BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY(BUET)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CSE 316: Microprocessors, Microcontrollers, and Embedded Systems Sessional

Group Project Report

Portable Heart Rate Monitoring System

Authors:	*ID:*
Shovito Barua Soumma	1605066
Fahim Shahriar Anik	1605065
Md. Nazmul Islam	1605088

September 26, 2019

Contents

1 Abstract

The proposed project measures the heart rate of a person using optical sensors and send the measured data directly to an android device. The optical sensor detects the variation of blood volume at the fingertip. In this project's scope, the pulse sensor will be an infrared light emitting diode (IR LED) which will be on the same side of the finger(Reflective type 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 use a black tape to cover the pulse sensor for accurate measurement

2 Acknowledgement

We would like to express our gratitude to our supervisor Mr. Tareq Mahmood for giving us technical advice and guidance.
Our sincere thanks goes to Mr.Tareq Mahmood 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.

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 and it can send the measured the data to an android device. For an average person this value lies between 60-100 bpm. The heart rate rises gradually during exercises and returns slowly to the rest value after exercise. 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

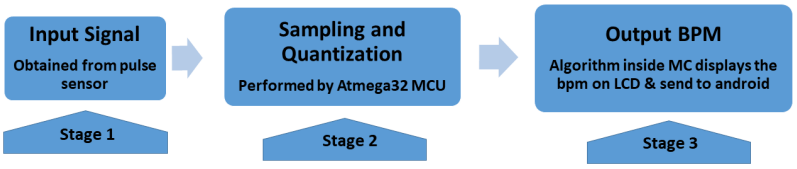


Figure 4.0.1: Methodology in a nutshell

4.1 Sensor - (principle)

Reflection method was the method used in this project where the light emitted from the IR bulb is reflected by the blood vessel and the change in volume is therefore detected as reflected light by the photodiode.

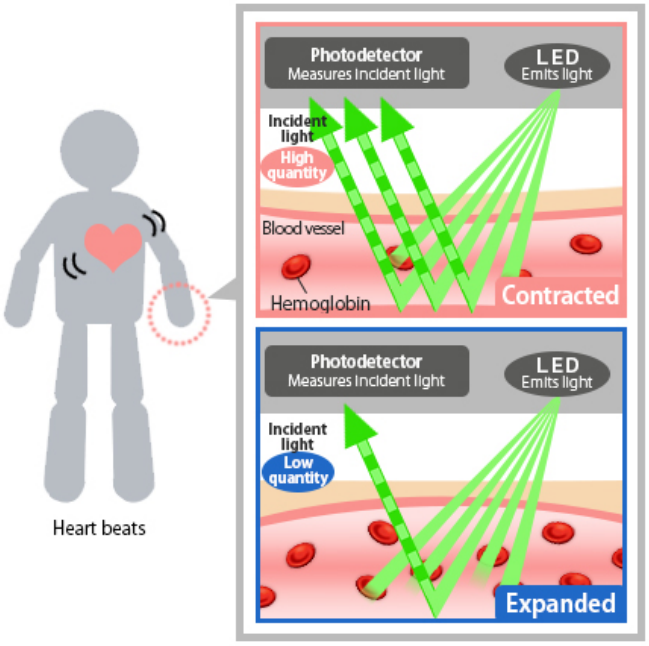


Figure 4.1.1: How IR LED light is reflected from the blood vessels[1]

4.2 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.

4.2.1 Microcontroller

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

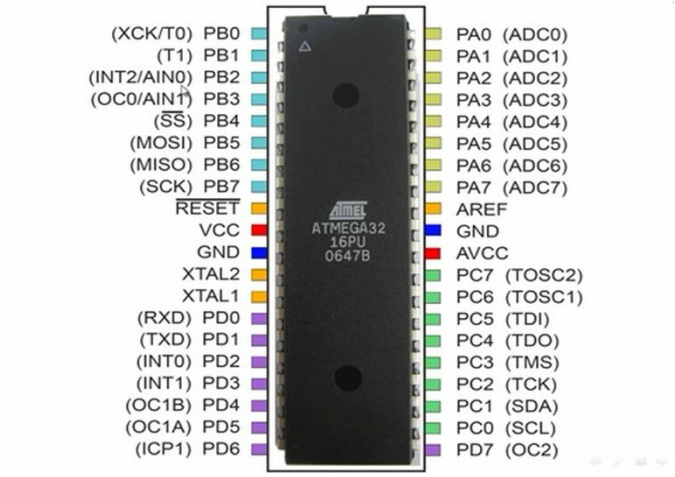
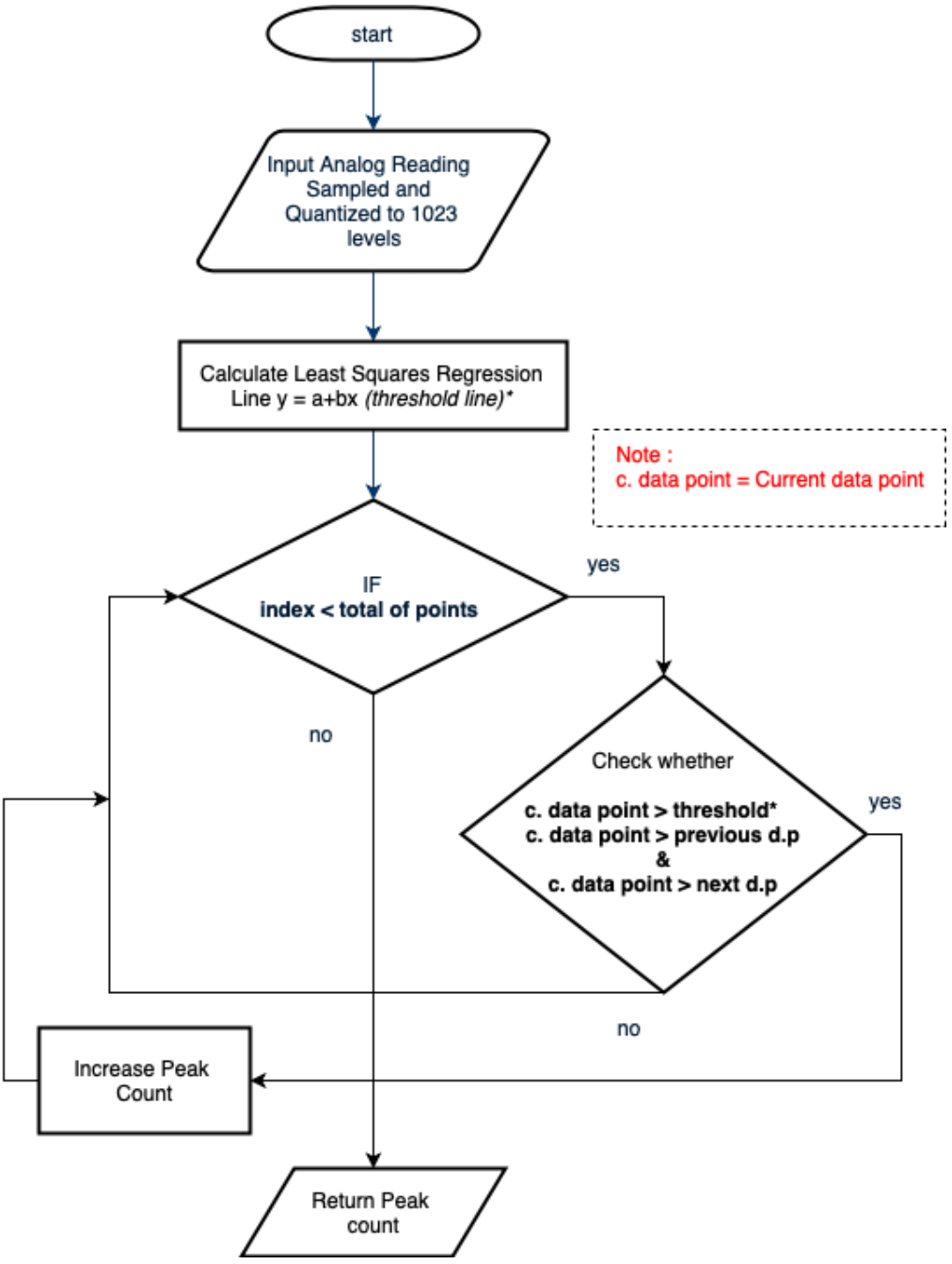


Figure 4.2.1: ATmega32 pinout

The AVR Minimum system board along with the AVR serial programmer was used to program the microcontroller.

4.2.2 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.

4.2.3 Least Squares Regression Line

"Least Squares line is a mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the offsets ("the residuals") of the points from the curve. The sum of the squares of the offsets is used instead of the offset absolute values because this allows the residuals to be treated as a continuous differentiable quantity. However, because squares of the offsets are used, outlying points can have a disproportionate effect on the fit, **a property which may or may not be desirable depending on the problem at hand.**" - mathworld.wolfram.com
Through the MCU algorithm the regression line parameters a and b were calculated Sample regression line:

$$y = a + bx \tag{1}$$

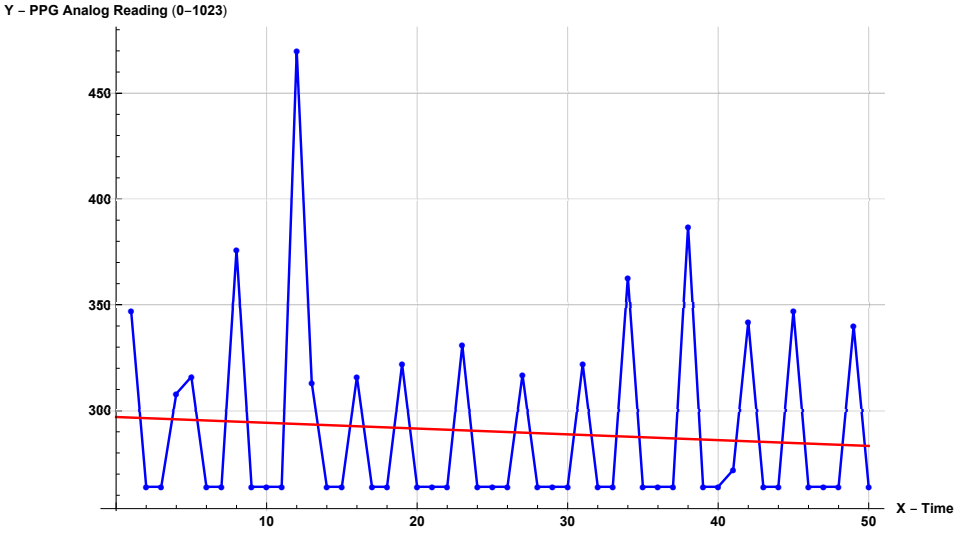
Sample slope:

$$b = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \tag{2}$$

Sample intercept:

$$c = \bar{y} - b\bar{x} \tag{3}$$

Out[ ]=



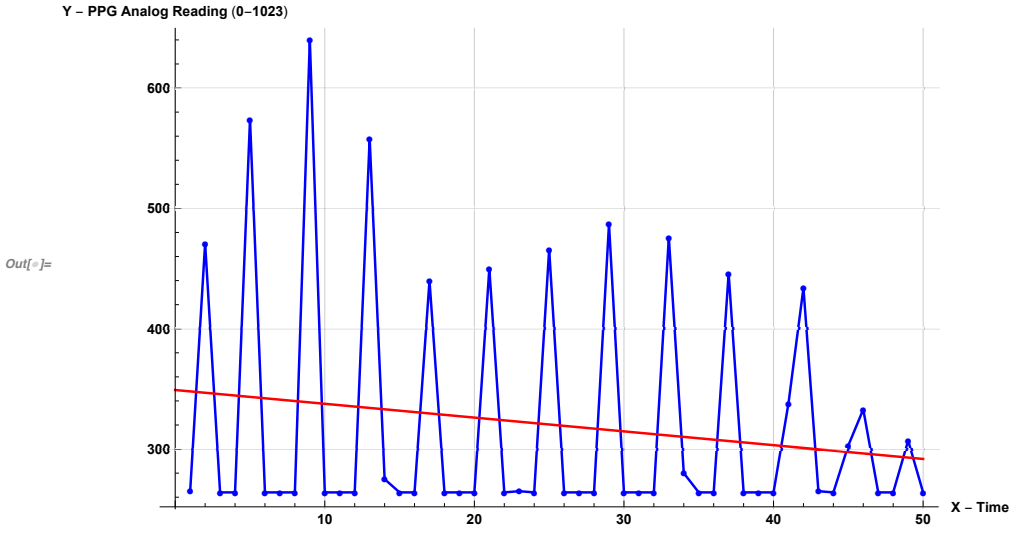


Figure 4.2.2: Determining threshold value line using simple linear regression

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

5 Implementation and Results

5.1 Hardware Design:

Hardware of this project can be considered as integrationof three units: the data acquisition, the MCU,communication and alarm unit.

5.1.1 Data Acquisition Unit:

This unit is mainly responsible for obtaining patient's vital parameters utilizing sensors.Sensors are devices that detect the variations and mainly are two types of them, Optical and solid state sensors.
The sensor used here is pulse sensor.

a. **Pulse sensor:**
It is an Open Source heart rate monitor which considered as a PPG device used to monitor the non-invasive heart rate.It measures the real-time heart beats and calculates BPM with the aid of algorithms implemented by us.
This sensor has two sides, the front one which has a heart shape is the side to be attached to the skin. The pins of the pulse sensors are three as shown in Figure 5.3.1 below.
If the frond side is facing you, then the most left pin is the GND while the middle one is the input voltage which will be connected to the +5v. The last one for outputting the electrical and will be wired with the analog pins of the atmega32 (here we used PA0 as ADC input pin).

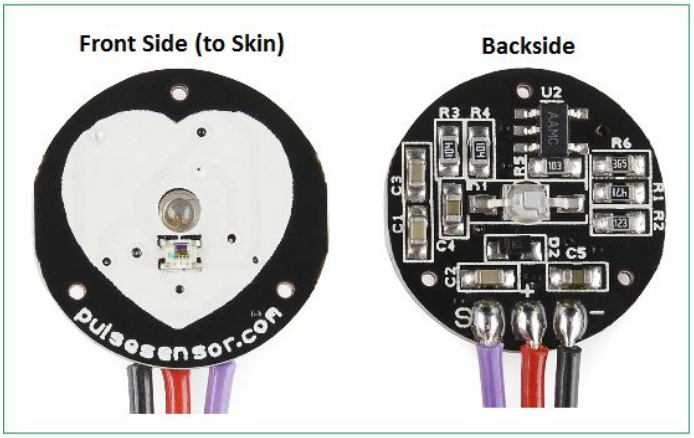


Figure 5.1.1: Pulse Sensor

The Pulse sensor converts the physical PPG into electrical signals. The sensor outputs a raw signal of analog voltage fluctuations, amplifies it and normalize the wave at V/2. With every beat of the heart, a pulse wave travel along all arteries to the tissues where the Pulse Sensor is attached.
When this pulse wave goes under the sensor, the signal experiences a rapid upward rise in its value. It falls back down toward the normal point and before the next pulse sensor goes under the sensor, the signal stabilizes to the ambient noise.
Due to the repetitive characteristic of the pulse wave, the peak is chosen as a reference point because it's recognizable. By applying calculation algorithm on the time between each two successive peaks the heart rate is measured. Ideally we want to find the instantaneous moment of the heart beat for accurate measurements.

According to heart researchers, the instantaneous moment is when the signal gets 25% or 50% of its amplitude. This pulse sensor first measures the IBI when the signal gets 50% of the amplitude, which from the BPM is derived from average of 10 IBI times. [10]

5.1.2 Communication between MCU and Android Device:

The Bluetooth serial port module is responsible the communication. It sends data from the MCU to be displayed in the Android application interfaces

HC-05 Bluetooth to Serial Port Module:
HC-05 is a Bluetooth module which is designed for wireless comunication. This module can be used in a master or slave configuration.
It works on serial communication (USART).It is a 6 pin module.The device can be used in 2 modes; data mode and command mode.The data mode is used for data transfer between devices whereas command mode is used for changing the settings of the Bluetooth module.Hence in this project we will be toying only with the Data Mode. The Command mode will be left to the default settings. The Device name will be HC-05 and the password will be 0000 or 1234 and most importantly the default baud rate for all Bluetooth modules will be 9600.
The module works on 5V supply and the signal pins operate on 3.3V, hence a 3.3V regulator is present in the module itself. Hence we need not worry about it. Out of the six pins only four will be used in the Operating mode. The pin connection is shown below inf figure 5.3.1

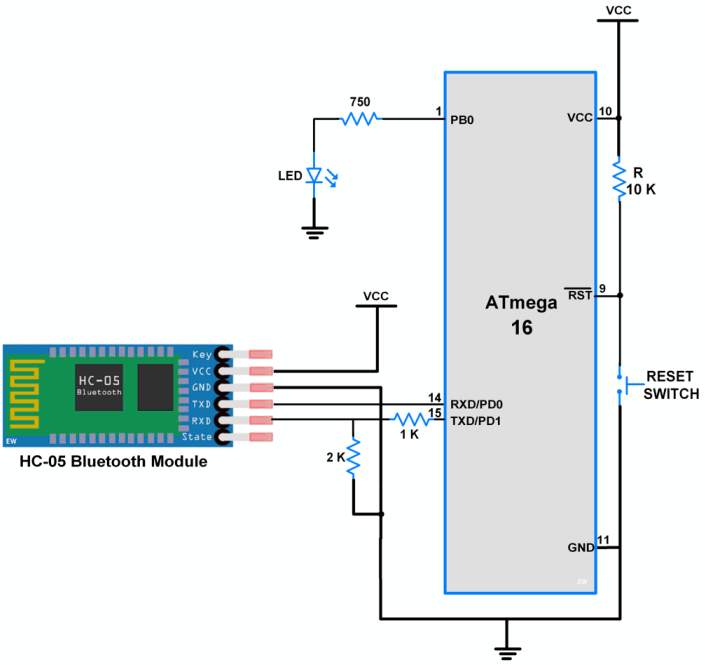


Figure 5.1.2: **HC05 Bluetooth Module Interfacing with ATmega microcontroller**

We take our digital pulse value from Atmega32 and send this data to an android device using HC05 bluetooth module.

5.1.3 Alarm Unit:

This unit is composed of a 5v DC buzzer which will peep if the heart rate increased or went below the specified threshold. The values of the threshold are 150 for maximum HR and 30 for the minimum. But here in this project just to test heart attack portion of our project we set the maximum threshold 76 (It has been hard coded)

Buzzer 5v DC:
Buzzer is an audio signaling device that has many applications includes timers and alarm devices. Many types of buzzers are available mainly they are electromechanical, mechanical and electrical buzzers.

Figure 5.1.3: **Buzzer**

5.2 Software Design:

Several software tools were used throughout the entire development procedures of this project. We used Atmel Studio 7, AVR Burner Tools and HC05 bluetooth terminal android app.

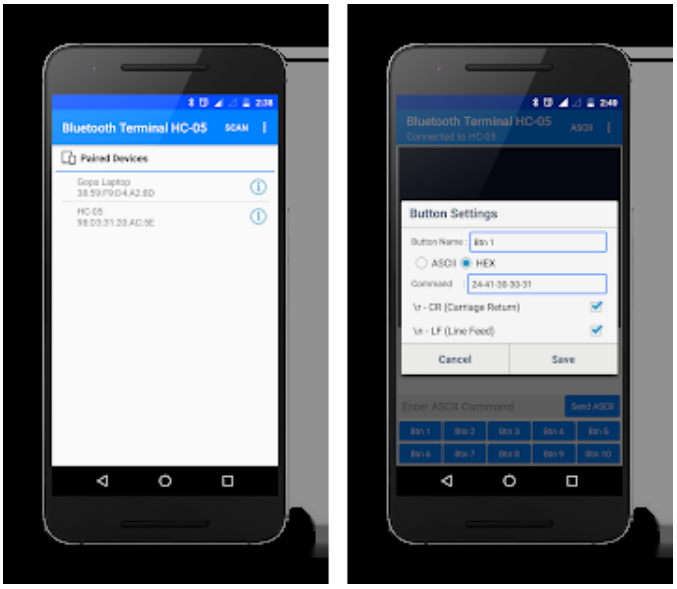


Figure 5.2.1: **Bluetooth Terminal (Android App)**

5.3 Hardware Implementation:

Is done to test the overall system functionalities, the hardware is implemented as it was discussed throughout this chapters with aid of figures including all components and using the software codes. The final system's circuit is shown below in Figure 3.19.

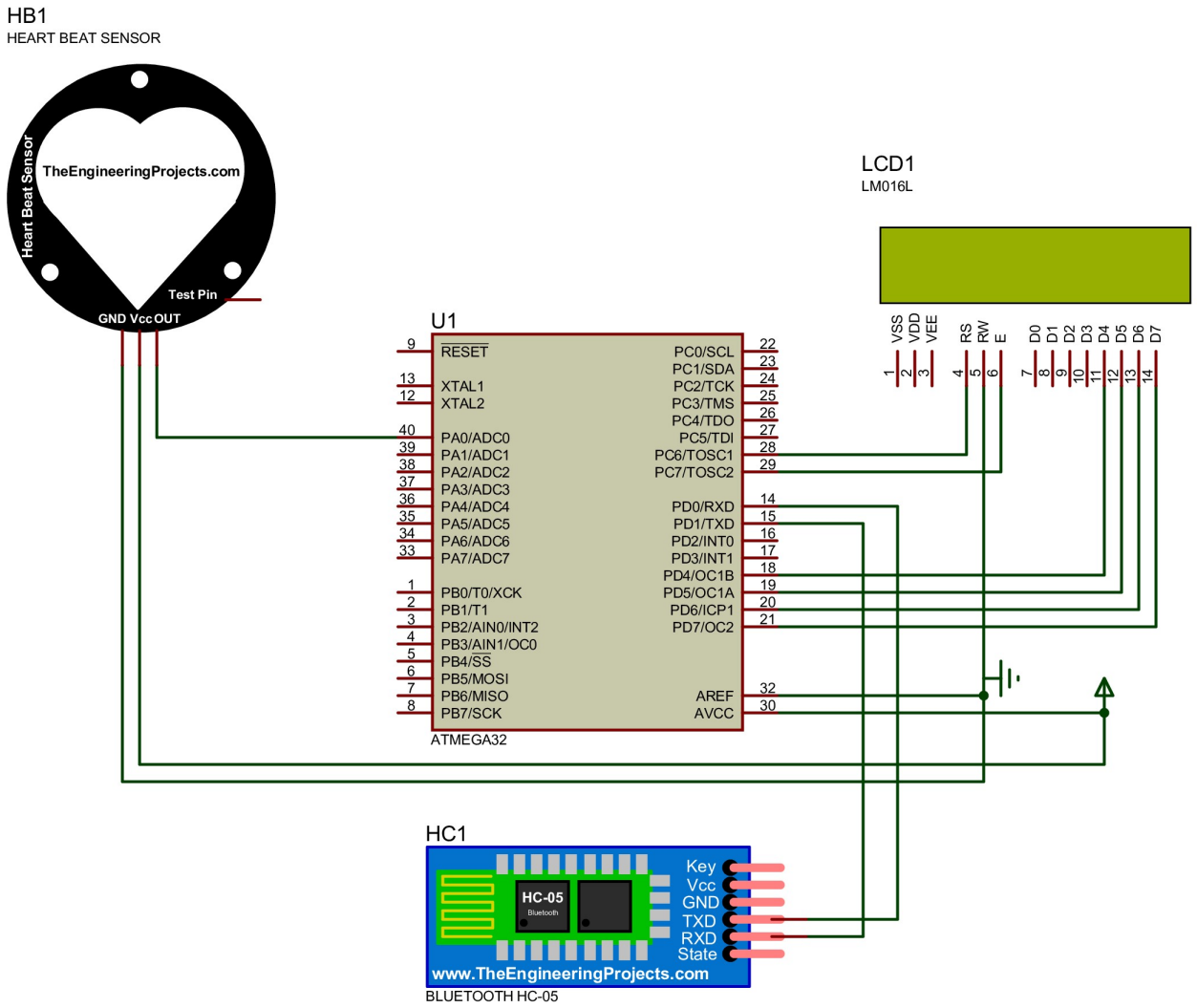


Figure 5.3.1: **Implementation of the Overall System**

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 could able to overcome that issue with the use of an appropriate active bandpass filter(Though we did not use filter here). The most difficult part of the project was to convert the analog signal to 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. This project can also be improved later to measure *body temperature* of a patient besides heart beat

7 Conclusion:

7.1 Conclusion:

A pulse sensor which considered as an infrared sensor that has a response to variations in light intensity instead was used.
The key objective of developing this project with the help of Android Open Source platform is to immediately alert Medical Emergency and the patient's emergency contacts about the health condition of patient.
We are developing prototype of this application using the continuous monitoring of parameters to detect and predict the heart attack and generate an alarm. The buzzer will turn ON when heart rate exceeds or goes below specified threshold level. This objective is met with measuring the heart rate. It is helpful where continuous monitoring is required under critical condition. In addition it is very usable device due to its portability which means the patients can carry it with him therefore no need to stay at hospitals because the Heart Rate Monitor is applicable almost everywhere.
Along with the Heart Rate Monitor, we developed an Android Application that allows both doctors and patients to interact with each other, records the data received from the heart monitor via Bluetooth as well as enable access to these records by the doctor.

7.2 Challenges and Limitations:

Several challenges were faced throughout the entire life cycle of the project. The first one was with the pulse sensor. It didn't detect accurate readings if it was placed with excessive or loose pressure on the body.
n order to have accurate data to some extent, a number of repeated measurements are required
The next challenge was the fact that some of the purchased components were provided without datasheet, specifically the pulse sensor used for this project which made it difficult to fully understand the sensor specifications. Hence, we depended on the basic information supplied by the vendors on their website.

Another challenge was with the Android development environment, which took considerable time for installation and setting up.

7.3 Accomplishments:

On the other hand, this project succeeded in achieving many of its proposed goals. These accomplishments can be summarized as:

- Reading vital signs signals.
- Process these vital signs signals.
- HR measuring and monitoring system.
- Implementing alarm system.
- Providing platform for communication between doctor and patient
- Use of relatively low cost and low power hardware components.

7.4 Future Work:

Further improvements can be applied to this project to enhance its performance:

- Design robust system to improve measuring efficiency even in the presence of noise. In addition to propose a new method for efficient transmission of data between the MCU and the Android application.
- To ensure the accuracy of heart rate monitor device, more testing can be performed to larger number of people with different ages and weights.
- More vital signs parameters should be added to increase the value of the project to the patients. These can include: Blood Pressure,Body Temperature, Respiratory Rate and other parameters.
- Implement pulse and other parameters measurements using the mobile phone camera along with other built-in sensors in order to obtain these parameters on demand if the patient started experiencing some symptoms or abnormalities.
- The MCU should send a control signal along with the measured data when detect a heart attack and the buzzer is turned ON. The control signal should enable GPS, instruct the application to send an SMS containing the measured data and the patient's location to the medical emergency and emergency contacts of the patient in order to get an ambulance and notify his relatives.
- The device should be miniaturized into a PCB making its weight lighter in order to make the device commercial for public use.
- Portable battery unit for the device to provide required power by the sensors and MCU

References

[1] Heart Rate Measurement Technology - Innovation - Epson. https://global.epson.com/innovation/core_technology/wearable/vital_sensing.html. Accessed: 2018-12-30.
[1] Joyce Smith, Rachel Roberts, Vital Signs for Nurses: An Introduction to Clinical Observations, WILEY-BLACKWELL, June 2011
[2] Arrthur c Guyton ,Text Book of Medical Physiology, ELSEVER SAUNDERS
[3]Medical Instrumentations, available [online], http://ww.webmed.com, October 2016
[4] American Heart Association, available [online], http://ww.heart.org, November 2016
[5] Measuring Heart Rate, available [online], http://ww.ACS.co.uk, November 2016
[6] Sudanese Association for Heart Diseases, available [online], http://ww.heart.org, February 2017

8 Appendices -

8.1 MCU Sourcecode in C

```
/*  Group 7 | UoM | ENTC 17Batch */
#ifndef F_CPU
#define F_CPU 8000000UL // 16 MHz clock speed
#endif

/*
#define D0 eS_PORTD0
#define D1 eS_PORTD1
#define D2 eS_PORTD2
#define D3 eS_PORTD3
*/

#define D4 eS_PORTD4
#define D5 eS_PORTD5
#define D6 eS_PORTD6
#define D7 eS_PORTD7
#define RS eS_PORTC6
#define EN eS_PORTC7

#include <avr/io.h>
#include <util/delay.h>

#include "stdlib.h"
#include "string.h"
#include "lcd.h"

/* bluetooth module */
#include "USART_RS232_H_file.h" /* include USART library */

//////////lcd r kaj

void lcd_disp(char data_points[], int r, int c, char w[])
{

   if (w == "clear")
```

```
36        Lcd4_Clear();
37     Lcd4_Set_Cursor(r, c);
38     Lcd4_Write_String(data_points);
39  }
40  ////////////lcd r kaj
41
42  void ADC_Init()
43  {
44     DDRA = 0x00;  /* Make ADC port as input */
45     ADCSRA = 0x87; /* Enable ADC, fr/128 */
46     ADMUX = 0x41; /* Vref: Avcc, ADC channel: 0 */
47  }
48
49  ////////////adc r kaj
50
51  int ADC_Read(char channel)
52  {
53     int Ain, AinLow;
54
55     ADMUX = ADMUX | (channel & 0x0f); /* Set input channel to read */
56
57     ADCSRA |= (1 << ADSC); /* Start conversion */
58     while ((ADCSRA & (1 << ADIF)) == 0)
59        ; /* Monitor end of conversion interrupt */
60     _delay_us(10);
61
62     AinLow = (int)ADCL; /* Read lower byte*/
63     Ain = (int)ADCH * 256; /* Read higher 2 bits and
64               Multiply with weight */
65     Ain = Ain + AinLow;
66     return (Ain); /* Return digital value*/
67  }
68
69  //adc r kaj
70  /////////
71
72  /*************** threshold ***************/
73  double m = 0, c = 0; // gradient and slope
74
75  double sumit(int data_points[], int length)
76  {
77     int i;
78     double sum = 0;
79     for (int i = 1; i <= length; i++)
80        sum += data_points[i];
81
82     return sum;
83  }
84
85  double xysum(int data_points[], int length)
86  {
87     int i;
88     double sum = 0;
```

```
89     for (int i = 1; i <= length; i++)
90        sum += (i)*data_points[i];
91
92     return sum;
93  }
94
95  void regression(int data_points[], double n)
96  {
97     double squarex = (n) * (n + 1) * (2 * n + 1) / 6.0;
98     double xbar = (n + 1) / 2.0;
99     double ybar = sumit(data_points, n) / n;
100
101    m = (xysum(data_points, n) - n * xbar * ybar) / (squarex - n * xbar * xbar);
102    c = ybar - m * xbar;
103 }
104 /*************** end threshold ***************/
105
106 void playBuzzer()
107 {
108    DDRB = 0xFF;
109    PORTB = 0xFF; // Turn ON the Buzzer conneted to PORTC
110    _delay_ms(3000); // Wait for some time
111    PORTB = 0x00; // Turn OFF the Buzzer connected to PORTC
112    _delay_ms(3000); // Wait for some time
113 }
114
115 int main(void)
116 {
117    DDRD = 0xFF; // #
118    DDRC = 0xFF; //for lcd
119    DDRA = 0x00; //Analog input
120
121    DDRB = 0xff;    /* make PORT as output port */
122    USART_Init(9600); /* initialize USART with 9600 baud rate */
123
124    ADC_Init();
125    //ADMUX = 0b01100000; // Configure ADC to be left justified, use AVCC as
           reference, and select ADC0 as ADC input
126    //ADCSRA = 0b10000111; // Enable the ADC and set the prescaler to max value (128)
127
128    Lcd4_Init(); //Initializing the LCD screen
129
130    _delay_ms(6000); //1000
131
132    int i;
133    int thresh = 300;
134    int count = 0;
135    int count2 = 0; //modified peak counting algo
136
137    /*  timing data */
138    double sampling_rate = 0.100; // actually this is the _delay_ms val
139    int time_limit = 10;    //in seconds
140    int se = time_limit / (sampling_rate * 2);
```

```
141     int data_points[se + 1];
142
143     data_points[0] = 0;
144     /* /timing data */
145
146     /* for debugging purposes -> h and l records the peaks */
147     int h = 0;
148     int l = 1023;
149
150     char val[6]; //temporary variable for itoa
151
152     for (i = 0; i < time_limit / (sampling_rate * 2); i++)
153     {
154        data_points[i + 1] = ADC_Read(0);
155
156        itoa(data_points[i + 1], val, 10);
157        //lcd_disp(val,1,0,"");
158
159        ///value to bluetooth module///
160
161        USART_SendString(val);
162        USART_SendString("\n");
163
164        ///value to bluetooth module///
165
166        lcd_disp("Measuring.. *_*", 1, 0, "clear");
167
168        int k = 16 * (data_points[i + 1] - 200) / 375;
169
170        char anim[16] = "";
171
172        int h = 0;
173        for (h = 0; h < k; h++)
174        {
175           anim[h] = '~';
176        }
177        itoa(k, val, 10);
178        lcd_disp(anim, 2, 0, "");
179        _delay_ms(sampling_rate * 100); //100
180     }
181
182     regression(data_points, se);
183
184     //recorded wave form
185     for (i = 0; i < time_limit / (sampling_rate * 2); i++)
186     {
187        thresh = (i + 2) * m + c;
188
189        int a = data_points[i + 1]; //current value
190        char temp[11] = "";
191        char ccount[3];
192
193        if (a > h)
```

```
            h = a; //max peak
        if (a < l)
            l = a; //min peak

        if (a > thresh)
        {
            count += 1; //peak counting]
            if (a > data_points[i] && a > data_points[i + 2])
            {
                count2 += 1;
            }
        }
    }

    //displaying the heart rate
    char bpm[16] = "Heart Rate : ";
    int HeartRatevalue;

    lcd_disp(bpm, 1, 0, "clear");

    itoa(count2 * (60 / time_limit), val, 10);
    HeartRatevalue = count2 * (60 / time_limit);
    if (HeartRatevalue > 76)
        playBuzzer();
    strcat(val, " bpm");
    lcd_disp(val, 2, 4, "");

    _delay_ms(2000);

    USART_SendString("Project Heart Beat Monitoring System\n Heart Beat:");
    USART_SendString(val);
    USART_SendString("\n");
    if (HeartRatevalue > 80)
    {
        USART_SendString("HEART ATTACK!!!!!!!");
    }
}
```

8.2 ATmega32 Datasheet - Summary

Features

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash program memory
  - 1024Bytes EEPROM
  - 2Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega32L
  - 4.5V - 5.5V for ATmega32
- Speed Grades
  - 0 - 8MHz for ATmega32L
  - 0 - 16MHz for ATmega32
- Power Consumption at 1 MHz, 3V, 25°C
  - Active: 1.1mA
  - Idle Mode: 0.35mA
  - Power-down Mode: < 1µA

8-bit AVR® Microcontroller with 32KBytes In-System Programmable Flash

ATmega32
ATmega32L

Summary

2503QS–AVR–02/11