

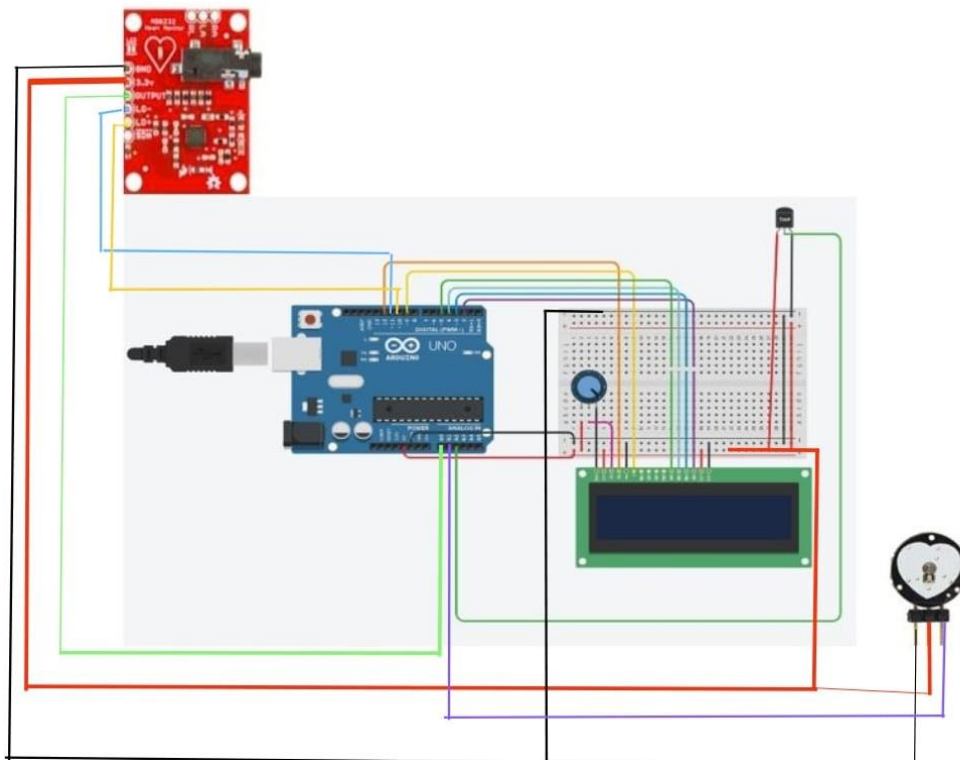
HEALTH MONITORING SYTEM

AIM: To interface arduino with pulse sensor, LM35 temperature sensor, liquid crystal display(LCD),and ECG sensor and design a GUI to display the ECG graph

COMPONENTS REQUIRED:

- Arduino UNO board
- USB Power Cable
- Pulse sensor
- AD8232 Heart Rate monitor
- LM35 Temperature sensor
- 16X2 LCD
- Connecting wires
- Potentiometer
- LED

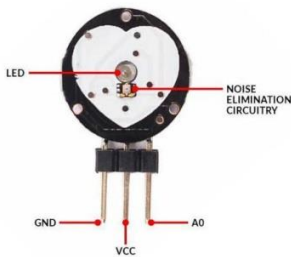
CIRCUIT DIAGRAM:



SENSORS USED :

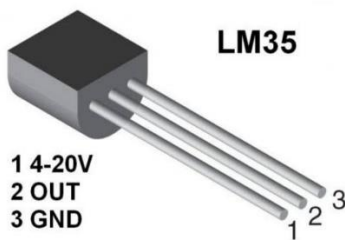
PULSE SENSOR

A Pulse Sensor works by shining green light(550nm) on the finger and measuring the amount of reflected light with the photo sensor. The oxygenated haemoglobin in arterial blood has a property of absorbing green light. As you keep shining light and taking photo sensor readings you quickly begin to obtain a heart beat pulse reading



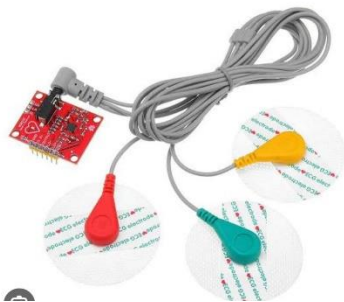
LM35 TEMPERATURE SENSOR

LM35 is a Temperature Sensor which can measure temperature in the range of -55.C to 150.C. It's a 3 terminal device that provides analog voltage proportional to the temperature. The temperature sensor in the Arduino converts the surrounding temperature to voltage. It further converts the voltage to Celsius, Celsius to Fahrenheit, and prints the Fahrenheit temperature on the LCD screen.



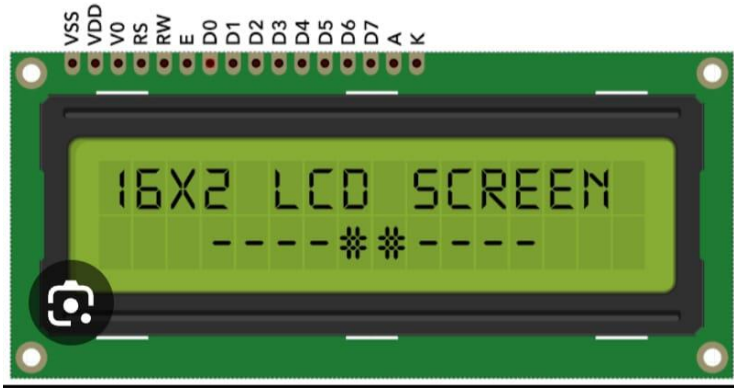
AD8232 HEART RATE MONITOR

This sensor is a cost-effective board used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG and output as an analog reading.



16x2 LCD

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The LCD is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data. The liquid crystal display (LCD) panel is designed to project on-screen information of a microcomputer onto a larger screen with the aid of a standard overhead projector, so that large audiences may view on-screen information without having to crowd around the TV monitor.



CODE:

```
#include <LiquidCrystal.h>

float temp;
int sensor = A2;
float tempc;
float tempf;
const int rs = 12, en = 9, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(12, 9, 5, 4, 3, 2);
int pulsePin = A1;
int blinkPin = 13;
volatile int BPM;
volatile int Signal
volatile int IBI = 600;
volatile boolean Pulse = false;
volatile boolean QS = false;
static boolean serialVisual = true;
volatile int rate[10]
volatile unsigned long sampleCounter = 0;
volatile unsigned long lastBeatTime = 0;
volatile int P = 512;
```

```
volatile int T = 512;  
  
volatile int thresh = 525;  
  
volatile int amp = 100;  
  
volatile boolean firstBeat = true;  
volatile boolean secondBeat = false;
```

```
void setup()  
{  
    Serial.begin(9600);  
pinMode(10, INPUT);  
pinMode(11, INPUT);  
    pinMode(blinkPin,OUTPUT);  
    Serial.begin(115200);  
    interruptSetup();
```

```
    lcd.begin(16, 2);  
    lcd.clear();  
}
```

```
void loop()  
{  
  
    {  
        temp=analogRead(sensor);  
tempc=(temp*4.88)/10;  
        lcd.setCursor(0,0);  
        lcd.println(tempc);  
        delay(4000);
```

```
    }  
    serialOutput();
```

```
    if (QS == true)  
    {
```

```
    serialOutputWhenBeatHappens();
    QS = false;
}

delay(20);
}

void interruptSetup()
{

    TCCR2A = 0x02;
    TCCR2B = 0x06;
    OCR2A = 0X7C;
    TIMSK2 = 0x02;
    sei();
}

void serialOutput()
{
    if (serialVisual == true)
    {
        arduinoSerialMonitorVisual('-', Signal);
    }
    else
    {
        sendDataToSerial('S', Signal);
    }
}

void serialOutputWhenBeatHappens()
{
```

```
if (serialVisual == true)
{
    Serial.print(" Heart-Beat Found ");
    Serial.print("BPM: ");
    Serial.println(BPM);
    lcd.setCursor(1,1);
    lcd.print("BPM: ");
    lcd.setCursor(5,1);
    lcd.print(BPM);
    delay(3000);
    lcd.clear();
}
else
{
    sendDataToSerial('B',BPM);
    sendDataToSerial('Q',IBI);
}
}

void arduinoSerialMonitorVisual(char symbol, int data )
{
    const int sensorMin = 0;
    const int sensorMax = 1024;
    int sensorReading = data;
    int range = map(sensorReading, sensorMin, sensorMax, 0, 11);
}

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

}

ISR(TIMER2_COMPA_vect)

{

cli();

Signal = analogRead(pulsePin);

sampleCounter += 2;

int N = sampleCounter - lastBeatTime;

if(Signal < thresh && N > (IBI/5)*3)

{

if (Signal < T)

{

T = Signal;

}

}

if(Signal > thresh && Signal > P)

{

P = Signal;

}

if (N > 250)

{

if ((Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3))

{

Pulse = true;

digitalWrite(blinkPin,HIGH);

IBI = sampleCounter - lastBeatTime;

lastBeatTime = sampleCounter;

if(secondBeat)

{

,

if secondBeat == TRUE

secondBeat = false;

```
    for(int i=0; i<=9; i++)
    {
        rate[i] = IBI;
    }
}

if(firstBeat)
if firstBeat == TRUE
{
    firstBeat = false;
    secondBeat = true;
    sei();
    return;
}

word runningTotal = 0;

for(int i=0; i<=8; i++)
{
    rate[i] = rate[i+1];
    runningTotal += rate[i];
}

rate[9] = IBI;
runningTotal += rate[9];
runningTotal /= 10;
BPM = 60000/runningTotal;
QS = true;
}
}

if (Signal < thresh && Pulse == true)
{
```



```
digitalWrite(blinkPin,LOW);

Pulse = false;

amp = P - T;

thresh = amp/2 + T;

P = thresh;

T = thresh;

}

if (N > 2500)
{
    thresh = 512;
    P = 512;
    T = 512;
    lastBeatTime = sampleCounter;
    firstBeat = true;
    secondBeat = false;
}

sei();

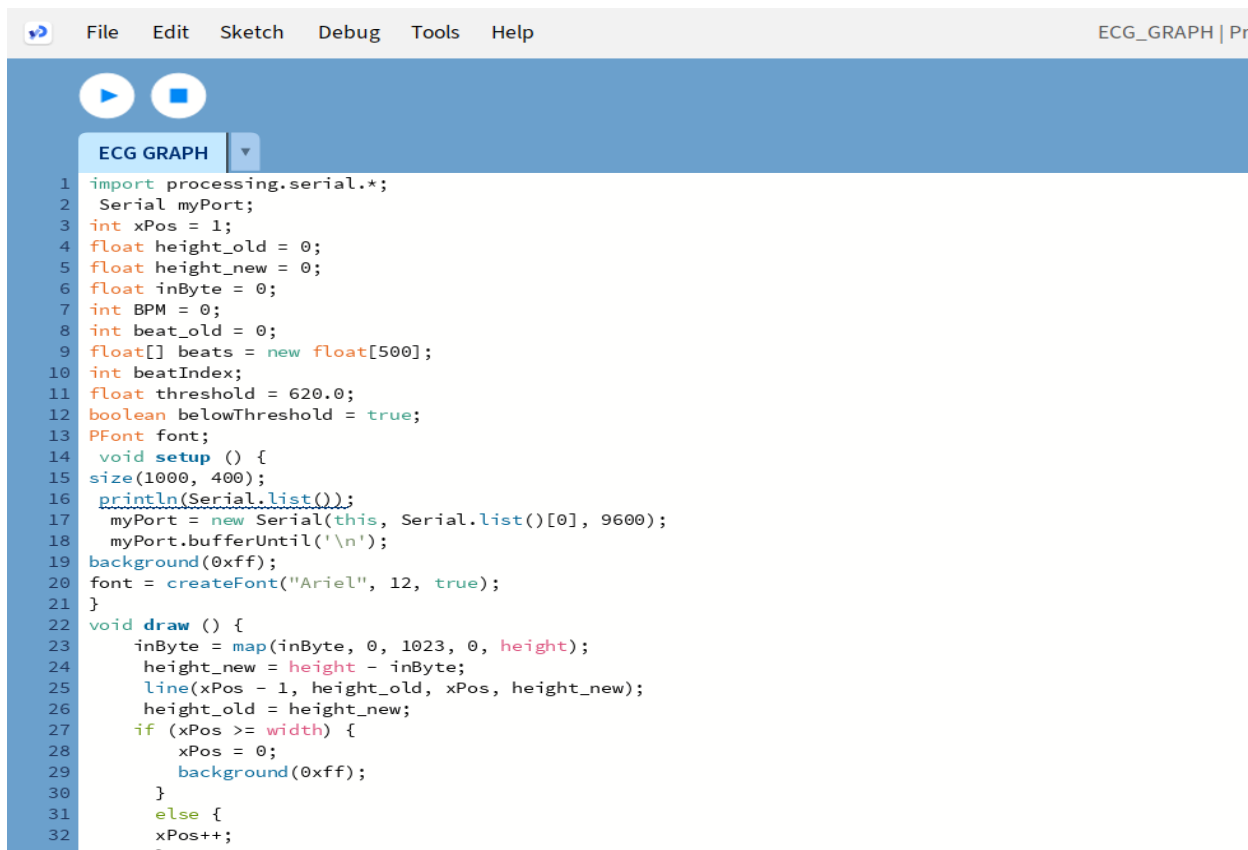
}

void ecg()
{
    if((digitalRead(10) == 1)|| (digitalRead(11) == 1)){
        Serial.println("!");
    }
    Else{
        Serial.println(analogRead(A0));
    }
    delay(1);
}
```

GUI DESIGN:

- The design of the GUI was made using the software Processing IDE .
- We started off the design by setting up a link between the arduino and processing IDE by using commands like Serial my port.
- We declared different variables that could initialize the horizontal position to 1, stores previous and current height values, stores the calculated beats per min value and also to store the received data from serial port.
- The void draw() function runs repeatedly and handles the main logic of the sketch.
- Then the map () function is used to map the received data from arduino in the range 0-1023 to the range 0-height of the sketch window.
- The condition if(xpos>=width) checks if the graph has reached the edge of the screen. If true, the xPos is reset, and the background is cleared.
- The BPM calculation is performed by comparing the received data with the threshold. If the value exceeds the threshold and belowThreshold is true, the calculateBPM() function is called.
- The calculateBPM() function calculates the time between beats, converts it to BPM, stores it in the beats array, and calculates the average BPM.
- BeatIndex is incremented using the modulo operator to cycle through the beats array.

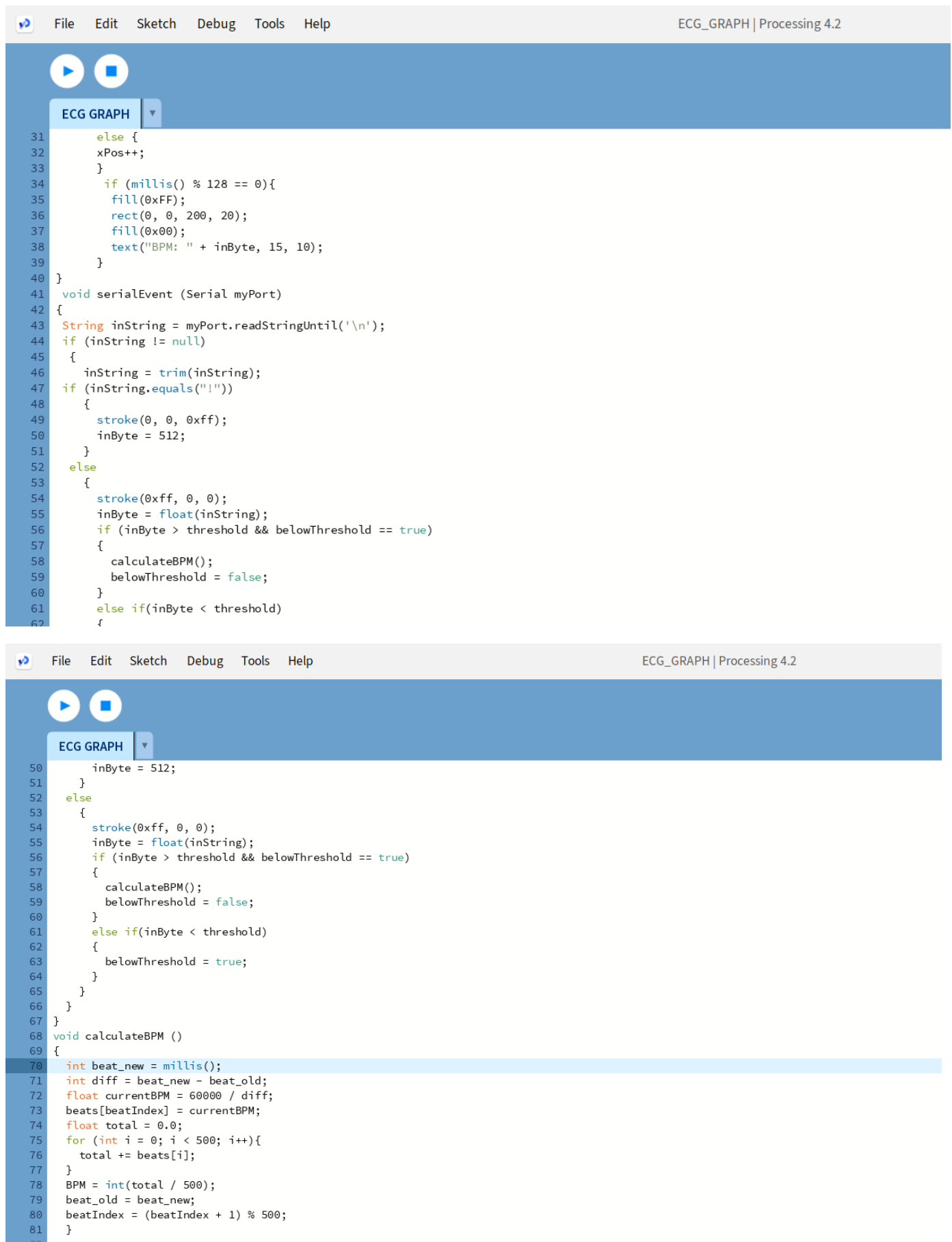
PROCESSING IDE CODE:



```

1  import processing.serial.*;
2  Serial myPort;
3  int xPos = 1;
4  float height_old = 0;
5  float height_new = 0;
6  float inByte = 0;
7  int BPM = 0;
8  int beat_old = 0;
9  float[] beats = new float[500];
10 int beatIndex;
11 float threshold = 620.0;
12 boolean belowThreshold = true;
13 PFont font;
14 void setup () {
15   size(1000, 400);
16   println(Serial.list());
17   myPort = new Serial(this, Serial.list()[0], 9600);
18   myPort.bufferUntil('\n');
19   background(0xff);
20   font = createFont("Ariel", 12, true);
21 }
22 void draw () {
23   inByte = map(inByte, 0, 1023, 0, height);
24   height_new = height - inByte;
25   line(xPos - 1, height_old, xPos, height_new);
26   height_old = height_new;
27   if (xPos >= width) {
28     xPos = 0;
29     background(0xff);
30   }
31   else {
32     xPos++;
33   }

```

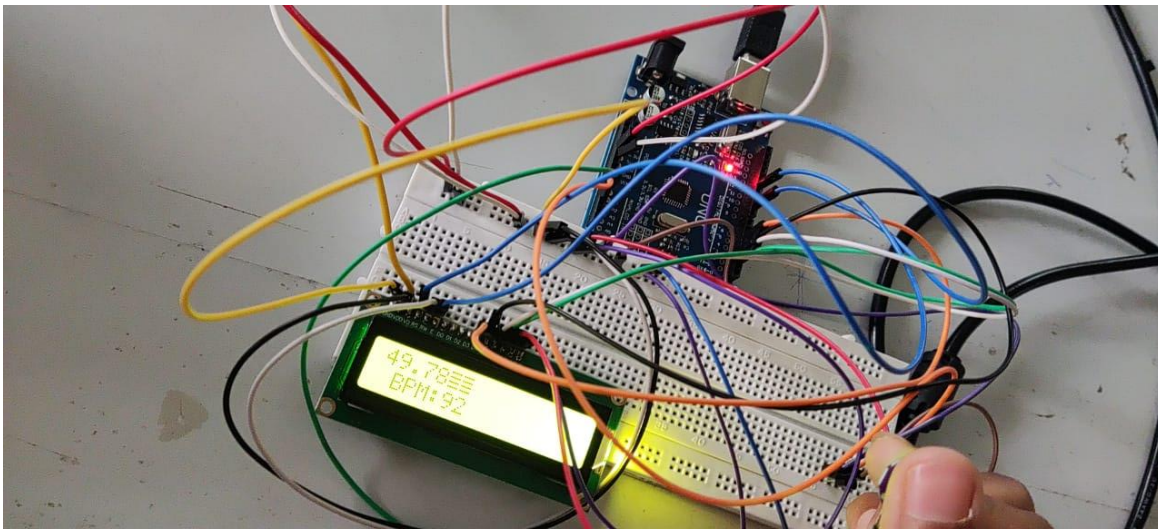


```

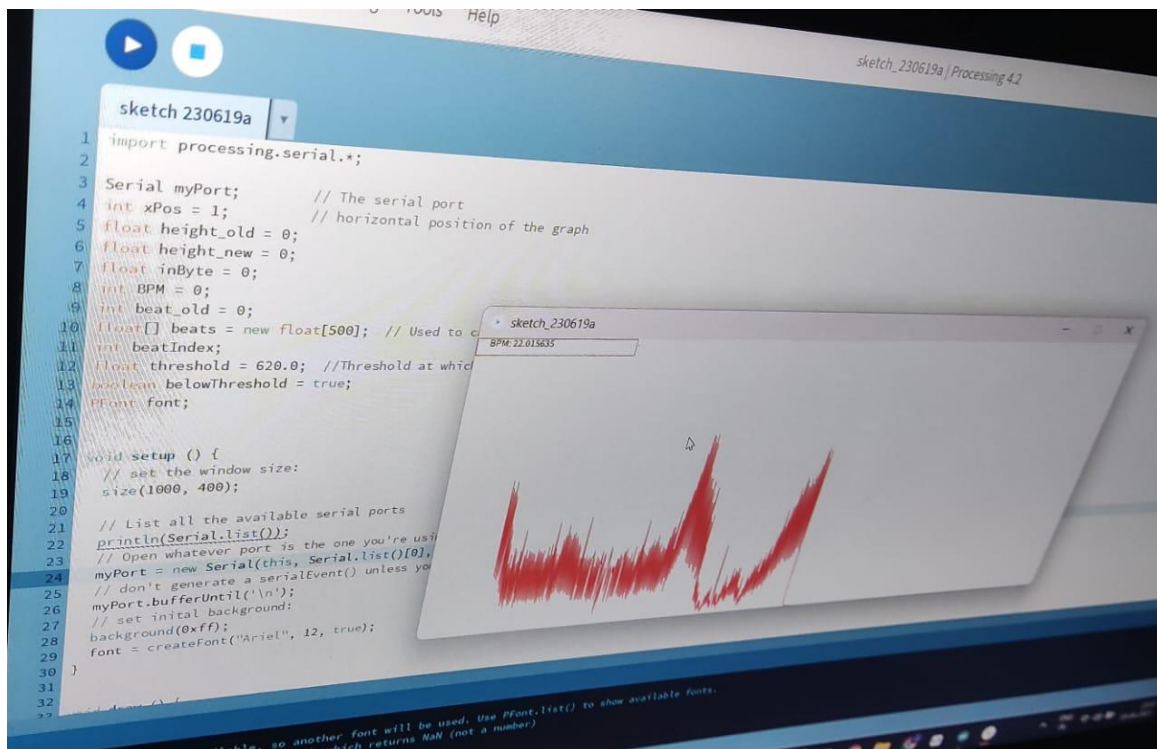
31     else {
32         xPos++;
33     }
34     if (millis() % 128 == 0){
35         fill(0xFF);
36         rect(0, 0, 200, 20);
37         fill(0x00);
38         text("BPM: " + inByte, 15, 10);
39     }
40 }
41 void serialEvent (Serial myPort)
42 {
43     String inString = myPort.readStringUntil('\n');
44     if (inString != null)
45     {
46         inString = trim(inString);
47         if (inString.equals("!"))
48         {
49             stroke(0, 0, 0xFF);
50             inByte = 512;
51         }
52     }
53     else
54     {
55         stroke(0xFF, 0, 0);
56         inByte = float(inString);
57         if (inByte > threshold && belowThreshold == true)
58         {
59             calculateBPM();
60             belowThreshold = false;
61         }
62         else if(inByte < threshold)
63         {
64             belowThreshold = true;
65         }
66     }
67 }
68 void calculateBPM ()
69 {
70     int beat_new = millis();
71     int diff = beat_new - beat_old;
72     float currentBPM = 60000 / diff;
73     beats[beatIndex] = currentBPM;
74     float total = 0.0;
75     for (int i = 0; i < 500; i++){
76         total += beats[i];
77     }
78     BPM = int(total / 500);
79     beat_old = beat_new;
80     beatIndex = (beatIndex + 1) % 500;
81 }

```

OUTPUT:



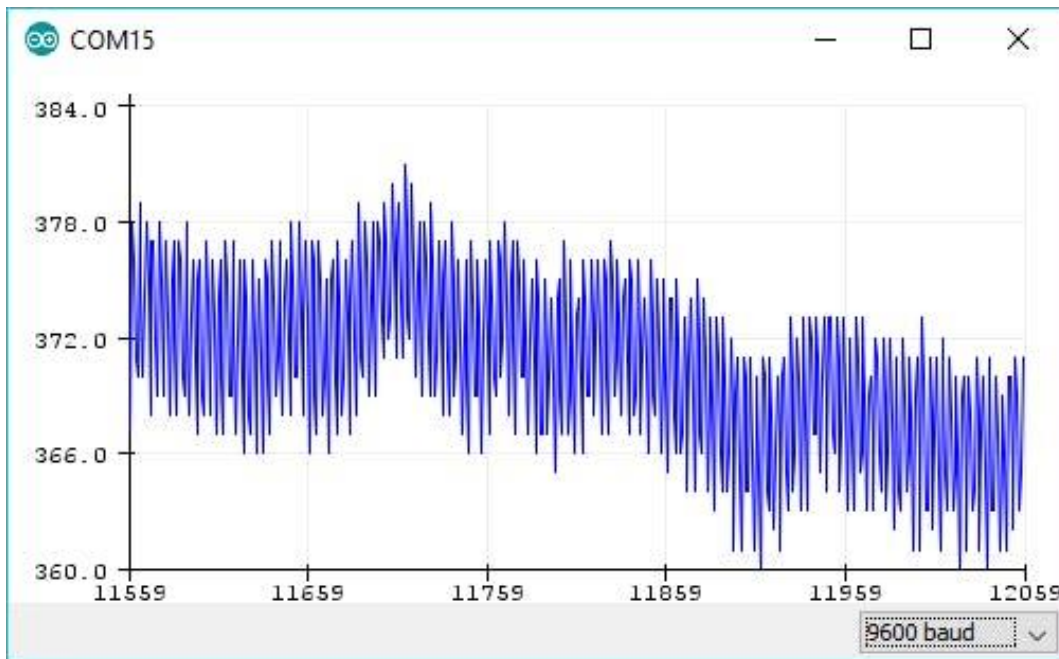
GUI



INFERENCE

We can also perform the same experiment in Arduino by enabling the Serial plotter feature.

The same has been displayed below.



RESULT

The heart rate and the surrounding temperature were measured . It was a combination of two different experiments. The thought behind combining these two ideas is that our surrounding temperature and the heart rate are related to each other. When there is a considerable change in temperature, the heart rate increases. This project can help us monitor the temperature of the patient's surroundings. This could give us an idea as to what could be the reason for the sudden increase of heart rate in the patient. Heart diseases are becoming a big issue for the last few decades and many people die because of certain health problems. Therefore, heart disease cannot be taken lightly. By analyzing or monitoring the ECG signal at the initial stage this disease can be prevented.