe the frequency of the cardiac cycle. It is considered one of the four vital signs. Usually it is calculated as the number of contractions (**heart beats**) of the heart in one minute and expressed as "beats per minute" (bpm). See "Heart" for information on embryofetal heart rates. The heart beats up to 120 times per minute in childhood.

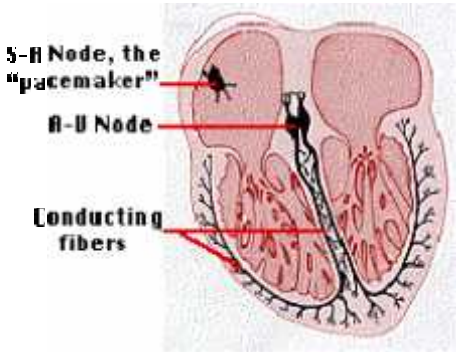

Figure - 3

When resting, the average adult human heart beats at about 70 bpm (males) and 75 bpm (females); however, this rate varies among people and can be significantly lower in athletes. The infant/neonatal rate of heartbeat is around 130-150 bpm, the toddler's about 100–130 bpm, the older child's about 90–110 bpm, and the adolescent's about 80–100 bpm.

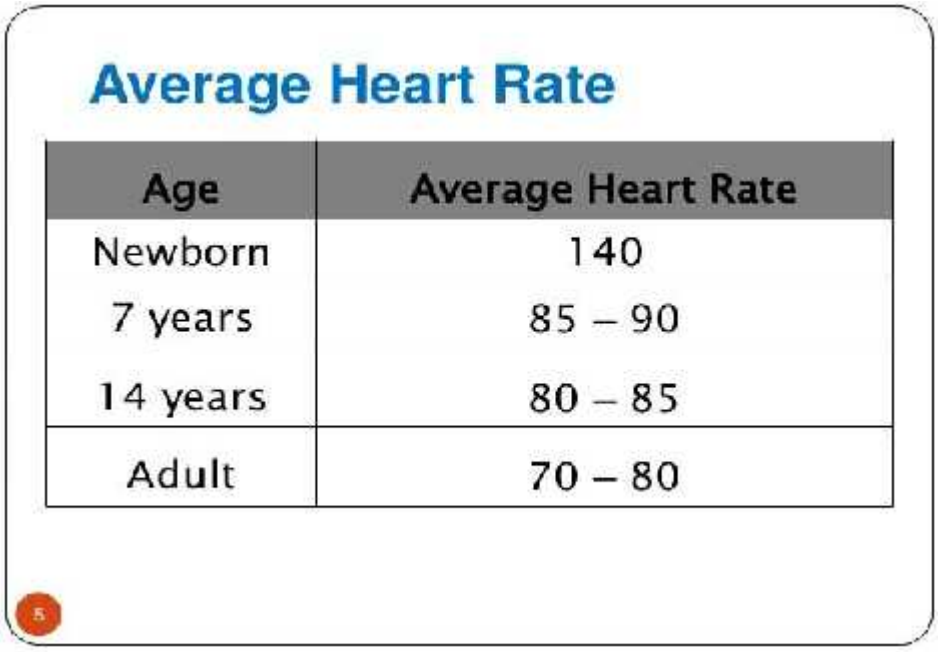

Figure - 4

The pulse is the most commonly used method of measuring the heart rate. This method may be inaccurate in cases of low cardiac output, as happens in some arrhythmias, where the heart rate may be considerably higher than the pulse rate.

Chapter - 3

3. Measurement of HBR

Introduction

Heart rate measurement indicates the soundness of the human cardiovascular system. This project demonstrates a technique to measure the heart rate by sensing the change in blood volume in a finger artery while the heart is pumping the blood. It consists of an infrared LED that transmits an IR signal through the fingertip of the subject, a part of which is reflected by the blood cells. The reflected signal is detected by a photo diode sensor. The changing blood volume with heartbeat results in a train of pulses at the output of the photo diode, the magnitude of which is too small to be detected directly by a microcontroller. Therefore, a two-stage high gain, active low pass filter is designed using two Operational Amplifiers (OpAmps) to filter and amplify the signal to appropriate voltage level so that the pulses can be counted by a matlab. The heart rate is displayed on a matlab result. The microcontroller used in this project is PIC16F877.

General block diagram is shown below;

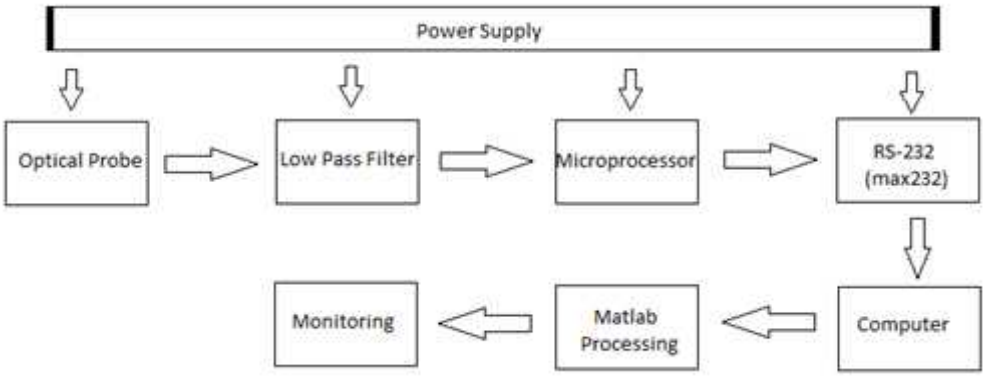

Figure - 5

Chapter - 4

4. Understanding of Circuit

4.1. Design of Optical Sensor

Optical sensor is working based on the light passes through into finger and it is received a photo transistor that placed otherside of the finger. This light source is infrared-led diode which emits the light as 860nm around. Heamoglobins are absorbed this wave. in the rest light is passed continiously. when the ventricule contraction occures, hemoglobins passes suddenly between the transmitter and receiver and prevents to passing. This is a signal changes for us. So, we can be obtained a signal every beat.

In our case, the sensor has been designed with different idea. we decided as transmitter and receiver are replaced 90° eachother. IR is placed tip of finger and Phototransistor is placed palm side. Figure is shown below. First idea is there is no bone between them. So, lack of light is decreased.

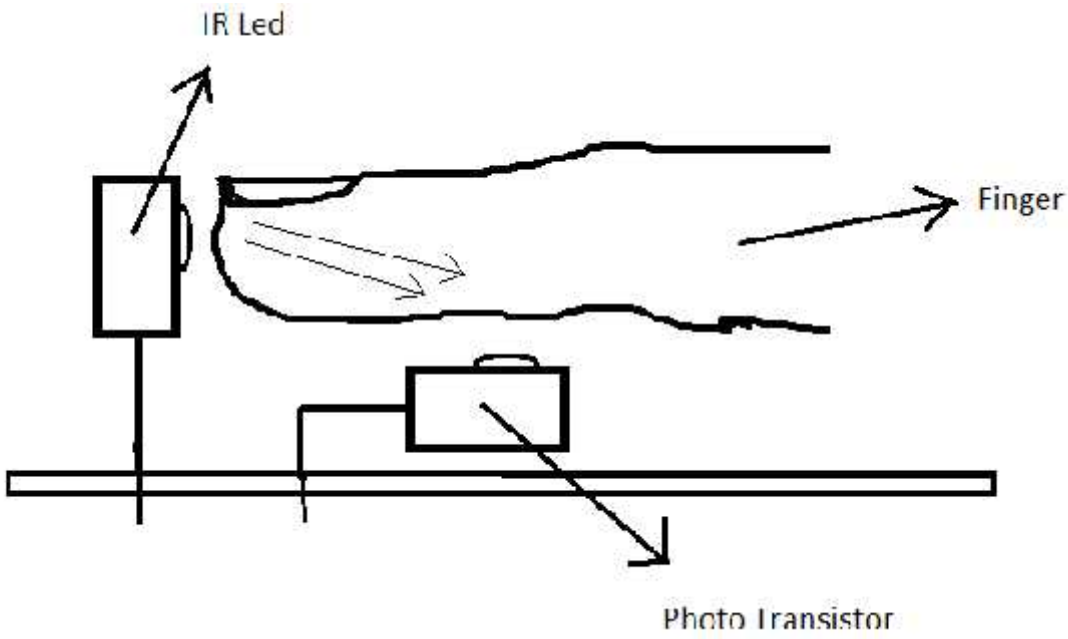

Figure - 6

4.2. Filter Circuit

Frequency of beatrates is avarage 70 beat/min. If there is brady cardia, it can be over 100 beat/min. so, the filter that we will designe can be answered 120 beat/min around. we need to filter as 3Hz around.

First we need to simple rc filter to decrease some analog unwanted signals to reject. 1uF condansator and 68k resistor perform a simple filter for the beggining of the first passing. Cut-off frequency is shown following formula;

F=1/2xPixRxC = 1/2x3.14x68x10^3x10^-6 = 2.34 Hz

Here we have 2 opamp circuit for filtering the signal again and amplifie the signal. There is a integrator circuit to filter the signal and gain. The cut-off frequency is same here and we have gain due to 6.8k and 680k. the formula is

Vout = Vin (1+ Rf/Rin) = Vin (1+ 680/6.8) = Vin x 101. this is our first gain.

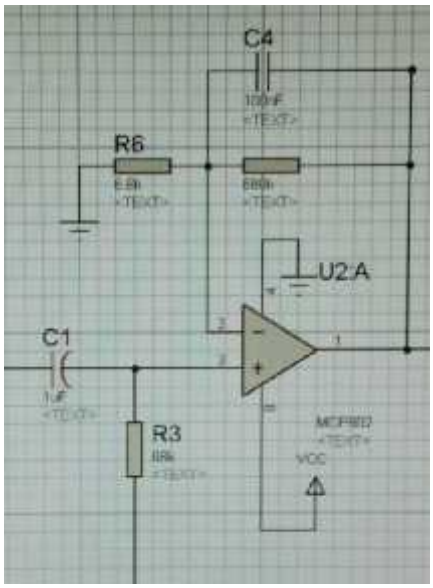

Figure - 7

The circuit in Figure - 7is repeated 2 times to increase the gain and lack of signals partition is increased.

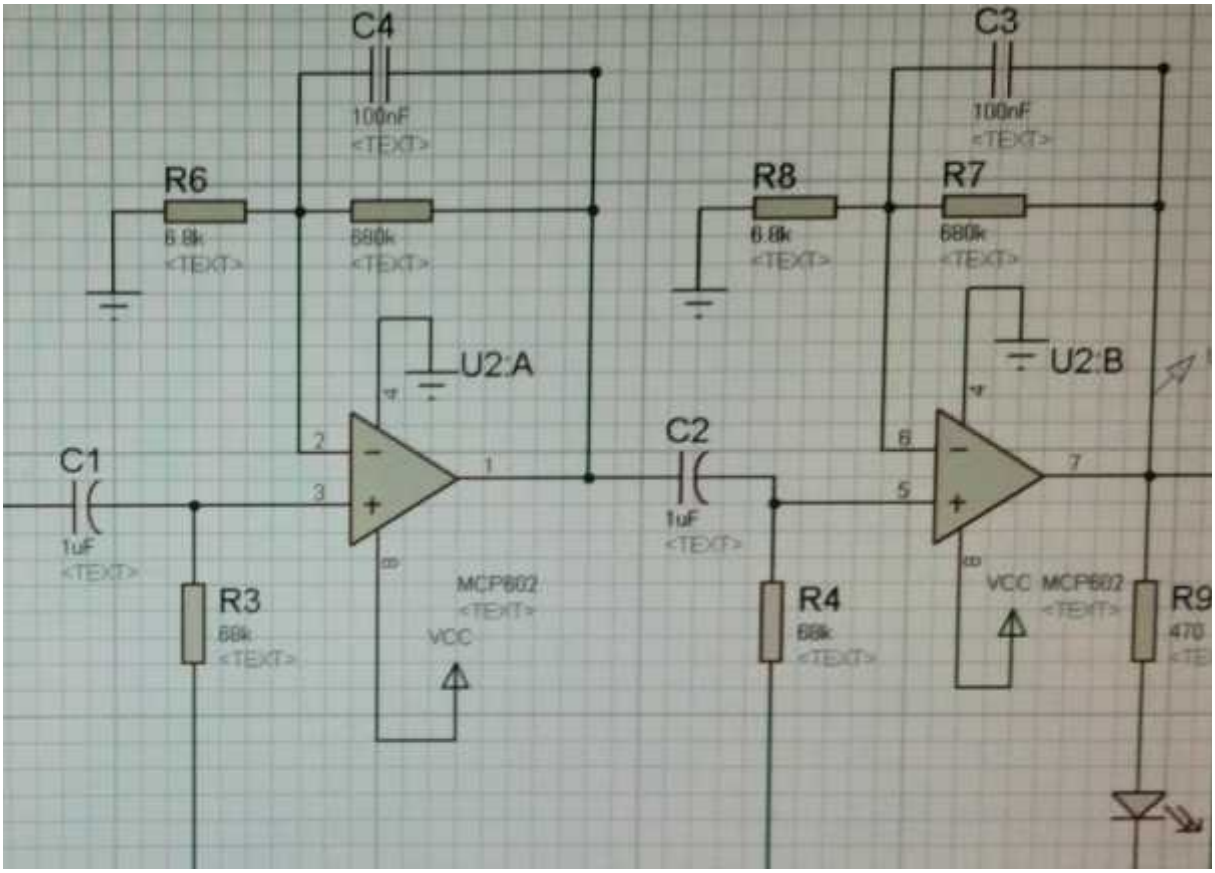

Figure - 8

End of the day the output of Low Pass Filter circuit has gain aproximately by 100k and cut-off frequency by 2.4 Hz. This circuit is enough to obtain beatrate signals.

There is a led placed output of the filter circuit which provides to see pulses.

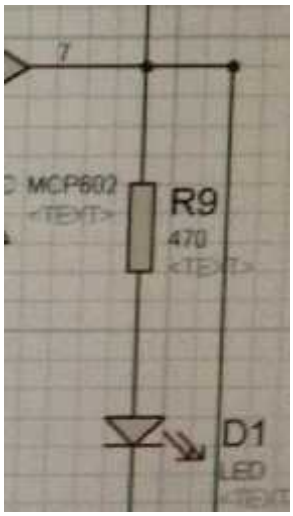

Figure - 9

4.3. Microprocessor Circuit

To upload the signal to matlab is needed to an Analog Digital Converter. It means that we should convert the signal into the computer language. the microprocessor takes the analog signal that we have already obtained and converts into digital signal and prepares the signal to send to Matlab.

4.3.1. Overview of PIC 16F877

PIC 16F877 is one of the most advanced microcontroller from Microchip. This controller is widely used for experimental and modern applications because of its low price, wide range of applications, high quality, and ease of availability. It is ideal for applications such as machine control applications, measurement devices, study purpose, and so on. The PIC 16F877 features all the components which modern microcontrollers normally have. The figure of a PIC16F877 chip is shown below.

Figure - 10

4.3.2. Features of PIC16F877

The PIC16FXX series has more advanced and developed features when compared to its previous series. The important features of PIC16F877 series is given below.

4.3.3 General Features

- High performance RISC CPU.
- ONLY 35 simple word instructions.
- All single cycle instructions except for program branches which are two cycles.
- Operating speed: clock input (200MHz), instruction cycle (200nS).
- Up to 368×8bit of RAM (data memory), 256×8 of EEPROM (data memory), 8k×14 of flash memory.
- Pin out compatible to PIC 16C74B, PIC 16C76, PIC 16C77.
- Eight level deep hardware stack.
- Interrupt capability (up to 14 sources).
- Different types of addressing modes (direct, Indirect, relative addressing modes).
- Power on Reset (POR).
- Power-Up Timer (PWRT) and oscillator start-up timer.
- Low power- high speed CMOS flash/EEPROM.
- Fully static design.
- Wide operating voltage range (2.0 – 5.56)volts.
- High sink/source current (25mA).
- Commercial, industrial and extended temperature ranges.
- Low power consumption (<0.6mA typical @3v-4MHz, 20µA typical @3v-32MHz and <1 A typical standby).

4.3.4. Peripheral Features

- Timer 0: 8 bit timer/counter with pre-scalar.
- Timer 1:16 bit timer/counter with pre-scalar.

- Timer 2: 8 bit timer/counter with 8 bit period registers with pre-scalar and post-scalar.
- Two Capture (16bit/12.5nS), Compare (16 bit/200nS), Pulse Width Modules (10bit).
- 10bit multi-channel A/D converter
- Synchronous Serial Port (SSP) with SPI (master code) and I2C (master/slave).
- Universal Synchronous Asynchronous Receiver Transmitter (USART) with 9 bit address detection.
- Parallel Slave Port (PSP) 8 bit wide with external RD, WR and CS controls (40/46pin).
- Brown Out circuitry for Brown-Out Reset (BOR).

4.3.5. Key Features

- Maximum operating frequency is 20MHz.
- Flash program memory (14 bit words), 8KB.
- Data memory (bytes) is 368.
- EEPROM data memory (bytes) is 256.
- 5 input/output ports.
- 3 timers.
- 2 CCP modules.
- 2 serial communication ports (MSSP, USART).
- PSP parallel communication port
- 10bit A/D module (8 channels)

4.3.6. Analog Features

- 10bit, up to 8 channel A/D converter.
- Brown Out Reset function.

- Analog comparator module.

4.3.7. Special Features

- 100000 times erase/write cycle enhanced memory.
- 1000000 times erase/write cycle data EEPROM memory.
- Self programmable under software control.
- In-circuit serial programming and in-circuit debugging capability.
- Single 5V,DC supply for circuit serial programming
- WDT with its own RC oscillator for reliable operation.
- Programmable code protection.
- Power saving sleep modes.
- Selectable oscillator options.

4.3.8. Pin Diagrams

PIC16F877 chip is available in different types of packages. According to the type of applications and usage, these packages are differentiated. The pin diagrams of a PIC16F877 chip in different packages is shown in the figure below.

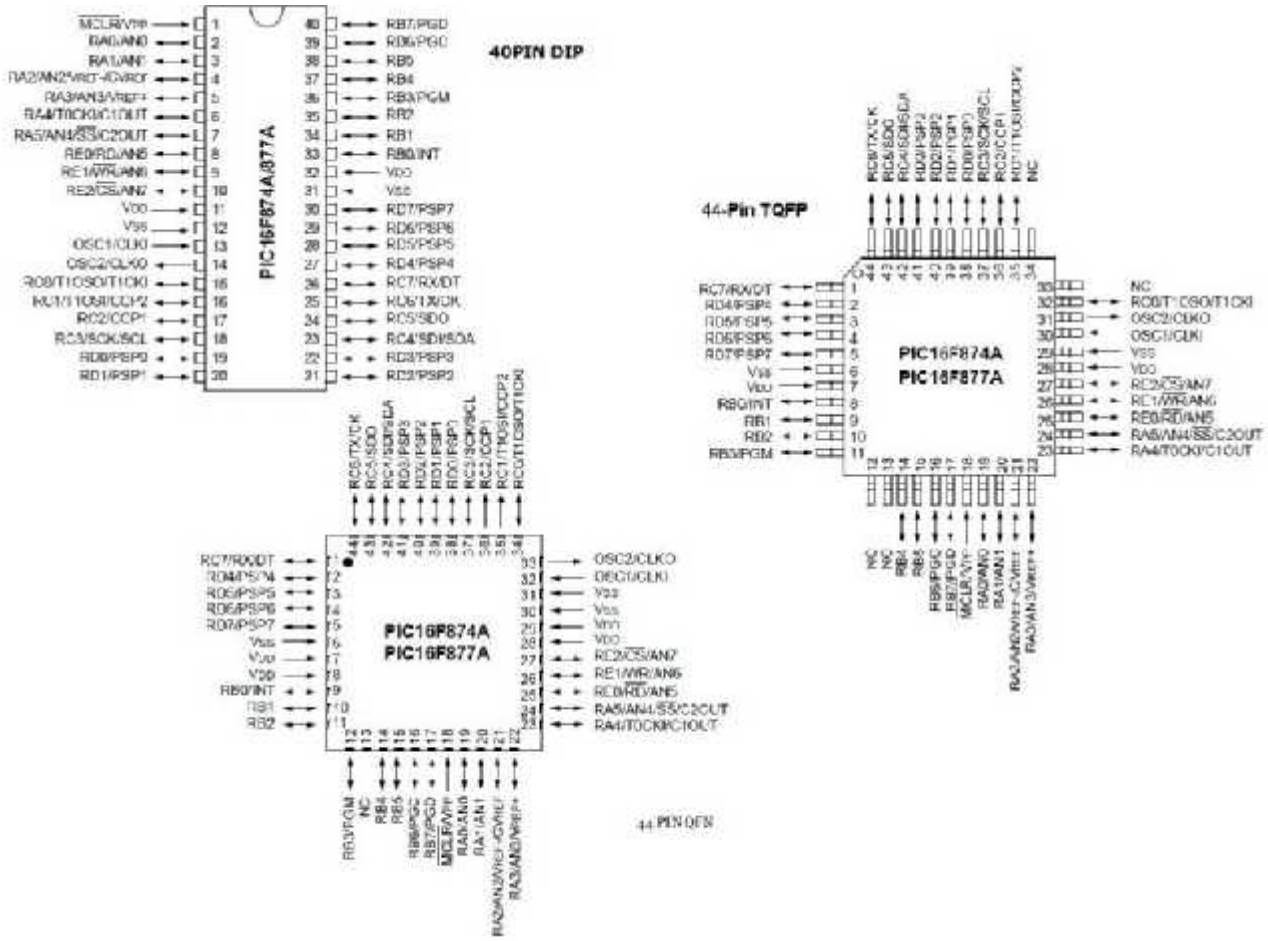

Figure - 11

4.3.9. Input/output ports

PIC16F877 has 5 basic input/output ports. They are usually denoted by PORT A (R A), PORT B (RB), PORT C (RC), PORT D (RD), and PORT E (RE). These ports are used for input/ output interfacing. In this controller, “PORT A” is only 6 bits wide (RA-0 to RA-7), ”PORT B” , “PORT C”,”PORT D” are only 8 bits wide (RB-0 to RB-7,RC-0 to RC-7,RD-0 to RD-7), ”PORT E” has only 3 bit wide (RE-0 to RE-7).

PORT-A	RA-0 to RA-5	6 bit wide
PORT-B	RB-0 to RB-7	8 bit wide
PORT-C	RC-0 to RC-7	8 bit wide
PORT-D	RD-0 to RD-7	8 bit wide
PORT-E	RE-0 to RE-2	3 bit wid

Figure -11

All these ports are bi-directional. The direction of the port is controlled by using TRIS(X) registers (TRIS A used to set the direction of PORT-A, TRIS B used to set the direction for PORT-B, etc.). Setting a TRIS(X) bit ‘1’ will set the corresponding PORT(X) bit as input. Clearing a TRIS(X) bit ‘0’ will set the corresponding PORT(X) bit as output.
(If we want to set PORT A as an input, just set TRIS(A) bit to logical ‘1’ and want to set PORT B as an output, just set the PORT B bits to logical ‘0’.)

- Analog input port (AN0 TO AN7) : these ports are used for interfacing analog inputs.
- TX and RX: These are the USART transmission and reception ports.
- SCK: these pins are used for giving synchronous serial clock input.
- SCL: these pins act as an output for both SPI and I2C modes.
- DT: these are synchronous data terminals.
- CK: synchronous clock input.
- SD0: SPI data output (SPI Mode).
- SD1: SPI Data input (SPI mode).
- SDA: data input/output in I2C Mode.
- CCP1 and CCP2: these are capture/compare/PWM modules.
- OSC1: oscillator input/external clock.
- OSC2: oscillator output/clock out.

- MCLR: master clear pin (Active low reset).
- Vpp: programming voltage input.
- THV: High voltage test mode controlling.
- Vref (+/-): reference voltage.
- SS: Slave select for the synchronous serial port.
- T0CK1: clock input to TIMER 0.
- T1OSO: Timer 1 oscillator output.
- T1OS1: Timer 1 oscillator input.
- T1CK1: clock input to Timer 1.
- PGD: Serial programming data.
- PGC: serial programming clock.
- PGM: Low Voltage Programming input.
- INT: external interrupt.
- RD: Read control for parallel slave port.
- CS: Select control for parallel slave.
- PSP0 to PSP7: Parallel slave port.
- VDD: positive supply for logic and input pins.
- VSS: Ground reference for logic and input/output pins.

4.4. Microprocessor Program in C

This mcu has a 13-bit ADC and it supported maximum 20Mhz Crystal. We use this mcu as a analog probe. So, the alghortihm that we need is below;

```
int value;

void main()

{

//setting for mcu setup

  setup_adc_ports(AN0);

  setup_adc(ADC_CLOCK_DIV_2);
```

```
  setup_psp(PSP_DISABLED);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);

  //TODO: User Code
  set_adc_channel(0);          //setting and chose the analog input A0.
  while(true){                 // infinit loop.
  value = read_adc();          // reading adc
    printf("%u\r\n",value);    // sending to max232
  }
}
```

The analog input A0 is used to obtain analog signal. The analog signal is converted into digital signal as 8-bit. then it is sent to max232 to convert into Pc.

4.5. Max-232 Circuit

The **MAX232** is an IC, first created in 1987 by Maxim Integrated Products, that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.

The drivers provide RS-232 voltage level outputs (approx. ± 7.5 V) from a single + 5 V supply via on-chip charge pumps and external capacitors. This makes it useful for implementing RS-232 in devices that otherwise do not need any voltages outside the 0 V to + 5 V range, as power supply design does not need to be made more complicated just for driving the RS-232 in this case.

The receivers reduce RS-232 inputs (which may be as high as ± 25 V), to standard 5 V TTL levels. These receivers have a typical threshold of 1.3 V, and a typical hysteresis of 0.5 V.

RS232 line type and logic level	RS232 voltage	TTL voltage to/from MAX232
Data transmission (Rx/Tx) logic 0	+3 V to +15 V	0 V
Data transmission (Rx/Tx) logic 1	-3 V to -15 V	5 V
Control signals (RTS/CTS/DTR/DSR) logic 0	-3 V to -15 V	5 V
Control signals (RTS/CTS/DTR/DSR) logic 1	+3 V to +15 V	0 V

Figure -12

The MAX232(A) has two receivers (converts from RS-232 to TTL voltage levels), and two drivers (converts from TTL logic to RS-232 voltage levels). This means only two of the RS-232 signals can be converted in each direction. Typically, a pair of a driver/receiver of the MAX232 is used for TX and RX signals, and the second one for CTS and RTS signals.

There are not enough drivers/receivers in the MAX232 to also connect the DTR, DSR, and DCD signals. Usually these signals can be omitted when e.g. communicating with a PC's serial interface. If the DTE really requires these signals, either a second MAX232 is needed, or some other IC from the MAX232 family can be used. Also, it is possible to directly wire DTR (DB9 pin #4) to DSR (DB9 pin #6) without going through any circuitry. This gives automatic (brain dead) DSR acknowledgment of an incoming DTR signal.

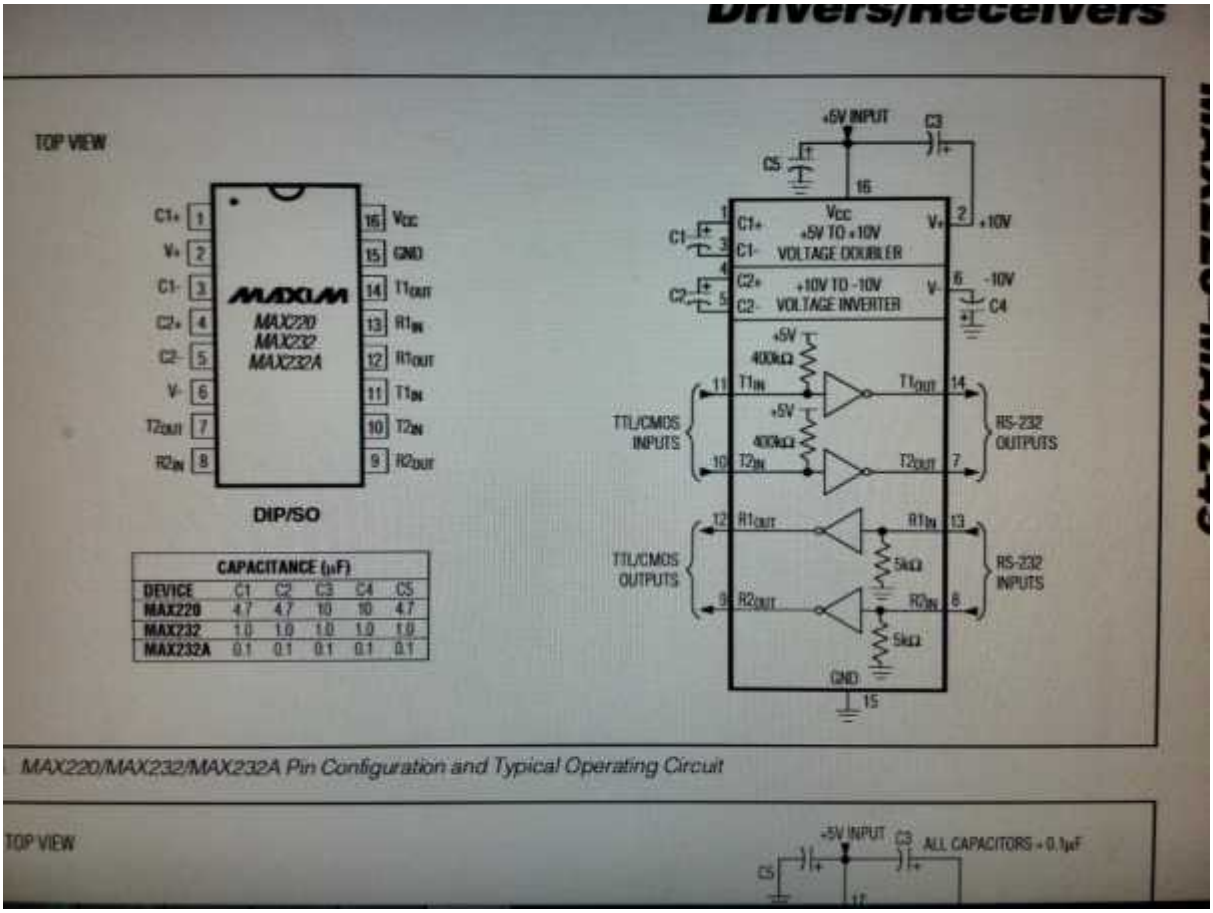

Figure - 13

4.6. MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include;

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to *learn* and *apply* specialized technology.Toolboxesare comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are

availableincludesignal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

4.6.1. The MATLAB System

The MATLAB system consists of five main parts;

a)The MATLAB language.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

b) The MATLAB working environment.

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

c)Handle Graphics.

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

d)The MATLAB mathematical function library.

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

e)The MATLAB Application Program Interface (API).

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and forreading and writing MAT-files.

4.7. Taking the signal into the MATLAB

4.7.1. Obtaining Pure Signal

The pure signal is obtained from microprocessor into the Matlab.

```
% Receiving signals to matlab
 clear all;
 Port = input('Enter your avalible COM port...','s');
 s=serial('Port');
  fopen(s);

i=1;
disp('Plotting... Please wait...');
tic;
while ( toc <= 16 )
    a(i)=toc;
    d(i)= str2double(fscanf(s));
    i=i+1;

end
    fin=toc;
    g = a.*0.019608;
    plot(g,d);
```

```
    xlabel('Second');
    ylabel('Voltage');
    axis([0 16 0 5]);
    grid;

fclose(s);
r =[g.',d.'];

% Definition variables and constants
fs=200; % sample rate


puresignal = r(:,2);
plot(puresignal);
title('Pure Signal');
figure
```

Example figure is shown below;

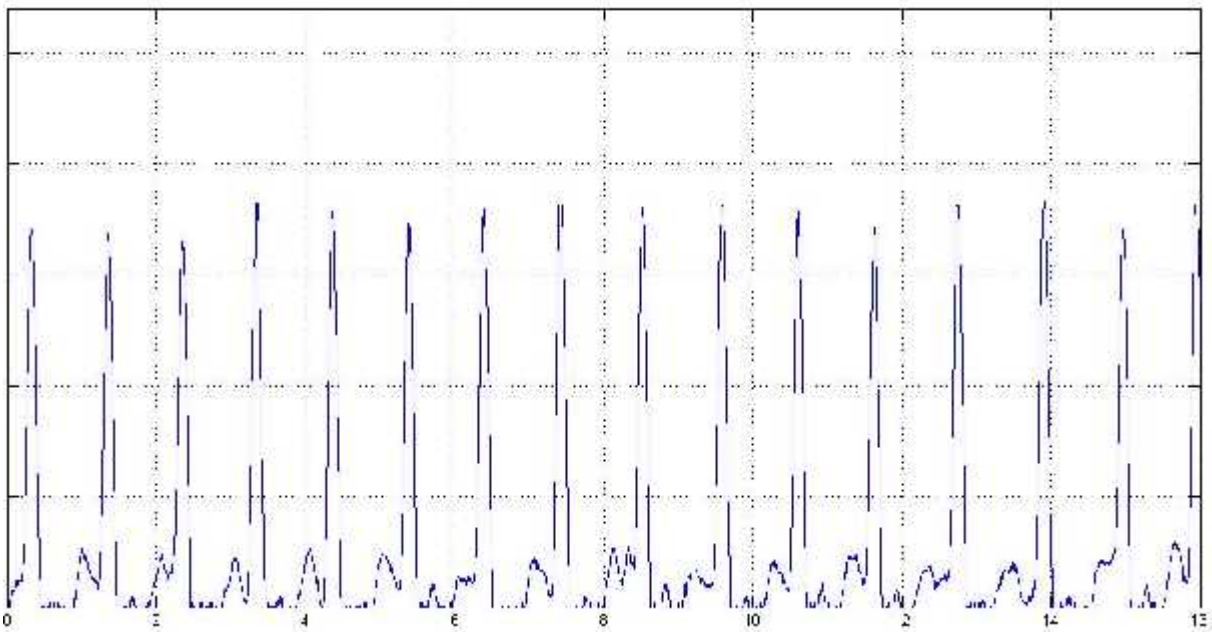

Figure - 14

4.7.2. Moving avarage filter

Moving averages are ubiquitous to technical analysis. I would therefore be remiss if I didn't comment on some techniques to improve their application. Two general characteristics of all moving averages are that they smooth the input data and they lag that data. Their use and application is almost always a tradeoff between these characteristics. The smoothing function means that the higher frequency components of the input data are removed, so moving averages are also referred to as low pass filters. That is, they pass only the low frequency components while removing the high frequency components.

Moving average lag is probably the most important characteristic for traders to understand quantitatively.

Here we have 10 point moving avarage filtert obtain frequency response our calculation. It is the first step the first filtration process.

```
% 10 point avarage filter
B=(1/10)*ones(1,10);
A=1;
freqz(B,A);
title('10 Point Moving Avarage Filtre');
figure
avragefilteredsignal=filter(B,A,puresignal);
plot(avragefilteredsignal)
title('Moving Avarage (Low Pass) passed signal');
figure
```

4.7.3. Derivative Based Filter

Derivative filters provide a quantitative measurement for the rate of change in pixel brightness information present in a digital image. When a derivative filter is applied to a digital image, the resulting information about brightness change rates can be used to enhance contrast, detect edges and boundaries, and to measure feature orientation.

Here we have code for derivative filter. In here we need to compress low level values of obtained signal. This technique provides us to see the beats rises after filtration.

```
% Derivative Based Filter
B=(1/1.0025)*[1 -1];% 1.0025 normalizationon value
A=[1 -0.995];
freqz(B,A);
title('Derivative Based Filter');
figure
derivativefilteredsignal=filter(B,A,avragefilteredsignal);
plot(derivativefilteredsignal)
title('Derivative Based (High Pass) passed signal');
figure
```

4.7.4. 50hz notch and rejection process

In this effort, we will assume that the dynamics of light incident on a spherical bead satisfies the model of a notch filter with a given notch frequency ω_n. The input into the system consists of light of a given wavelength (correspondingly an excitation frequency ω) and a known magnitude denoted as *A*. The output from the system consists of a light waveform with a given attenuated amplitude denoted as *B*.

4.7.5. System identification of a Notch Filter

The following methodology will utilize a measurement of *B* given a known input waveform consisting of *A* and ω, to estimate the notch frequency ω_n.To compute the notch frequency we employ the model of a second order notch filter with a given transfer function given by

$$\frac{Y(s)}{R(s)} = \frac{s^2 + \mu_n^2}{s^2 + 2\mu_n s + \mu_n^2}. \qquad (1)$$

Figure - 15 shows the resulting magnitude bode plot of (1).

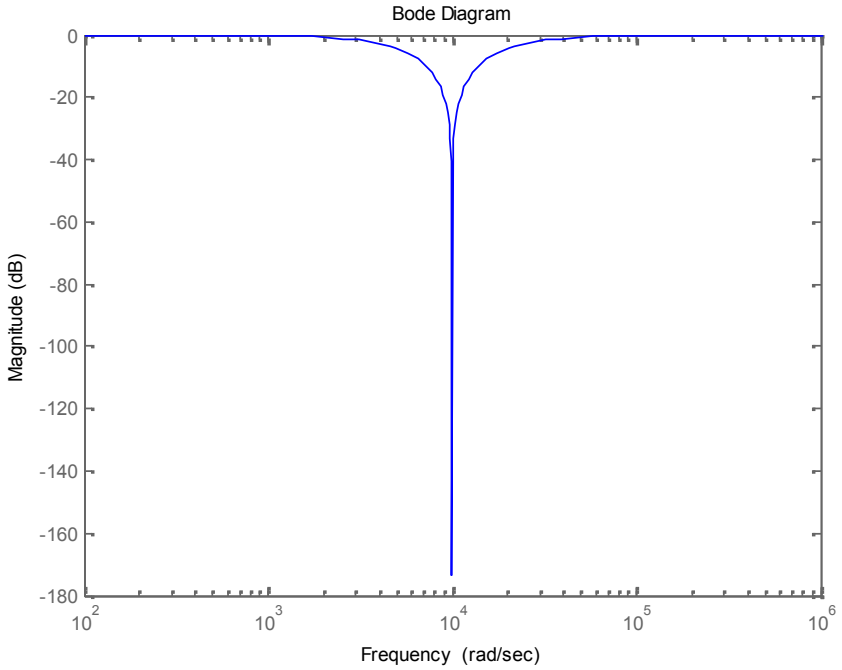

Figure - 15

The input signal *R*(*s*) can be modeled as a pure cosine function described by $r(t) = A\cos\mu t$. Using this excitation function into (1) produces a steady state solution for *y*(*t*) as follows

$$y(t) = B_1 \cos(\mu t) + B_2 \sin(\mu t), \qquad (2)$$

where B_1 and B_2 are defined as

$$B_1 = \frac{A\left|\mu_n^2 - \mu^2\right|}{(\mu_n^2 - \mu^2)^2 + 4(\mu\mu_n)^2}(\mu_n^2 - \mu^2), \quad B_2 = -\frac{A\left|\mu_n^2 - \mu^2\right|}{(\mu_n^2 - \mu^2)^2 + 4(\mu\mu_n)^2}(2\mu\mu_n) \qquad (3)$$

.

Using (2) and (3), we can compute the magnitude of the output signal *B* to be

$B = \dfrac{A\left|\mu_n^2 - \mu^2\right|}{\sqrt{(\mu_n^2 - \mu^2)^2 + 4(\mu\mu_n)^2}}$, which can be further simplified as follows

$$B = \frac{A\left|\mu_n^2 - \mu^2\right|}{\mu_n^2 + \mu^2}, \tag{4}$$

From (4), if we choose an excitation frequency that is smaller than the notch frequency, then we can ignore the absolute sign and solve for μ_n to yield an explicit solution for μ_n as follows

(5a)

$$\mu_n = \sqrt{\frac{A+B}{A-B}}\mu \ .$$

If we choose an excitation frequency that is larger than the notch frequency, then we can replace the absolute sign with a negative sign and solve for μ_n to yield an explicit solution for μ_n as follows

$$\mu_n = \sqrt{\frac{A-B}{A+B}}\mu \ . \tag{5b}$$

Now since we do not explicitly know what the notch frequency is, thus we can not determine whether the excitation frequency is larger or smaller than μ_n. However, we can compute two estimates of the notch frequency for a given data set of *A*, *B*,ω. So, if we compute another two estimates of the notch frequency for another data set that is relatively close to the first data set, then we can compare which of the two estimates does not vary. Thus, if we know the input and output signal amplitudes and the input signal frequency, we can determine the system notch frequency.

Basicly the noise come from electricity 50Hz or 60Hz must be rejected. Our code is shown below;

```
% 50hz notch and rejection process
B=conv([1 1],[0.6310 -0.2149 0.1512 -0.1288 0.1227 -0.1288
0.1512 -0.2149 0.6310]);
A=1;
freqz(B,A);
title('Comb Filter');
figure
comb=filter(B,A,derivativefilteredsignal);
plot(comb)
title('Comb (rejection 50Hz and Harmonics) passed signal');
```

4.7.6. Differentiator and Square Operation

Here we use differentiator like a high pass filter. then we can apply square operation to increase the differens between high values and low values. for example, we have two values that are 2 and 5. difference is 3. but after the square operation, 4 and 25. now the difference is 19. thus we can observe now better.

The code is below;

```
% Differentiator
outputofdifferentiator=diff(comb);

% Squaring Operation
squareoutput=outputofdifferentiator.*outputofdifferentiator;
```

4.7.7. Moving window integrator

The slope of the R wave alone is not a guaranteed way to detect a QRS event.Many abnormal QRS complexes that have large amplitudes and long durations (notvery steep slopes) might not be detected using information about slope of the Rwave only. Thus, we need to extract more information from the signal to detect aQRS event.Moving window integration extracts features in addition to the slope of the Rwave.

This code prepared to develop the system in the future. In this case the code provides to obtain in window each rises peaks.

Code is shown below;

```
% Moving Window Integrator
window=ones(1,30); % N=30 moving window
integral= medfilt1(filter(window,1,squareoutput),10);
delay = ceil(length(window)/2);
integral = integral(delay:length(integral));
```

4.7.8. QRS detection and calculating beat rates

This procedure provides to obtain how it is working qrs cycle and the converting into beat rate calcualtion. the code can be marked peak points of the qrs or beat rate graphics and calculate into beat rate and puls.

The code is below finally;

```
% QRS Detection
max_h = max(integral);
thresh = 0.15;
poss_reg = integral > (thresh*max_h);
```

```
left = find(diff([0 poss_reg’])==1);
right = find(diff([poss_reg’ 0])==-1);

for i=1:length(left)
    [maxvalue(i) maxloc(i)] = max( comb(left(i):right(i)) );
    maxloc(i) = maxloc(i)-1+left(i); % offset ekle
    [mindeger(i) minloc(i)] = min( comb(left(i):right(i)) );
% minvalue gives us point Q
    minloc(i) = minloc(i)-1+left(i); % addition offset
% Offset for Q
end


maxposition=ones(1,4000)*(-max(comb));
maxposition(1,maxloc)=max(comb);


figure

plot(comb)
title(’Determined signal point of R’);
hold on

plot(maxloc,maxvalue,’g-’)

figure
plot(comb)
title(’Marked signal point of R’);
hold on

plot(maxposition,’r’)
legend(’Signal that pass from Comb filter’,’R’);
```

```
% Counting of beats


beat = length(maxloc)

if beat < 60
    disp('RESULT: BreadyCardia');
elseif beat > 100
    disp('RESULT: TachyCardia');
else
    disp('RESULT: NormalCardia');
end
```

Thus, the beatrate is calculated and monitored.

```
q=input('Do you want to send e-mail to doctor? y /
n...','s');

if strcmp(q,'y') == true
    mailgo;
end
```

Here we called "mailgo" sub-algorhythm and asked to send an e-mail all datas to chosen doctor.

```
function mailgo
o='terin adali';
name = input('Which doctor would you like to
send?','s');
disp('Sending...');
w = strcmp(name,o);
if w==true
```

```
    adress = ('xxxxx@gmail.com');
else
    adress = name;
end

myaddress = 'xxxxxxxxxx@gmail.com';
mypassword = 'xxxxxxxxx';

setpref('Internet','E_mail',myaddress);
setpref('Internet','SMTP_Server','smtp.gmail.com');
setpref('Internet','SMTP_Username',myaddress);
setpref('Internet','SMTP_Password',mypassword);

props = java.lang.System.getProperties;
props.setProperty('mail.smtp.auth','true');
props.setProperty('mail.smtp.socketFactory.class', ...
'javax.net.ssl.SSLSocketFactory');
props.setProperty('mail.smtp.socketFactory.port','465');

sendmail(adress, 'Patient: Murat GÜNDÜZ','Heart Beat
Rate Graphic','beatrate.jpg');
disp('Sucsessful');
```

This alghorithm uses gmail server to create an e-mail and send it to chosen e-mail adress.

Chapter - 5

5. Simple User Prosedure for how to work and how to use

Step -1

Plug the Power supply to the device.

Step - 2

Plug the Power supply to electricity.

Step - 3

Open the Matlab main screen. Be sure that the analyser program inside the wokspace folder.

Path must be like this (root\Users\Somebody\Documents\Matlab\HBRAnalyser.m)

Step -4

Plug the rs232 port to computer.

Step - 5

Put the finger on the Optical sensor. Be sure the puls indicator Led (Red) sens the pulses.

Step - 6

Type the HBRAnalyser on the Matlab and press enter. Then type the port name (format as 'comX') then press enter again. Wait for calculation around 15 second. After that, It shows your beat rate and graphics about your beats.

Conclusion

The purpose of this project was to determine the effects of exercise on heart rate. We did this by doing a series of activities that affected heart rate, including sitting, standing, walking, jogging, and running.The data showed that the heart rate increased with increasing exercise, going from 66 bpm for walking up to 106 bpm for running, so the data did support the hypothesis. An explanation of the data is that as exercise increases, more oxygen is needed by the muscles for cellular respiration to produce more energy, so the heart must beat faster to supply the O2 to the muscles. Some errors may have occurred during the trying. Since heart rate was taken by counting for 15 seconds, and multiplying by 4, there is an error of +/- 4 bpm built in to the procedure. Miss-counting could also have been a factor, causing the heart rate to appear higher or lower by a few beats, but not affecting the overall results of the generally. This project could have been improved by using heart rate monitors to get an exact count of beats, and by doing several trials of each activity to find an average heart rate for each activity.

Component List

* 1 x Pic16F877 MicroChip Microprocessor

* 1 x LM324 Low Power Operational Amplifier

* 1 x Max-232 Serial Interface Converter from TTL to CMOS

* 2 x 1uF Polarised Capacitor

* 2 x 100nF Capasitor

* 2 x 22pF Capasitor

* 2 x 68k Ohm Film Resistor

* 2 x 6.8k Ohm Film Resistor

* 2 x 680k Ohm Film Resistor

* 1 x 150 Ohm Film Resistor

* 1 x220 Ohm Film Resistor

* 1 x 470 Ohm Film Resistor

* 1 x 1k Ohm Film Resistor

* 1 x 10k Ohm Film Resistor

* 1 x 4Mhz Crystal Oscilator

* 1 x Infra Red Led Diode

* 1 x Photo Transistor

* 1 x Red Led Diode

* 15x6 Partinex

* 1 x 220 Ohm Linear Potentiometer

* 1 x 5V Regulated Power Supply

Referances

- http://www.worldinvisible.com/apologet/humbody/heart.htmns(US)!
- http://www.eetimes.com/document.asp?doc_id=1278571
- http://www.mathworks.com/matlabcentral/fileexchange/31780-ecg-qrs-detection-heart-beat-in-one-minute/content/heart_rate_qrs_detect.m
- http://www.physionet.org/physiotools/matlab/ECGwaveGen/
- http://www.edaboard.com/thread299158.html
- http://www.matpic.com/
- BME-312 Lecture Notes By Ali IŞIN