Project Report on

HEARTBEAT AND TEMPERATURE

MONITOR WITH APPLICATIONS IN LIE DETECTION

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| Sl. No. | TOPIC | Page No. |
| 1. | ABSTRACT OF THE PROJECT | 3 |
| 2. | CIRCUIT DIAGRAM | 4 |
| 3. | ARDUINO CODE | 5 |
| 4. | PHYSICAL MODEL STRUCTURE | 12 |
| 5. | SCREENSHOTS OF THE OUTPUT | 13 |
| 6. | REFERENCES |  |

**ABSTRACT OF THE PROJECT**

DESCRIPTION :

Our project aims to mimic real life applications for lie detection like the polygraph machine using Arduino. The model comprises of two primary sensors:

1. Heart Rate Pulse Sensor Module
2. DHT11 humidity and temperature sensor

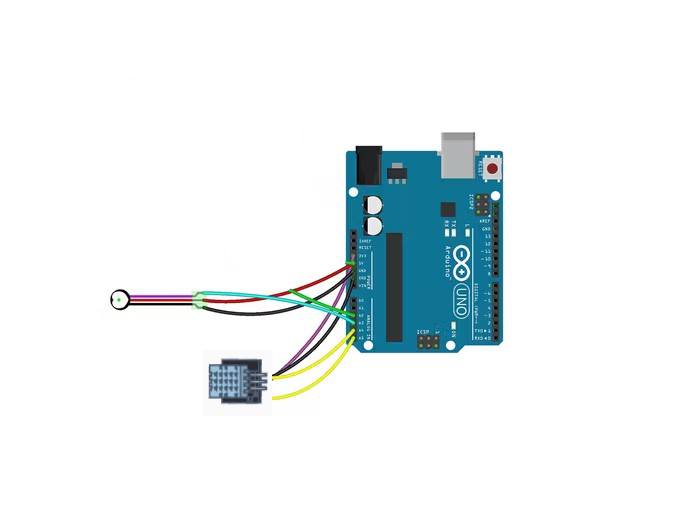
FUNCTIONS OF SENSORS:

* The heart rate pulse sensor module is used to monitor the heart rate and its variations. It records it as the number of beats per minute.
* The DHT11 temperature and humidity sensor measures the temperature and humidity of the surrounding and displays it on the serial monitor

GENERAL WORKING:

* The 2 sensors- the temperature and humidity sensor and the heart pulse sensor modules are secured to the arduino uno board and are analyzed by the arduino IDE.
* The heart rate is detected by the pulse sensor module and is obtained on the serial plotter and the processing development environment which draws 2 dimensional graphics for the heart rate as sketches.
* An experienced examiner can now use this as a lie detector machine to detect the lie with high accuracy.
* The impedance pneumography can also be used with ecg electrodes to measure the breath rate and GSR sensors can be added on for measuring the galvanic skin response through electrical conductance of the skin for a higher accuracy of the polygraph machine

## CIRCUIT DIAGRAM



## ARDUINO & PROCESSING IDE CODE

**CODE FOR THE HEART RATE RUNNING ON THE PROCESSING SOFTWARE:**

import processing.serial.\*;

PFont font;

PFont portsFont;

Scrollbar scaleBar; Serial port;

int Sensor; int IBI;

int BPM; int[] RawY; int[] ScaledY; int[] rate; float zoom; float offset; color eggshell = color(255, 253, 248); int heart = 0; int PulseWindowWidth = 490; int PulseWindowHeight = 512; int BPMWindowWidth = 180; int BPMWindowHeight = 340; boolean beat = false;

String serialPort;

String[] serialPorts = new String[Serial.list().length]; boolean serialPortFound = false;

Radio[] button = new Radio[Serial.list().length\*2]; int numPorts = serialPorts.length; boolean refreshPorts = false;

void setup() {

size(700, 600); frameRate(100); font = loadFont("Arial-BoldMT-24.vlw"); textFont(font); textAlign(CENTER); rectMode(CENTER); ellipseMode(CENTER);

scaleBar = new Scrollbar (400, 575, 180, 12, 0.5, 1.0);

RawY = new int[PulseWindowWidth]; ScaledY = new int[PulseWindowWidth]; rate = new int [BPMWindowWidth]; zoom = 0.75; resetDataTraces(); background(0);

drawDataWindows(); drawHeart();

fill(eggshell); text("Select Your Serial Port",245,30); listAvailablePorts();

}

void draw() { if(serialPortFound){

background(0); noStroke(); drawDataWindows(); drawPulseWaveform(); drawBPMwaveform(); drawHeart();

fill(eggshell);

text("Pulse Sensor Amped Visualizer v1.5",245,30); text("IBI " + IBI + "mS",600,585); text(BPM + " BPM",600,200);

text("Pulse Window Scale " + nf(zoom,1,2), 150, 585);

scaleBar.update (mouseX, mouseY); scaleBar.display();

} else {

autoScanPorts();

if(refreshPorts){

refreshPorts = false;

drawDataWindows(); drawHeart(); listAvailablePorts();

}

for(int i=0; i<numPorts+1; i++){

button[i].overRadio(mouseX,mouseY); button[i].displayRadio();

}

}

}

void drawDataWindows(){

noStroke(); fill(eggshell);

rect(255,height/2,PulseWindowWidth,PulseWindowHeight); rect(600,385,BPMWindowWidth,BPMWindowHeight);

}

void drawPulseWaveform(){

RawY[RawY.length-1] = (1023 - Sensor) - 212; zoom = scaleBar.getPos(); offset = map(zoom,0.5,1,150,0); for (int i = 0; i < RawY.length-1; i++) { RawY[i] = RawY[i+1]; float dummy = RawY[i] \* zoom + offset;

ScaledY[i] = constrain(int(dummy),44,556);

}

stroke(250,0,0); noFill();

beginShape(); for (int x = 1; x < ScaledY.length-1; x++) {

vertex(x+10, ScaledY[x]);

}

endShape();

}

void drawBPMwaveform(){

if (beat == true){

beat = false; for (int i=0; i<rate.length-1; i++){

rate[i] = rate[i+1];

}

BPM = min(BPM,200); float dummy = map(BPM,0,200,555,215); rate[rate.length-1] = int(dummy);

}

stroke(250,0,0); strokeWeight(2);

noFill();

beginShape(); for (int i=0; i < rate.length-1; i++){

vertex(i+510, rate[i]);

}

endShape();

}

void drawHeart(){

fill(250,0,0); stroke(250,0,0);

heart--; heart = max(heart,0); if (heart > 0){

strokeWeight(8);

}

smooth(); bezier(width-100,50, width-20,-20, width,140, width-100,150); bezier(width-100,50, width-190,-20, width-200,140, width-100,150); strokeWeight(1);

}

void listAvailablePorts(){ println(Serial.list()); serialPorts = Serial.list();

fill(0);

textFont(font,16); textAlign(LEFT); int yPos = 0; int xPos = 35; for(int i=serialPorts.length-1; i>=0; i--){

button[i] = new Radio(xPos, 95+(yPos\*20),12,color(180),color(80),color(255),i,button); text(serialPorts[i],xPos+15, 100+(yPos\*20));

yPos++; if(yPos > height-30){

yPos = 0; xPos+=200;

}

}

int p = numPorts;

fill(233,0,0);

button[p] = new Radio(35, 95+(yPos\*20),12,color(180),color(80),color(255),p,button);

text("Refresh Serial Ports List",50, 100+(yPos\*20));

textFont(font); textAlign(CENTER);

}

void autoScanPorts(){

if(Serial.list().length != numPorts){ if(Serial.list().length > numPorts){

println("New Ports Opened!"); int diff = Serial.list().length - numPorts; serialPorts = expand(serialPorts,diff); numPorts = Serial.list().length;

}else if(Serial.list().length < numPorts){ println("Some Ports Closed!"); numPorts = Serial.list().length;

}

refreshPorts = true; return;

}

}

void resetDataTraces(){

for (int i=0; i<rate.length; i++){

rate[i] = 555;

}

for (int i=0; i<RawY.length; i++){ RawY[i] = height/2;

}

}

**CODE FOR THE HUMIDITY AND THE TEMPERATURE SENSOR:**

#include "dht.h" #define dht\_apin A0 dht DHT; void setup(){ Serial.begin(9600); delay(500);

Serial.println("DHT11 Humidity & temperature Sensor\n\n"); delay(1000);

}

void loop(){

DHT.read11(dht\_apin);

Serial.print("Current humidity = ");

Serial.print(DHT.humidity);

Serial.print("% ");

Serial.print("temperature = ");

Serial.print(DHT.temperature); Serial.println("C "); delay(5000);

}

**BPM USING SERIAL MONITOR**

#define USE\_ARDUINO\_INTERRUPTS true #include <PulseSensorPlayground.h>

const int PulseWire = 3; const int LED13 = 13; int Threshold = 550;

PulseSensorPlayground pulseSensor;

void setup() { Serial.begin(9600); if (pulseSensor.begin()) {

Serial.println("PulseSensor object created!");

}

}

void loop() {

int myBPM = pulseSensor.getBeatsPerMinute();

if (pulseSensor.sawStartOfBeat()) {

Serial.println("♥ A HeartBeat Happened ! ");

Serial.print("BPM: ");

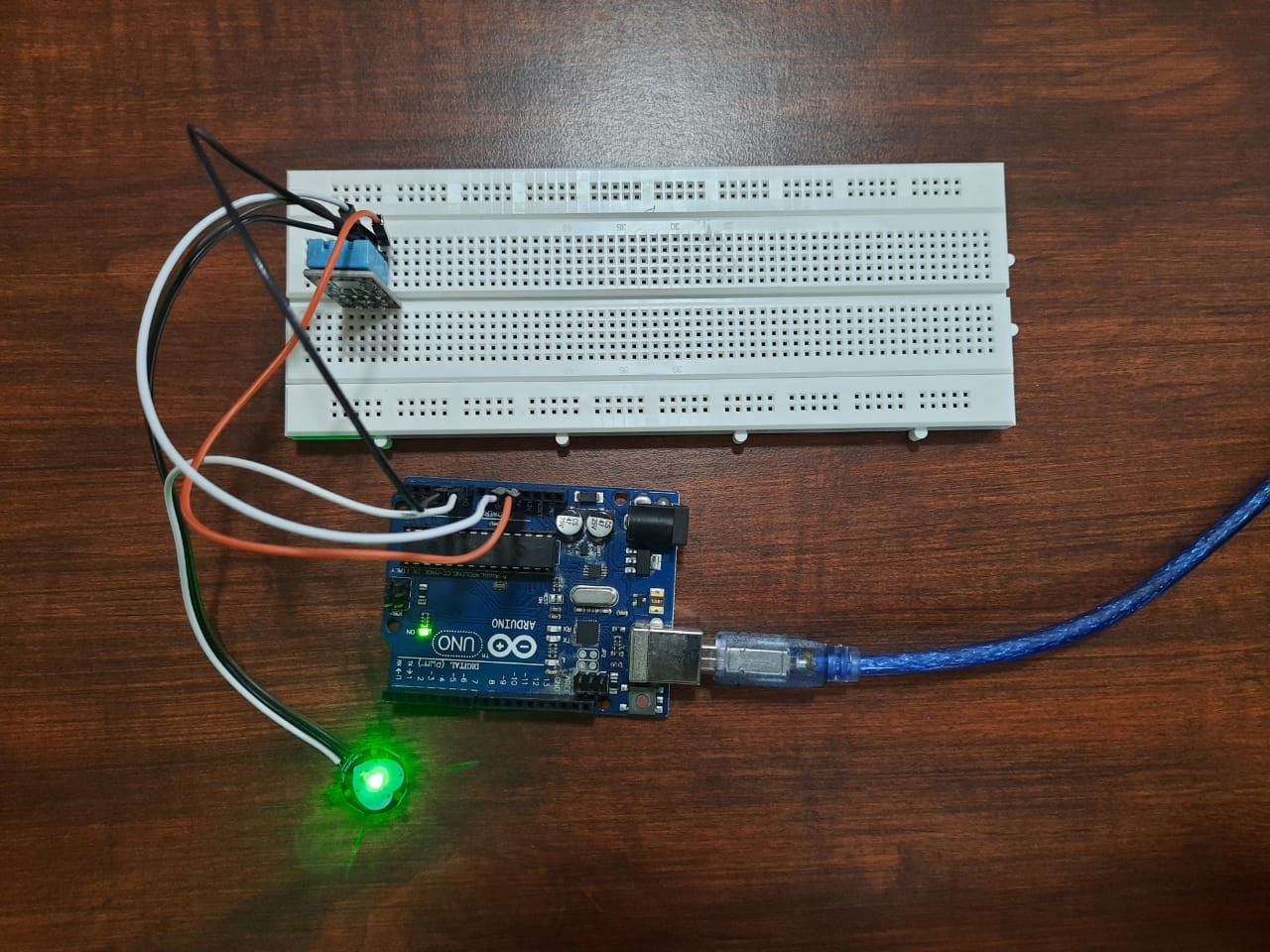
Serial.println(myBPM);

}

delay(20);

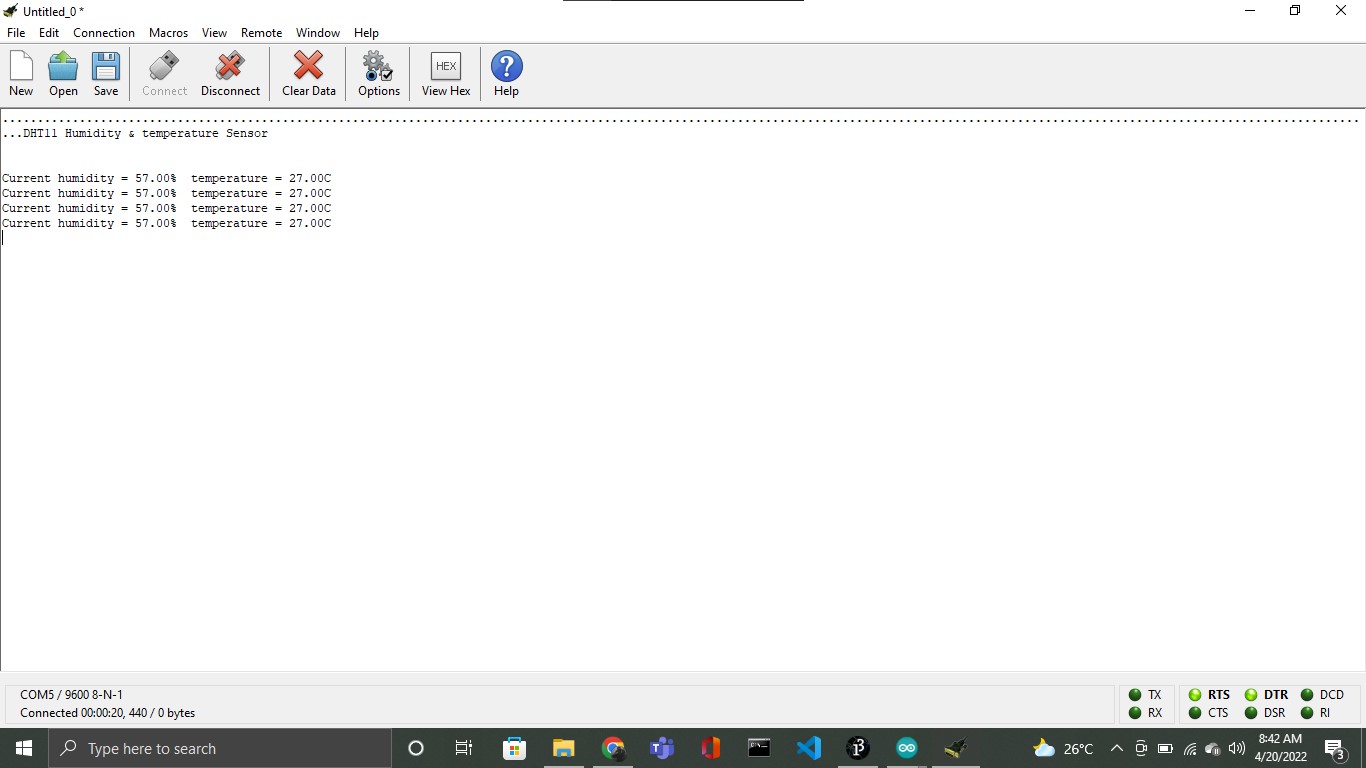
}

## PHYSICAL MODEL STRUCTURE

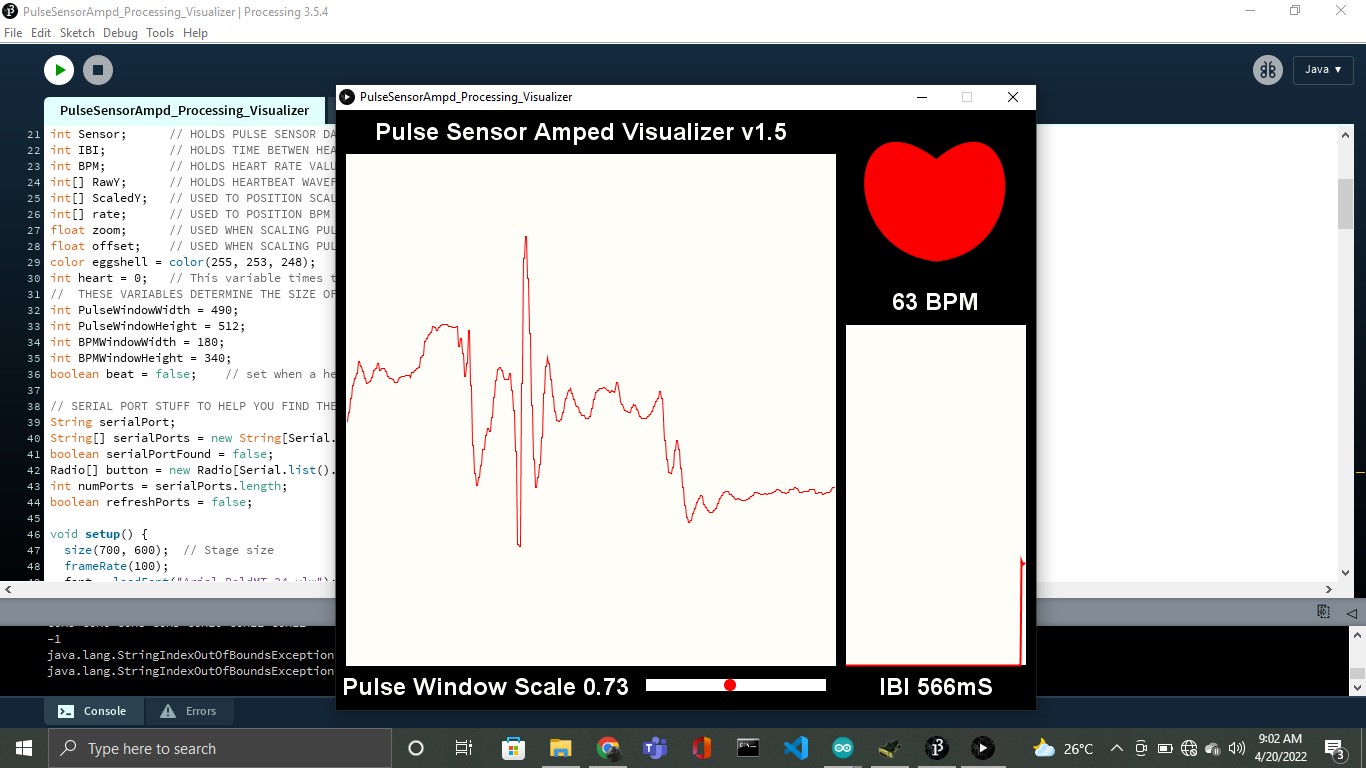


## SCREENSHOTS OF THE OUTPUT

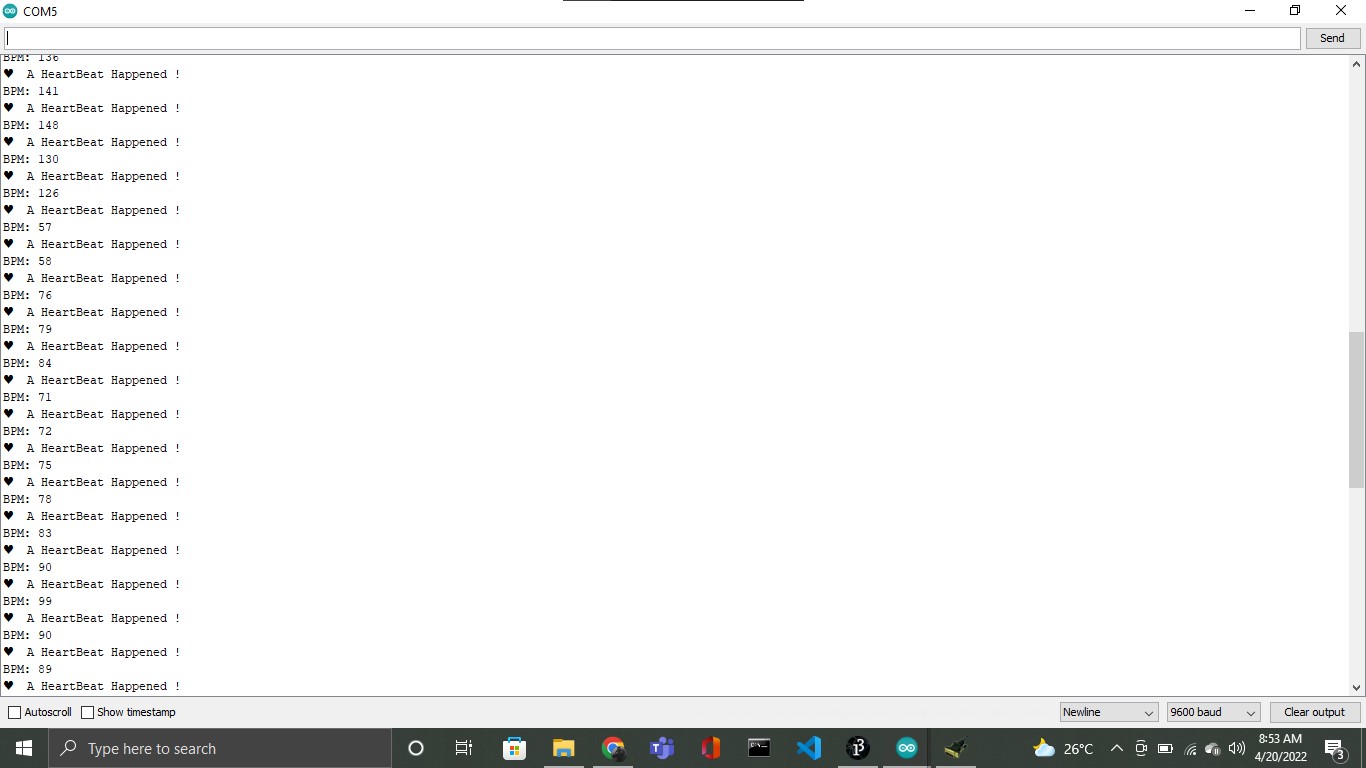
Humidity sensor readings on serial monitor:



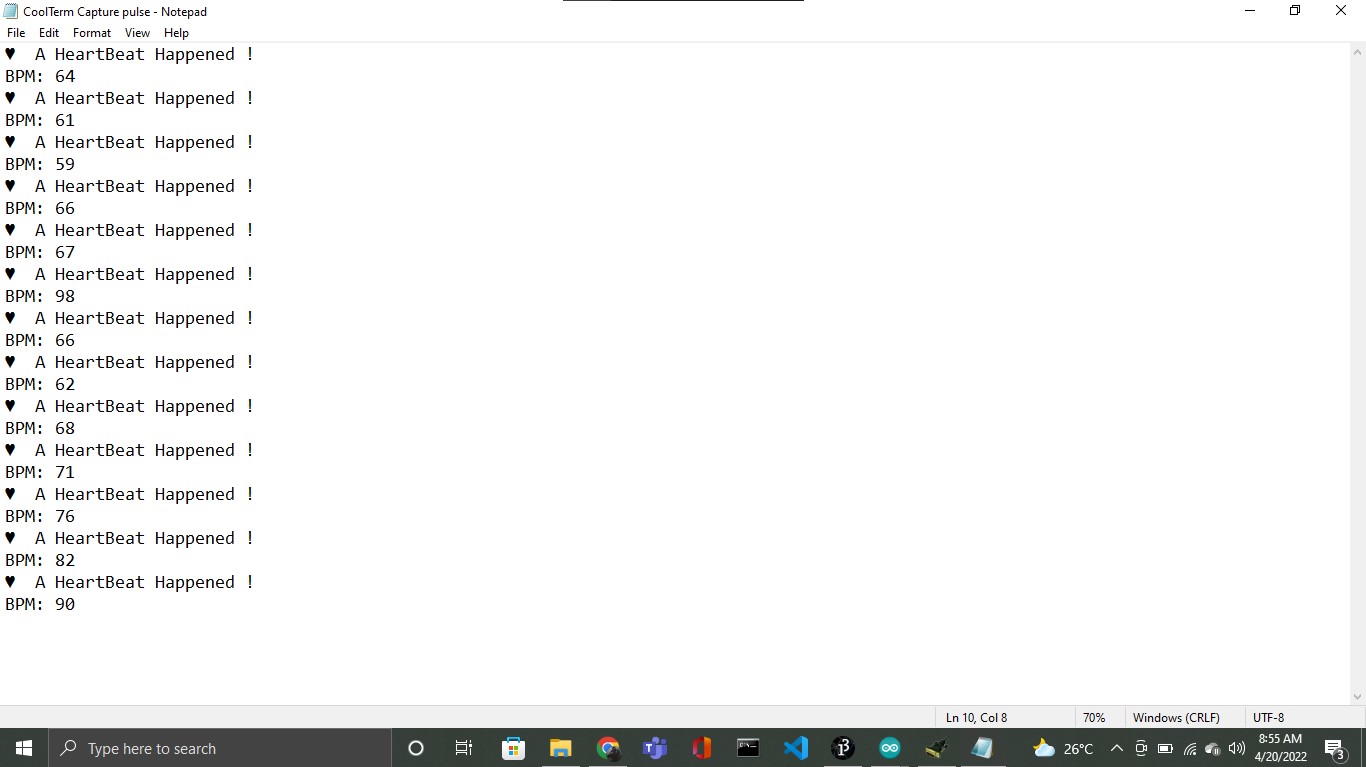
Pulse sensor reading interface and polygraph creation using Processing IDE:



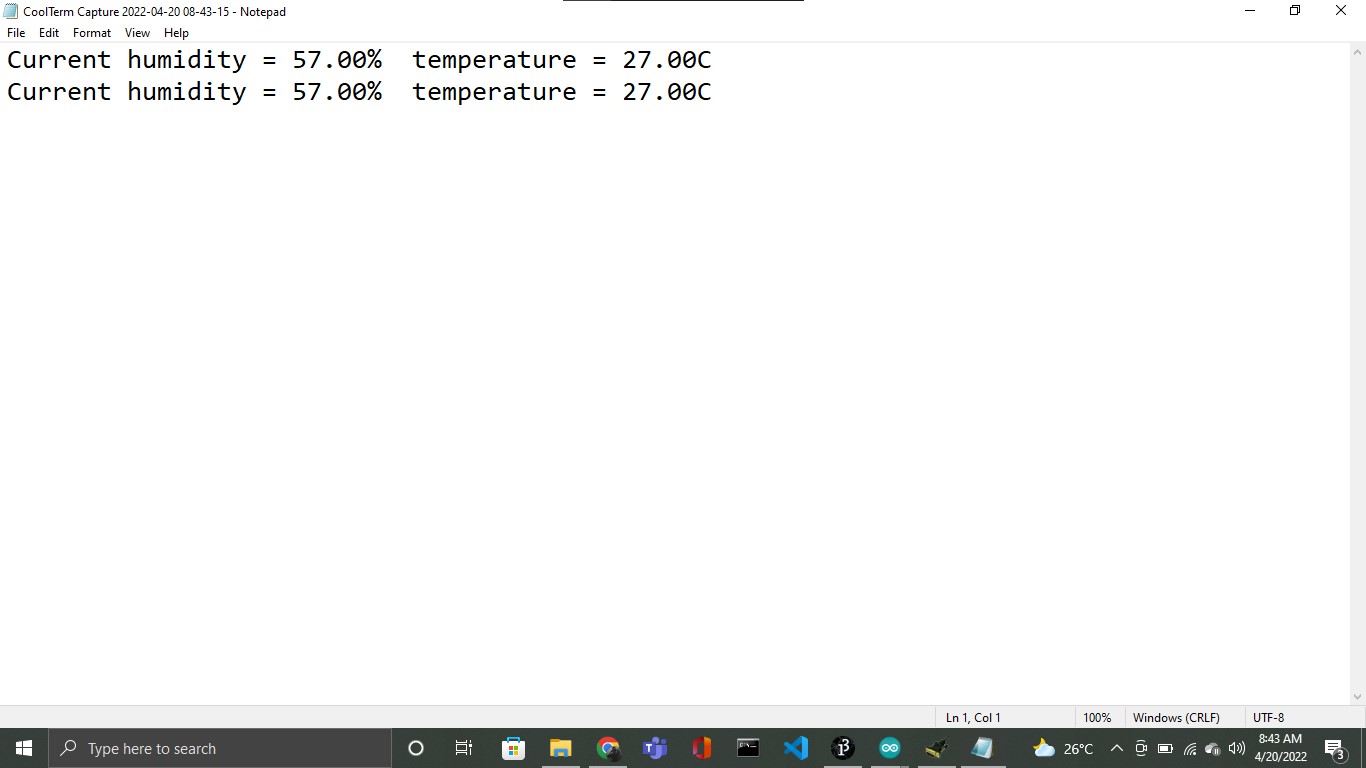
Pulse sensor reading using serial monitor:



Saving the data from pulse sensor into a text file:



Saving the data from humidity sensor into a text file:



## REFERENCES

1. <http://docs.arduino.cc/>
2. <https://processing.org/environment>
3. <https://processing.org/reference/libraries>