TABLE OF COTENTS

ABSTRACT

The purpose of this project is to develop a prototype of Pulse Measurement System with wireless connectivity. As, world is moving towards smart systems with wireless communications, this project shows the potential of wireless technologies. A mixture of IoT, Cyber-Physical Systems and wireless technologies can revolutionize the future. On sensor connected with a microcontroller can be made smart with intelligent algorithms and codes.
Pulse sensor amped is connected with Arduino Uno board, LCD screen and a HC-05 Bluetooth module. Beats per minute can be measured accurately with this pulse measurement system and the final value of BPM can be seen on LCD screen. Moreover, serial communications are also enabled for computer or an Android device. Serial Plotting of live heartbeat can also be seen on computer or an Android device. Two LEDs are connected, one LED blinks with the heartbeat and one LED fades with each beat. This project is kind of all in one Pulse measurement system.

LIST OF FIGURES

ABBREVIATIONS

STEM	Science, Technology, Engineering and Mathematics
PWM	Pulse Width Modulation
IDE	Integrated Development Environment
BPM	Beat Per Minute
LED	Light Emitting Diode
LCD	Liquid Crystal Display
BT	Bluetooth
IoT	Internet of Things
LDR	Light Dependent Resistance
SRAM	Static Random Access Memory
EEPROM	Electronically Erasable Programmable Read-Only Memory
PCB	Printed Circuit Board
ICSP	In-Circuit Serial Programming
ISP	In-System Programming
OP-AMP	Operational Amplifier
IBI	Inter Beat Interval
I/O	Input/output
DFU	Device Firmware Upgrade
TX/RX	Transmit and Receive
UART	Universal Asynchronous Receiver/Transmitter
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
BPS	Bit Per Second
EDR	Enhanced Data Rate
SSP	Serial Port Protocol
R/W	Read/ Write
PPG	Poly-Plethysmograph
ADC	Analog to Digital Converter

1. Introduction:

Measuring the heart rate is the most basic and frequent human vital sign. It can be measured using electrocardiography, pulse oximetry and other monitoring methods. The simplest method is counting by radical palpation, because the pulse measured by radical palpation should corresponds to the number of heart beats.

This project is about the development of pulse measurement system using microcontroller (Arduino Uno board) and pulse sensor amped. Small LCD screen (2x16) is also connected to display BPM and HC-05 Bluetooth module enables wireless connectivity for serial plotting and serial monitoring. Serial plot shows the live heartbeat, signal value from the pulse sensor and the Inter-Beat-Interval (IBI) on laptop screen or on any android device.

Microcontroller is heart of any embedded system but that also requires strong programming. C is the most widely used programming language for embedded systems with compilers available almost for every microcontroller or microprocessor in market. In this project, Arduino IDE is used to write code and flash the Arduino. Arduino IDE is quite user friendly and based on C language. Algorithm is designed in such a way that LCD will only display the final a most accurate value of BPM after stabilizing, but when we open a live serial plotter it shows the continuously changing values of BPM, signal and IBI. "*BlueGraph*" Android application is used for serial plotting on Android device while connecting via Bluetooth.

2. State of the art:

2.1. Arduino Uno:

Arduino Uno is the most used board in the family of Arduino boards. It is suitable for both beginners and advanced users [1].

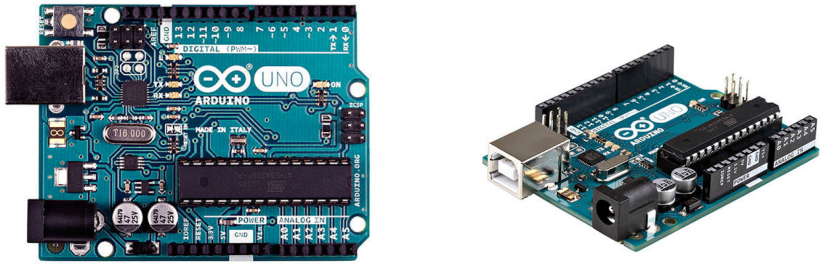

Figure 1: Arduino Uno

The Arduino Uno is a microcontroller board that is based on ATmega328. It is a perfect plug and play microcontroller and contains all necessary things needed to support a microcontroller; simply connect it to computer with USB cable or power it with AC-to-DC adaptor or a battery.

- 14 digital input/output pins. In these 14 I/O pins 6 can be used as PWM (Pulse width
- Modulation) outputs and 6 Analog inputs.
- 16MHz ceramic resonator.
- USB connection.
- Power jack.
- ICSP (In-Circuit Serial Programming)/ ISP (In-System Programming) Header.
- Reset button.

Uno board is different from all previous boards because it does not require any FTDI USB-to-serial driver chip. Instead, it features the ATmega16U2 and that is programmed as a USB-to-serial converter.
Revision 2 of Uno also has resistor pulling out the 8U2 HWB line to ground and that makes it easier to put into DFU (Device Firmware Upgrade) mode [1].
Arduino Uno have two types of serial communications one with TX/RX pins that uses TTL logic levels and the other is SoftwareSerial. Serial is used for communication between Arduino board and a computer or a device. Every Arduino board at least has one serial port those are also known as UART or USART. Serial ports are pin 0 (RX) and 1(TX) on the Arduino Uno and connected via USB to the computer. You can use these two pins either as a digital

Input/ Output or as serial ports but not both functions at the same time can be used.
Arduino Uno has built-in serial commutation pins (0 and 1) that is possible with the support of hardware chip called UART but at the same time the SoftwareSerial function enables to convert any other digital pin on Arduino into a serial port. That's why it is possible in Arduino to have multiple serial ports with speed up to 115200 bps.
In this project SoftwareSerial is used for HC-05 Bluetooth module. Arduino might have multiple SoftwareSerial ports but there are some limitations like you can only receive data from one port at a time [3].

ARDUINO MICROCONTROLLER

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog I/O Pins	6
EEPROM	1 KB
DC Current per I/O Pins	40 mA on I/O Pins; 50 mA on 3,3 V Pin

GENERAL

Input Voltage	7-12 V
Digital I/O Pins	20 (of wich 6 provide PWM output)
PWM Output	6
PCB Size	53.4 x 68.6 mm
Weight	25 g
Product Code	A000066 (TH); A000073 (SMD)

Figure 2: Arduino Uno general information [1]

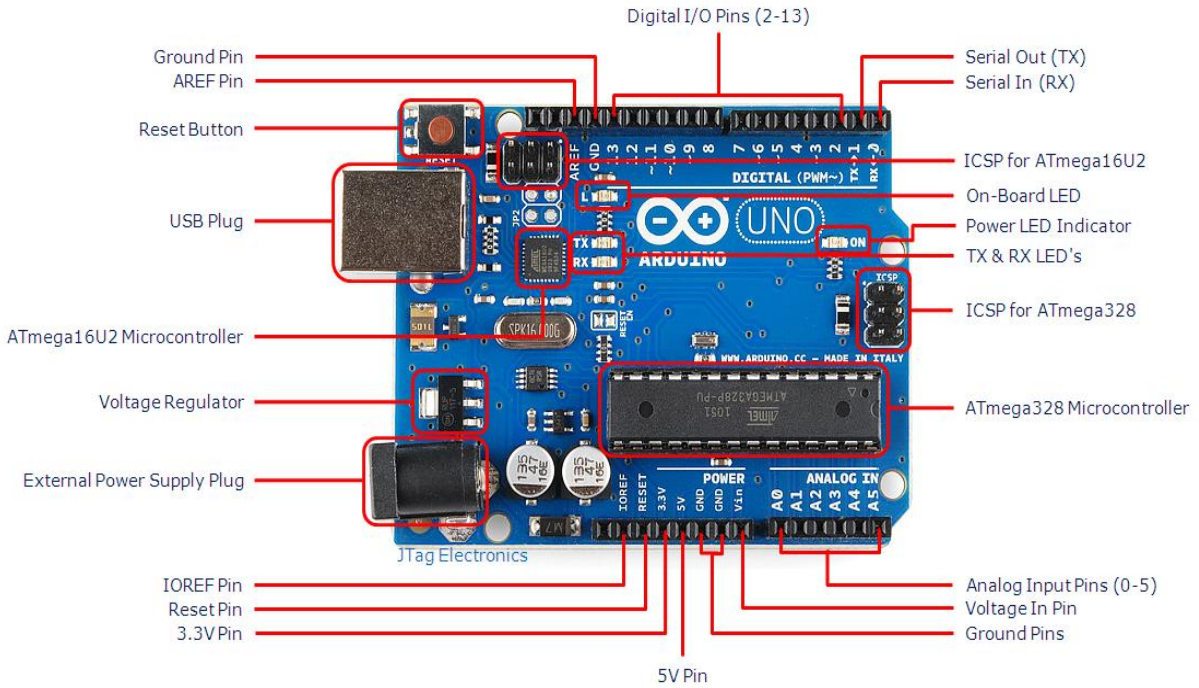

Figure 3: Labeled Arduino Uno [4]

2.2. Arduino IDE:

The Arduino/ Genuino Uno is programmed with Arduino IDE. Arduino Integrated Development Environment or Arduino IDE is an open source Arduino software that makes it easy to write code and upload it to board. Arduino IDE was written in programming language Java. It supports C and C++ using special rules to organize code.

It contains the text editor to write codes, message area, text console and the toolbar with series of menus. The written programs on Arduino IDE are called Sketches and are saved with extension .ino [2].

2.3. Pulse Sensor Amped:

The pulse sensor works on basic principle of optoelectronics. All it contains is a LED (Light Emitting Diode), LDR (Light Dependent Resistor) and a microcontroller to measure the pulse rate.

Pulse rate is the measure of our heart beat, when our heart beats and supplies blood to the body the blood vessels expand and contracts when our blood enters and leaves. Normal heart beat of a healthy person is around 72 to 84 times a minute.

In order to measure the pulse rate with an electronic pulse sensor, LED passes light from one side of our finger and LDR with receive the light on the other side. When the heart supplies blood and passes it from our vessels the light will be absorbed by the red blood cells and light will not be received at LDR and the vice versa. When LDR receives less light the value of resistance will be increased. This value of resistance is converted into voltage variation using a signal conditioning circuit usually an OP-AMP (Operational Amplifier).

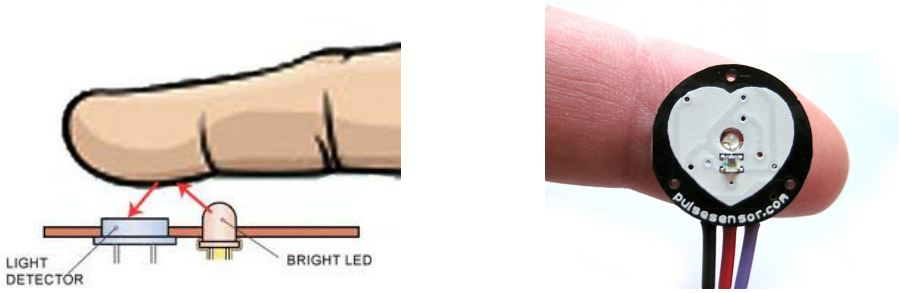

Figure 4: Pulse sensor amped

Pulse sensor Amped is a plug-and-play heart rate sensor for Arduino and Arduino compatible devices. General specifications of pulse sensor amped.

Diameter	0.625”
Overall thickness	0.125”
Working voltage	3V to 5V
Working current	~4mA at 5V

Table 1: Pulse sensor general information

In figure pulse sensor contains three wires purple, red and black that shows S, + and – respectively.

S (Purple wire)	Signal, connected to microcontroller’s digital input
+ (Red wire)	Supply, 3V to 5V
– (Black wire)	Ground

Table 2: Pulse sensor I/O wires

2.4. Bluetooth module (HC-05):

HC-05 module simply can be used as Bluetooth SPP (Serial Port Protocol) module. It is especially designed for transparent wireless serial connection setup. HC-05 is fully qualified V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with 2.4GHz radio transceiver and baseband. Originally it does not have pin headers and should be soldered on a breakout board to give connections efficiently [7].

Main features of HC-05 module are:

- HC-05 can be set to Master or Slave.
- Mostly runs on 3.3V.
- Module has two modes of operation, Command mode where AT commands can be sent to configure the module (AT Mode is not described in this project) and the Data mode where it transmits and receives data to another module.
- Range is approximately 10 meters (30 feet).
- Integrated antenna and edge connectors.
- Supported baud rates are: 9600,19200,38400,57600,115200,230400,460800 bps. Default baud rate is 9600bps while in this project baud rate is set to 115200 bps for fast communication.

Figure 5: HC-05 Bluetooth module

3. Pulse measurement system:

Pulse measurement system is designed in two main steps. First step is to design a circuit, how Arduino board can be connected with pulse sensor, Bluetooth module, LCD, jumper wires, LEDs, Potentiometer and a computer. All of these required a specific circuit design to connect them efficiently and collect data. Second step is to write a strong C language code in Arduino IDE to flash the microcontroller that's the main part where magic happens. A single mistake in code or inappropriate algorithm can give the unpleasant results.

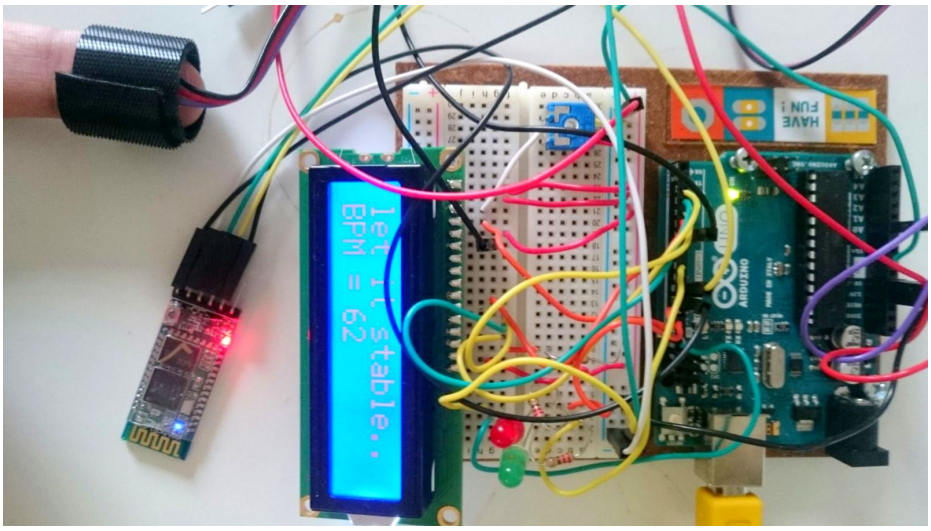

Figure 6: Pulse measurement system with Arduino

First step can be further divided into more steps. In this project connections for each module are explained separately for better understanding. Furthermore, the code and the algorithm is explained after the connections.

3.1. LCD connections:

A small size 2x16 LCD is connected to show the BPM on screen to use the pulse sensor system when it is not connected with any computer or an Android device. *LiquidCrystal* library in Arduino IDE can be used to control the LCD display and compatible with the Hitachi HD44780 driver.
LCDs have a parallel interface and microcontroller manipulates several interface pins at once to control the display. The interface consists of following pins [5].

- **Register Select (RS) pin:**

It controls the data register or the instruction register of the LCD's memory where you are writing data. Data register hold the things that goes on screen and the instruction register looks for the instructions on what to do next, both of them can be used.

- **Read/ Write (R/W) pin:**

It selects reading or writing mode.

- **Enable pin:**

That enables writing to the registers.

- **Data pins:**

The state (high or low) of 8 data pins (D0 – D7) are the bits you are writing to a register when you write or the values you are reading when you read.

There are some more pins like **display contrast pin (Vo)**, **power supply pins (+5V and Gnd)** and **LED backlight (Bklt+ and Bklt-)** those can be used to control display contrast, power the LCD and turn on or off the backlight respectively.
Hitachi compatible LCDs can be controlled in two modes: 4-bit and 8-bit. The 4-bit mode requires 7 I/O pins from Arduino and 8-bit mode requires 11pins. In this project 4-bit mode is used because this mode is able to do almost everything [5].

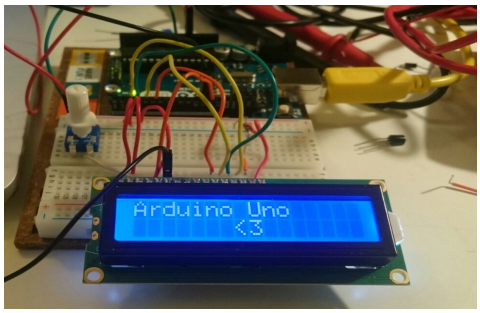

Figure 7: LCD with Arduino Uno

3.1.1. Hardware required:

Hardware required to connect 2x16 LCD in 4-bit mode with Arduino board is described below:

- Arduino Uno.
- LCD screen.
- Potentiometer 10K ohm.
- Resistor 220 ohm.
- Hook-up wires.
- Bread board.

3.1.2. Circuit:

Pins connected to Arduino and LCD with the schematic diagram is shown below:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 6
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2

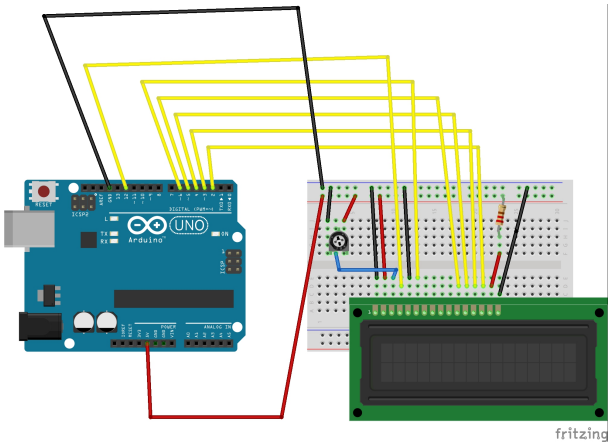

Figure 8: Circuit Design (Developed on Fritzing [6])

The code for LCD is defined in the coding section of this project.

3.2. Pulse Sensor Amped connections:

Pulse sensor amped connection with Arduino is very simple. It has three wires as described earlier. Red, Black and Purple wires should be connected with 3.3V pin, Ground and input pin (A0) respectively.

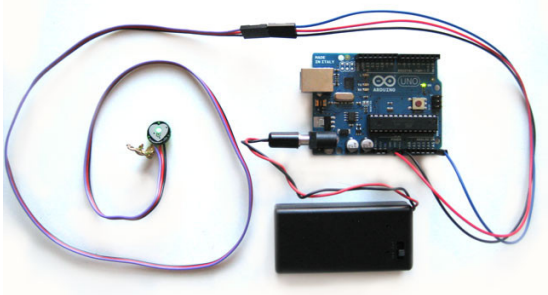

Figure 9: Pulse sensor connected with Arduino and powered with battery

Precise voltage for pulse sensor is very important, increase in voltage can not only damage the sensor but also can give wrong and unexpected results. In this project pulse sensor is connected with 3.3V pin. Figure 7, describes the simple connections of pulse sensor with Arduino when the Arduino is powered with external battery, but that does not matter if the board is powered with USB or external battery the connections of pulse sensor remains same with the same code.
Detailed Algorithm and the code is defined in the coding section of this project.

3.3. Bluetooth module (HC-05) connections:

HC-05 Bluetooth module connections are very simple. Serial communication is possible with module's TX/RX pins of module. It is assumed that the module is already configured with AT mode.

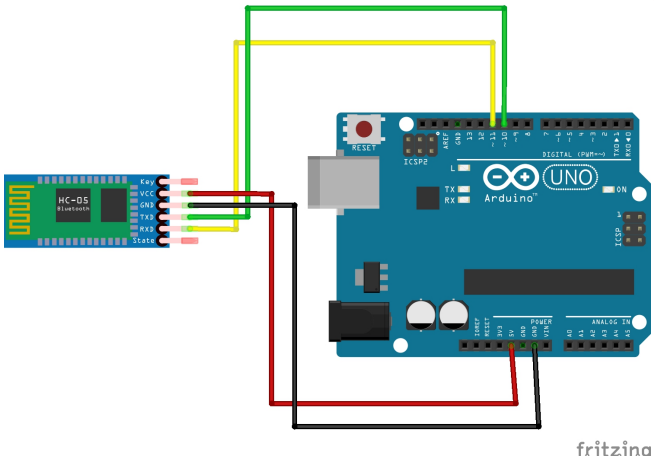

Figure 10: HC-05 with Arduino (Arduino's SoftwareSerial function is used)

HC-05 Bluetooth module pin description is:

- TX (Pin 1) to Digital pin 10 *(SoftwareSerial)*.
- RX (Pin 2) to Digital pin 11 *(SoftwareSerial)*.
- VCC/ 3.3V (Pin 12) to 5V pin.
- GND (Pin 13) to GND pin.
- KEY (Pin 34): should be connected in AT Mode only.

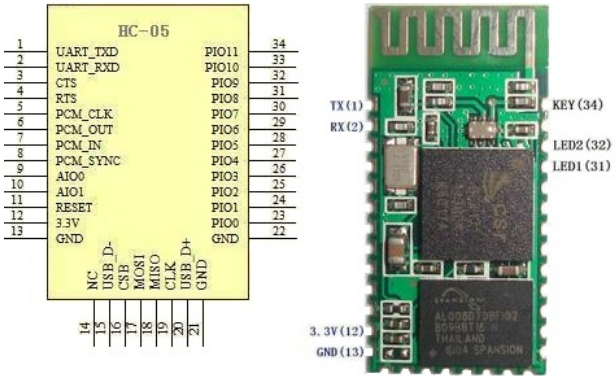

Figure 11: HC-05 Bluetooth module Pin description [9]

Coding for HC-05 Bluetooth module is explained later in the coding section, that involves softwareSerial, serial plotting and Android application (BlueGraph).

3.4. LED connections:

Pulse sensor system in this project also has two LEDs connected with Arduino to show physical output. LED cannot be connected directly to the output pin of Arduino because more current can burn the LED. A 220-ohm resistor is connected before each LED to limit the current through it.

Two LEDs are connected in this project.

- BlinkPin (PIN 13): LED blinks on User's live heartbeat.
- FadePin (PIN 8): LED fades on User's live heartbeat

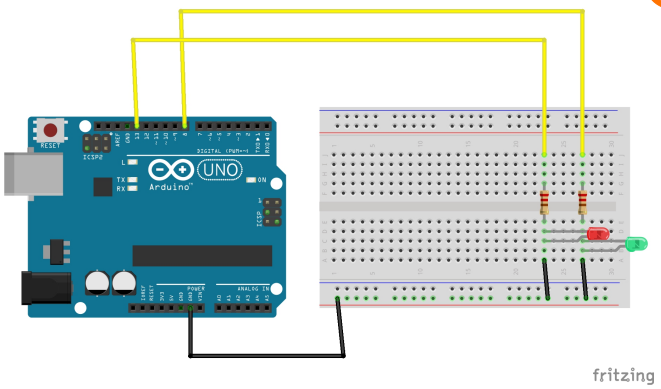

Figure 12: LED connections

4. Coding:

Coding and Algorithm is the mind of pulse sensor system, where magic happens. Code is based on C language and written in Arduino IDE, as discussed earlier. Coding is done for each and every part of pulse sensor system Pulse sensor amped, Bluetooth module, LCD, LEDs and for serial communications. Code can be explained in sections for better understanding.

Code and Algorithm for Pulse sensor amped is of most importance because sensor is the most important part. It collects data and send for processing and plotting.

It is very essential to know about the kind of signal from pulse sensor before generating an algorithm. Pulse sensor generates photo-plethysmograph (PPG). This graph is an analog fluctuation in voltage that is generated in response to the light intensity. If the amount light of incident on the sensor remains constant, signal value also remains constant 512 close to the midpoint of ADC range (Arduino has 10-bit ADC) and if quantity of light is more, the signal goes up and the vice versa [10].

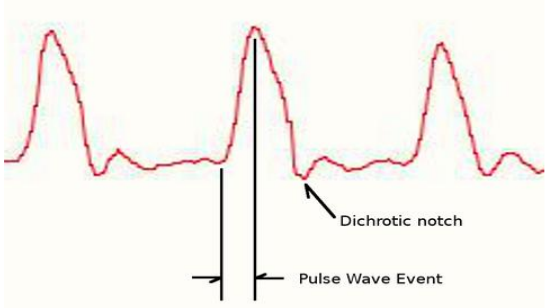

Figure 13: Signal from the pulse sensor [10]

In coding for pulse sensor, the main is to find Inter Beat Interval (IBI) form the PPG and with IBI we can calculate pulse rate or BPM.

When heart beats a wave generated even in very small human arteries, figure shown below. Let's say the initial point of wave is pint 'T' and the maximum point or peak is 'P'. Some researcher says that the initial point 'T' should not be the base value, it should be the 50% of peak to avoid the heart noises that value should be taken. Code is generated here according to this consideration, measure IBI by timing between moments when the signal crosses 50% of wave amplitude during upward rise. Inter Beat Interval can be calculated from interval between two consecutive peaks.

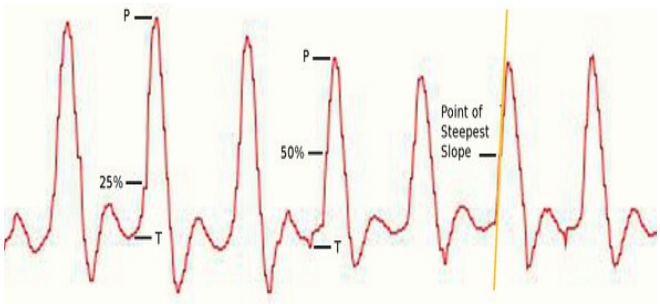

Figure 14: Pulse sensor PPG

In the coding section, first declare all variables.

```
int pulsePin = 0;                 //Pulse sensor purple wire is connected to pin 0
int blinkPin = 13;                //Pin to blink LED with each beat
int fadePin = 8;                  //Pin that controls fading LED with each beat
int fadeRate = 0;                 //used to fade LED with PWM on fadepin
volatile int BPM ;                //That holds raw value of beats per minute (BPM), updated after every 2mS
volatile int Signal;              //holds the incoming raw signal
volatile int IBI = 600;           //Inter beat Interval
volatile boolean Pulse = false;   //"True" when User's live heartbeat is detected. "False" when not a "live beat".
volatile boolean QS = false;      //becomes true when Arduoino finds a beat
volatile int justStarted = 0;     //counter to collect specified no. of value of BPM
```

lines with "//" are comments, not the part of code.

```
volatile int rate[10];                    //Array to hold latest 10 values of IBI
volatile unsigned long sampleCounter = 0;     //to determine pulse timing
volatile unsigned long lastBeatTime = 0;      //to find IBI
volatile int P =512;                      //to find peak in pulse wave
volatile int T = 512;                     //to find trough in pulse wave
volatile int thresh = 512;                //to find instant moment of heartbeat
volatile int amp = 100;                   //to hold amplitude of pulse waveform
volatile boolean firstBeat = true;        // used to seed rate array so we startup with reasonable BPM
volatile boolean secondBeat = false;      // used to seed rate array so we startup with reasonable BPM
```

It is also very important now to decide how fast the pulse sensor take values from User. An interrupt rate should be designed that have high enough resolution to get reliable measurements. In Arduino Uno, ATmega328 microcontroller has 8-bit hardware timer (Read ATmega328 Data sheet for details, as it is separate topic and cannot be discussed in this project) that gives 500Hz sample rate or beat to beat timing resolution is 2mS.

Code for setting interrupt rate for Arduino Uno should be:

```
void interruptSetup(){
  TCCR2A = 0x02;
  TCCR2B = 0x06;
  OCR2A = 0x7C;
  TIMSK2 = 0x02;
  sei();
}
```

Now the next step should be to correct the ISR vector. As Arduino has ATmega328 microcontroller, the ISR vector for Timer2 could be like this:

```
ISR(TIMER2_COMPA_vect){
  cli();
  Signal = analogRead(pulsePin);          //read the Pulse sensor
  sampleCounter += 2;                     //Keep track of time in mS with this variable
  int N = sampleCounter - lastBeatTime;   //monitor the time since last beat, to avoid the noise
```

ISR vector in the upper code enables sensor take values after every 2mS and detect beats. Signal would be the analog read of sensor connected at analog pin (A0) and *sampleCounter* is used to keep track of time. Track the highest and the lowest values of PPG wave to get accurate amplitude.
'P' is the peak and 'T' is the trough. 'N' is used to avoid noises. 3/5 IBI that must pass before 'T' gets updated as a way to avoid noise and false readings.

```
if( (Signal < thresh) && N > (IBI/5)*3){      //avoid dichrotic noise by waiting 3/5 of last IBI
      if (Signal < T){                        //T is the trough
        T = Signal;                           //keep the track of lowest point in pulse wave
      }
    }
    if(Signal > thresh && Signal > P){        //thresh conditions help avoid noises
        P = Signal;                           //P is the peak
      }
```

Let's check either we have a pulse.

```
if (N > 250){                                          //to avoid high frequency noise, as N is the time since last beat in mS.
   if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
   Pulse = true;                                       //set the pulse flag when there is an expected pulse
   digitalWrite(blinkPin,HIGH);                        //turn on the pin 13 LED
   IBI = sampleCounter - lastBeatTime;                 //time between beats in mS
   lastBeatTime = sampleCounter;                       //for the next value the last beat turns into sample counter
```

In the upper code, to avoid frequency noise 250 millisecond minimum 'N' should be passed. Moreover, there would be a pulse when the waveform rises past the thresh value and 3/5 of last IBI. Time to set pulse flag and *pulsePin* LED would be turn on. IBI can be calculated easily as shown above.
Next step is to verify that either the collected *firstBeat* is real or not, if the *firstBeat* is real the process should be shifted to the *secondBeat* otherwise return the process to the start.

```
if(secondBeat){                    //if this is the second beat, if secondBeat == TRUE
     secondBeat = false;           //secondBeat flag
     for(int i=0; i<=9; i++){      //seed the running total to get realistic BPM at startup
       rate[i] = IBI;
     }
   }
              if(firstBeat){          //if it's the first time we found a beat, if firstBeat == TRUE
                 firstBeat = false;   //firstBeat flag
                 secondBeat = true;   //secondBeat flag
                 sei();               //enables interrupt again
                 return;              //IBI value is unreliable, discard it
              }
```

Once the process has been passed from all earlier steps, the IBI value can be trusted. Now, seed an array (rate []) of ten latest values of IBI to calculate BPM.
Let's calculate BPM.

```
  word  runningTotal = 0;          //clear the runningTotal variable
    for(int i=0; i<=8; i++){       //shift data in the rate array
      rate[i] = rate[i+1];         //drop the oldest IBI value
      runningTotal += rate[i];     //add up the 9 oldest IBI values
    }
    rate[9] = IBI;                 //add the latest IBI value to rate array
    runningTotal += rate[9];       //add the latest IBI value to runningTotal
    runningTotal /= 10;            //calculate average of last 10 IBI values
    BPM = 60000/runningTotal;      //calculate BPM
    QS = true;                     //Qualified flag
    if (BPM > 0 && BPM < 110){     //Set value counter for LCD to display very accurate value of BPM
     justStarted++;
      }

    }
}
```

Now the BPM has been calculated and can be shown on plot or LCD but there are still some loose ends like, finding no-beat.

```
if (Signal < thresh && Pulse == true){   //values are going down, the beat is over
   digitalWrite(blinkPin,LOW);           //turn off pin 13 LED
   Pulse = false;                        //reset the pulse flag to do it again
   amp = P - T;                          //get amplitude of pulse wave
   thresh = amp/2 + T;                   //set thresh at 50% of amplitude
   P = thresh;                           //reset these for next time
   T = thresh;
 }
```

Some more things to discuss before the ISR is done like, if there are no beats.

```
  if (N > 2500){                          //if the 2.5 seconds are gone without a beat
     thresh = 512;                        //set thresh default
     P = 512;                             //set P default
     T = 512;                             //set T default
     lastBeatTime = sampleCounter;        //bring the lastBeatTime up to date
     firstBeat = true;                    //set these to avoid noise
     secondBeat = false;                  //get heartbeat back

   }
   sei();                                 //enable interrupts when it's done!
}
```

As, code in Arduino IDE consist of two main sections. First section is the **void *setup()*** and second is the **void *loop()***. Functions to be called in both sections are:

- **Void setup ():**
  - Serial handling (begin).
  - Interrupt setup.
  - LCD starting display.
- **Void loop ():**
  - Serial handling (output/ print).
  - Serial output when heart beats.
  - LCD BPM display.
  - LED fade to beat.

The main part of code in Arduino IDE consisting all above functions would be like this:

```
#include <LiquidCrystal.h>    //LCD library
#include <SoftwareSerial.h>   //SoftwareSerial library
#define Serial_Monitor 1      //Option 1 for output type
#define Serial_Plotter 2      //Option 2 for output type

SoftwareSerial Genotronex(10, 11); //SOftwareSerial pins for BT module and give then random name (Genotronex) // RX with pin 11, TX with pin 10
```

Selecting the output type and LCD Pin connections.

```
static int outputType = Serial_Plotter;           // for serial monitor set here output = Serial_Monitor

LiquidCrystal lcd(12, 6, 5, 4, 3, 2);             //LCD PIN connection
```

Void setup() section:

```
void setup() {
  pinMode(blinkPin,OUTPUT);
  pinMode(fadePin,OUTPUT);
  Serial.begin(115200);
        Genotronex.begin(115200);                              // Software serial begin for plotting on Android App
        Genotronex.println("Data from Pulse Sensor is here: (BPM, IBI, Signal) ");
  lcd.begin(16, 2);
  interruptSetup();
//  analogReference(EXTERNAL);
  lcdStartingDisplay();                        //LCD Display starting lines

}
```

Void loop() section:

```
void loop() {

serialOutput();

  if (QS == true){

    fadeRate = 255;

serialOutputWhenHeartBeats ();
lcdBPMDisplay ();


QS = false;


}

   ledFadeToBeat();
   delay(20);
}
```

The upper all functions used in *Void setup()* and *Void loop()* section can be explained separately.

4.1. Serial handling (begin):

In *Void setup()* section serial handling is just about initializing the serials, by using ***Serial.begin(115200)*** command for serial monitor and ***Genotronex.begin(115200)*** for softwareSerial to Bluetooth module. Where 115200 is the baud rate, as fast communication is required to be done.

4.2. Interrupt setup:

In *Void setup()* section InterruptSetup has also been called, InterruptSetup() function is defined in the previous coding section for Pulse sensor amped.

4.3. LCD starting display:

In *Void Setup()* section lcdStartingDisplay() function has been called, that can be seen below.

```
// Defining function for initializing LCD displays and statement

void lcdStartingDisplay(){
   lcd.setCursor(0,0);
   lcd.print("..Pulse Sensor..");
   lcd.setCursor(0,1);
   lcd.print(".....System.....");
   delay(3000);
   lcd.clear();

   lcd.begin(16,2);
   lcd.print("....Designed by Hassan Ali Bukhari");
   for (int i=0; i<19; i++ ){
    lcd.scrollDisplayLeft();
    delay(350);
   }
delay(500);
   lcd.clear();
   lcd.print("....Thanks to Prof. Rebner and Prof. Thies for support ");
   for (int i=0; i<23; i++ ){
    lcd.scrollDisplayLeft();
    delay(350);
   }
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Calculating...");
}
```

4.4. Serial handling (output/ print):

There are two main types of serial outputs, first is serial monitor and second is serial plotter. Furthermore, both types of serial outputs also consist softwareSerial commands to monitor and plot data on Android device via Bluetooth connection.

```
//defining two functions:
// 1. for serial monitor for process visualization.(Signal, BPM and IBI)
// 2. for serial plotting seial plotting (also for bluetooth serial plotting using mobile application "BlueGraph")

void serialOutput(){
  switch(outputType){
    case Serial_Monitor:
    sendDataToProcessing ('S', Signal);
    break;

    case Serial_Plotter:
      Serial.print(BPM);
            Genotronex.print(BPM);        //software serial for BT-05
      Serial.print(",");
            Genotronex.print(",");        //software serial for BT-05
      Serial.print(IBI);
            Genotronex.print(IBI);        //software serial for BT-05
      Serial.print(",");
            Genotronex.print(",");        //software serial for BT-05
      Serial.println(Signal);
            Genotronex.println(Signal);   //software serial for BT-05
     break;
     default:
     break;
  }
}
```

Serial output when heart beats function is.

```
// defining function for serial monitor as above but seial will show "BPM" and "IBI" when heart beats.

void serialOutputWhenHeartBeats (){
  switch (outputType){
    case Serial_Monitor:
    sendDataToProcessing ('B', BPM);
    sendDataToProcessing ('Q', IBI);
    break;
    default:
    break;
  }
}
```

Definition of function for send data to processing.

```
//Defining function to sending data to seial monitor

void sendDataToProcessing(char symbol, int data ){
  Serial.print(symbol);
  Serial.println(data);
}
```

4.5. LCD BPM display:

When it comes to BPM, LCD should display most accurate value of BPM after stabilizing rather changing BPM value after each second. This code is written for the same purpose to get one value of BPM after it stabilize.

```
//Another simple function for  LCD BPM display

void lcdBPMDisplay(){

            if(justStarted > 30 && justStarted < 60){            //Printing BPM on LCD after 10 beats are captured
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("let it stable..");
            lcd.setCursor(0,1);
            lcd.print("BPM = ");
            lcd.print(BPM);
            }
}
```

4.6. LED fade to beat:

There is also one last function to be called in *Void loop()* section, it controls the fading of LED with heartbeat.

```
void ledFadeToBeat(){                    //LED fade function
  fadeRate-= 15;                         //set LED fade value
  fadeRate= constrain(fadeRate,0,255);   //LED fade from going into negative numbers
  analogWrite(fadePin, fadeRate);        //fade LED
}
```

5. Android connectivity:

Pulse sensor system can also be connected with Android device via Bluetooth. HC-05 Bluetooth module enables it to plot User's live heartbeat on mobile phone screen. "BlueGraph" application has been used with successful results. This technique behind plotting is very common, send data to the device via serial communication. BPM, IBI and Signal values can be sent easily to the Android device with 115200bps baud rate and the Android app will plot these three things accordingly. A separator "," must be used between BPM, IBI and Signal values while sending it to device. In Figure 15, Red, Green and Blue lines show the IBI, Signal and BPM respectively [11].

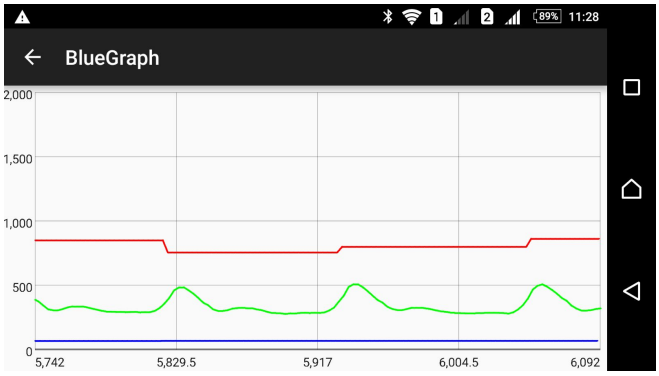

Figure 15: Plotting on "BlueGraph" Android app

6. Applications:

- Medical instruments.

- Sports.

- Health care.

7. Risks:

- Carefully connect all jumper wires on breadboard and Arduino pins, loose connections can lead to bad results.
- HC-05 Bluetooth module should be soldered with breakout board that has pin headers, do not solder wires direct to the module it can damage the module because of heat.
- LEDs are very sensitive more current can damage them, restrict the current by using resistors.
- Assemble everything in an organized way, no wires should be connected with each other.
- Keep the board and pulse sensor away from extensive heat or water.
- Use transparent lamination for pulse sensor, do not try it on oily surface.
- Do not connect TX/RX serial communication pins directly to the RS232 serial port because these pins work on 5 - 3.3V depending upon the board while RS232 operate at +/- 12V and can damage the board.
- Do not connect the Pulse sensor with higher voltage I can damage the sensor and can also give wrong values.
- Power the Arduino via USB or do not power it with battery more than 5V.
- Do not move while measuring BPM, because the sensor is light sensitive and can give bad results in worst case.

8. Conclusion:

The main goal of this project was to develop a prototype of pulse sensor system with wireless connectivity in which data can be accessed remotely, but this project is also a roadmap for the future oriented smart and efficient medical instruments in which data can be accessed remotely via cloud. It shows the potential of embedded systems, IoT and wireless technologies. Pulse rate can be measured accurately with this pulse measurement system and live heartbeat can be seen on computer or an Android device.

In future it is also possible that a human heart or a pacemaker is connected with an Android device and even the doctor can remotely have access to the patient's health status via cloud. A sensor can be made smart when connected with a microcontroller and decision making power can be enabled with intelligent codes and algorithms. World is moving towards smart, efficient and intelligent systems. In near future, homes, cities, industries and sensors are going to be replaced with smart homes, smart cities, smart industries, smart sensors.

References:

1. https://www.arduino.cc/en/main/arduinoBoardUno
2. https://www.arduino.cc/en/Main/Software
3. https://www.arduino.cc/en/Reference/softwareSerial
4. http://www.jtagelectronics.com/wp-content/uploads/2015/08/Arduino-Uno-R3-with-Part-Labels.jpg
5. https://www.arduino.cc/en/Tutorial/HelloWorld
6. http://fritzing.org/account/login/?next=/projects/create/
7. https://www.itead.cc/wiki/Serial_Port_Bluetooth_Module_(Master/Slave)_:_HC-05
8. https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To
9. https://developer.mbed.org/media/uploads/edodm85/1-881-.jpg
10. https://pulsesensor.com/pages/pulse-sensor-amped-arduino-v1dot1
11. https://play.google.com/store/apps/details?id=sakulstra.androidbluetooth&hl=en