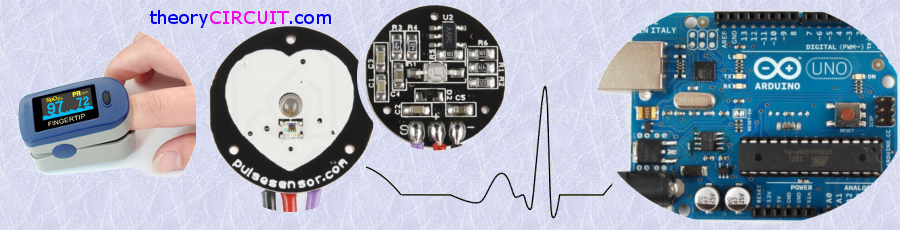
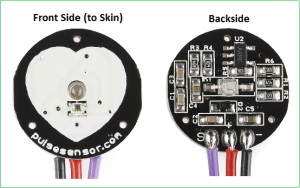
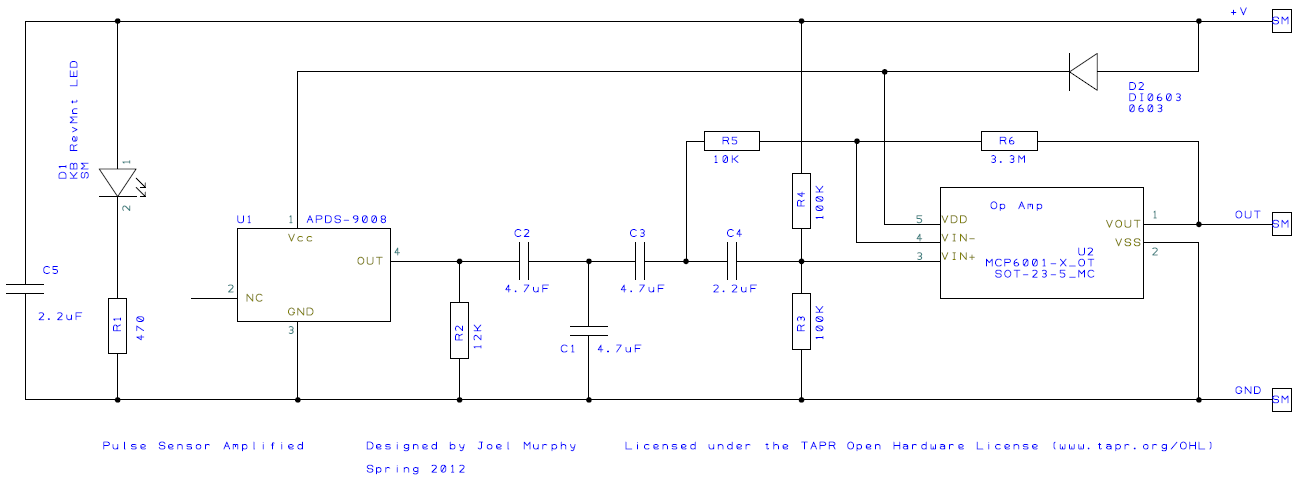
Pulse Sensor with Arduino

[](http://www.theorycircuit.com/wp-content/uploads/2016/08/finger-pulse-sensor-arduino.png)

Heart Rate data can be used in many Electronic design and microcontroller projects. But the heart rate data is difficult to read, however the Pulse Sensor Amped help us to read heart rate. The Pulse Sensor Amped is a plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart-rate data into their projects.It essentially combines a simple optical heart rate sensor with amplification and noise cancellation circuitry making it fast and easy to get reliable pulse readings.

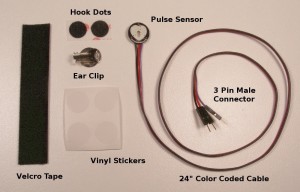
[](http://www.theorycircuit.com/wp-content/uploads/2016/08/pulse-sensor-image.png)

Pulse Sensor Schematic

[](http://www.theorycircuit.com/wp-content/uploads/2016/08/pulse-sensor-schematics.png)

Here the APDS-9008 Miniature surface Mount Ambient light photo sensor from Avago technologies plays an important role and it is responsible to detect concentration of Light bounces back through fingertip or earlobe. D1 LED gives light source for this operation. Signal from the ambient light sensor amplified through op Amp and the signal is ready to read by the microcontroller.

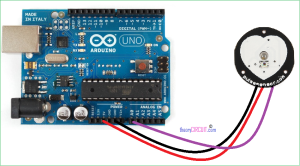
The pulse Sensor Kit

[](http://www.theorycircuit.com/wp-content/uploads/2016/08/pulse-sensor-kit.jpg)

The pulse sensor amped kit Available in

* [Sparkfun.com](https://www.sparkfun.com/products/11574)
* [Pulsesensor.com](http://pulsesensor.com/products/pulse-sensor-amped)

Pulse Sensor Arduino Hookup

[](http://www.theorycircuit.com/wp-content/uploads/2016/08/arduino-pulse-sensor.png)

Connect the sensor’s power supply pins to the arduino board supply pin as Red – 5V, Black – GND and Purple – A0 (analog input 0) its over. This Analog input reading can be displayed in serial terminal of Arduino IDE or it can be drawn as pulse by using Processing IDE.

Pulse Sensor Arduino Code

/\*>> Pulse Sensor Amped 1.2

<<This code is for Pulse Sensor Amped by Joel Murphy and YuryGitmanwww.pulsesensor.com

Code Version 1.2 by Joel Murphy &YuryGitman  Spring 2013

This update fixes the firstBeat and secondBeat flag usage so that realistic BPM is reported.\*/

//  VARIABLES

intpulsePin=0//Pulse Sensor purple wire connected to analog pin 0

intblinkPin=13;// pin to blink led at each beat

intfadePin=5;// pin to do fancy classy fading blink at each beat

intfadeRate=0;// used to fade LED on with PWM on fadePin

// these variables are volatile because they are used during the interrupt service routine!

volatileintBPM;// used to hold the pulse rate

volatileintSignal;// holds the incoming raw data

volatileintIBI=600;//holds the time between beats,must be seeded!

volatilebooleanPulse=false;// true when pulse wave is high, false when it's low

volatilebooleanQS=false;//becomes true when Arduoino finds a beat

voidsetup(){

pinMode(blinkPin,OUTPUT);//pin that will blink to your heartbeat!

pinMode(fadePin,OUTPUT);//pin that will fade to your heartbeat!

**Serial**.begin(115200); // we agree to talk fast!

interruptSetup();//sets up to read Pulse Sensor signal every 2mS

  // UN-COMMENT THE NEXT LINE IF YOU ARE POWERING The Pulse Sensor AT LOW VOLTAGE,

  // AND APPLY THAT VOLTAGE TO THE A-REF PIN

  //analogReference(EXTERNAL);

}

voidloop(){

 sendDataToProcessing('S',Signal);// send Processing the raw Pulse Sensor data

 if(QS==true){// Quantified Self flag is true when arduino finds a heartbeat

       fadeRate=255; // Set 'fadeRate' Variable to 255 to fade LED with pulse

       sendDataToProcessing('B',BPM);   // send heart rate with a 'B' prefix

       sendDataToProcessing('Q',IBI);   // send time between beats with a 'Q' prefix

       QS=false;                      // reset the Quantified Self flag for next time

    }

 ledFadeToBeat();

   delay(20); //  take a break

}

voidledFadeToBeat(){

   fadeRate-=15; //  set LED fade value

   fadeRate=constrain(fadeRate,0,255);   //  keep LED fade value from going into negative numbers!

   analogWrite(fadePin,fadeRate); //  fade LED

 }

voidsendDataToProcessing(charsymbol,intdata){

**Serial**.print(symbol);                // symbol prefix tells Processing what type of data is coming

**Serial**.println(data);                // the data to send culminating in a carriage return

 }

Pulse Sensor Processing Code

/\*

THIS PROGRAM WORKS WITH PulseSensorAmped\_Arduino-xx ARDUINO CODE

THE PULSE DATA WINDOW IS SCALEABLE WITH SCROLLBAR AT BOTTOM OF SCREEN

PRESS 'S' OR 's' KEY TO SAVE A PICTURE OF THE SCREEN IN SKETCH FOLDER (.jpg)

MADE BY JOEL MURPHY AUGUST, 2012

\*/

importprocessing.serial.\*;

PFont font;

Scrollbar scaleBar;

Serial port;

int Sensor; // HOLDS PULSE SENSOR DATA FROM ARDUINO

int IBI; // HOLDS TIME BETWEN HEARTBEATS FROM ARDUINO

int BPM; // HOLDS HEART RATE VALUE FROM ARDUINO

int[] RawY; // HOLDS HEARTBEAT WAVEFORM DATA BEFORE SCALING

int[] ScaledY; // USED TO POSITION SCALED HEARTBEAT WAVEFORM

int[] rate; // USED TO POSITION BPM DATA WAVEFORM

float zoom; // USED WHEN SCALING PULSE WAVEFORM TO PULSE WINDOW

float offset; // USED WHEN SCALING PULSE WAVEFORM TO PULSE WINDOW

color eggshell = color(255, 253, 248);

int heart = 0; // This variable times the heart image 'pulse' on screen

//  THESE VARIABLES DETERMINE THE SIZE OF THE DATA WINDOWS

intPulseWindowWidth = 490;

intPulseWindowHeight = 512;

intBPMWindowWidth = 180;

intBPMWindowHeight = 340;

boolean beat = false; // set when a heart beat is detected, then cleared when the BPM graph is advanced

void**setup**() {

  size(700, 600); // Stage size

  frameRate(100);

  font = loadFont("Arial-BoldMT-24.vlw");

  textFont(font);

  textAlign(CENTER);

  rectMode(CENTER);

  ellipseMode(CENTER);

// Scrollbar constructor inputs: x,y,width,height,minVal,maxVal

  scaleBar = new Scrollbar (400, 575, 180, 12, 0.5, 1.0); // set parameters for the scale bar

  RawY = newint[PulseWindowWidth]; // initialize raw pulse waveform array

  ScaledY = newint[PulseWindowWidth]; // initialize scaled pulse waveform array

  rate = newint [BPMWindowWidth];// initialize BPM waveform array

  zoom = 0.75; // initialize scale of heartbeat window

// set the visualizer lines to 0

 for (inti=0; i<rate.length; i++){

    rate[i] = 555;// Place BPM graph line at bottom of BPM Window

   }

 for (inti=0; i<RawY.length; i++){

    RawY[i] = height/2; // initialize the pulse window data line to V/2

 }

// GO FIND THE ARDUINO

  println(Serial.list());//print a list of available serial ports

  // choose the number between the [] that is connected to the Arduino

  port = new Serial(this, Serial.list()[0], 115200); // make sure Arduino is talking serial at this baud rate

  port.clear();// flush buffer

  port.bufferUntil('\n'); // set buffer full flag on receipt of carriage return

}

void**draw**() {

  background(0);

  noStroke();

// DRAW OUT THE PULSE WINDOW AND BPM WINDOW RECTANGLES

  fill(eggshell); // color for the window background

  rect(255,height/2,PulseWindowWidth,PulseWindowHeight);

  rect(600,385,BPMWindowWidth,BPMWindowHeight);

// DRAW THE PULSE WAVEFORM

  // prepare pulse data points

  RawY[RawY.length-1] = (1023 - Sensor) - 212; // place the new raw datapoint at the end of the array

  zoom = scaleBar.getPos();                      // get current waveform scale value

  offset = map(zoom,0.5,1,150,0); // calculate the offset needed at this scale

  for (inti = 0; i< RawY.length-1; i++) { // move the pulse waveform by

    RawY[i] = RawY[i+1];                         // shifting all raw datapoints one pixel left

    float dummy = RawY[i] \* zoom + offset; // adjust the raw data to the selected scale

    ScaledY[i] = constrain(int(dummy),44,556); // transfer the raw data array to the scaled array

  }

  stroke(250,0,0); // red is a good color for the pulse waveform

  noFill();

  beginShape(); // using beginShape() renders fast

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

    vertex(x+10, ScaledY[x]); //draw a line connecting the data points

  }

  endShape();

// DRAW THE BPM WAVE FORM

// first, shift the BPM waveform over to fit then next data point only when a beat is found

 if (beat == true){ // move the heart rate line over one pixel every time the heart beats

   beat = false; // clear beat flag (beat flag waset in serialEvent tab)

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

     rate[i] = rate[i+1];                  // shift the bpm Y coordinates over one pixel to the left

   }

// then limit and scale the BPM value

   BPM = min(BPM,200); // limit the highest BPM value to 200

   float dummy = map(BPM,0,200,555,215); // map it to the heart rate window Y

   rate[rate.length-1] = int(dummy); // set the rightmost pixel to the new data point value

 }

 // GRAPH THE HEART RATE WAVEFORM

 stroke(250,0,0); // color of heart rate graph

 strokeWeight(2); // thicker line is easier to read

 noFill();

 beginShape();

 for (inti=0; i< rate.length-1; i++){ // variable 'i' will take the place of pixel x position

   vertex(i+510, rate[i]); // display history of heart rate datapoints

 }

 endShape();

 // DRAW THE HEART AND MAYBE MAKE IT BEAT

  fill(250,0,0);

  stroke(250,0,0);

  // the 'heart' variable is set in serialEvent when arduino sees a beat happen

  heart--;                    // heart is used to time how long the heart graphic swells when your heart beats

  heart = max(heart,0); // don't let the heart variable go into negative numbers

  if (heart > 0){ // if a beat happened recently,

    strokeWeight(8); // make the heart big

 }

  smooth(); // draw the heart with two bezier curves

  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); // reset the strokeWeight for next time

// PRINT THE DATA AND VARIABLE VALUES

  fill(eggshell); // get ready to print text

  text("Pulse Sensor Amped Visualizer 1.1",245,30); // tell them what you are

  text("IBI " + IBI + "mS",600,585); // print the time between heartbeats in mS

  text(BPM + " BPM",600,200); // print the Beats Per Minute

  text("Pulse Window Scale " + nf(zoom,1,2), 150, 585); // show the current scale of Pulse Window

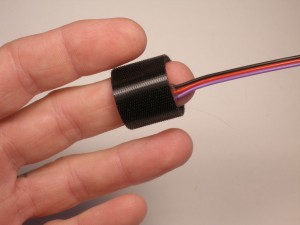
//  DO THE SCROLLBAR THINGS

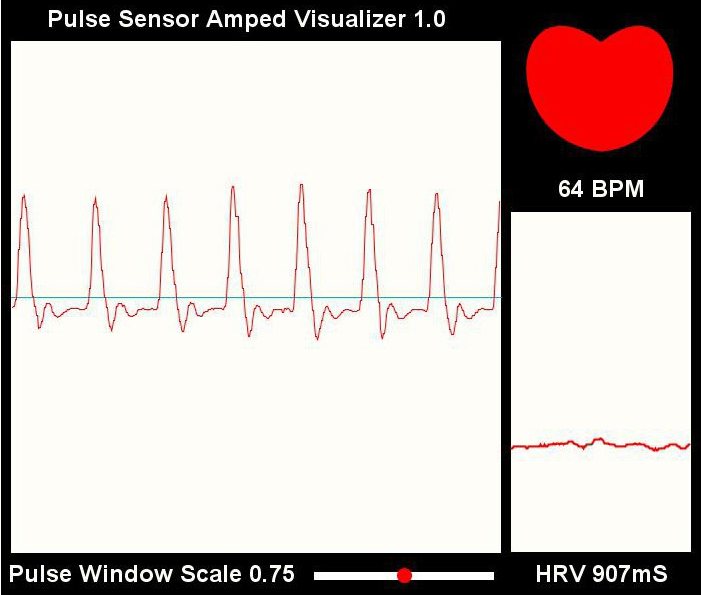
  scaleBar.update (mouseX, mouseY);

  scaleBar.display();

  //

 }  //end of draw loop

[](http://www.theorycircuit.com/wp-content/uploads/2016/08/pulse-sensor-place.jpg)

[](http://www.theorycircuit.com/wp-content/uploads/2016/08/pulse-screenshot.png)

La configuration pour la port serial

voidsetup(){

pinMode(blinkPin,OUTPUT);

pinMode(fadePin,OUTPUT);

**Serial**.begin(115200);

interruptSetup();

}

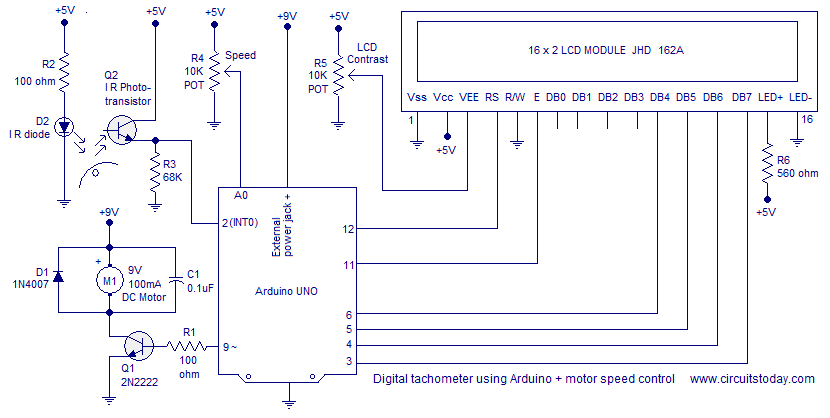
**Tachometerusingarduino**

Last Updated on [January 13, 2015](http://www.circuitstoday.com/tachometer-using-arduino) by [praveen](http://www.circuitstoday.com/author/praveen) in [Arduino](http://www.circuitstoday.com/category/arduino) with [10 Comments](http://www.circuitstoday.com/tachometer-using-arduino#comments)

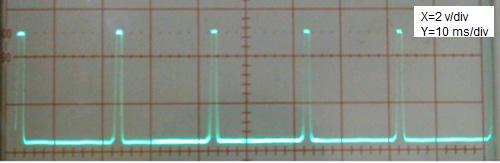
**Digital tachometer using arduino plus speed control.**

Tachometer is a device used for measuring the number of revolutions of an object in a given interval of time. Usually it is expressed in revolutions per minute or RPM. Earlier tachometers purely mechanical where the revolution is transferred to the tachometer through mechanical coupling (cable or shaft) , the rpm is determined using a gear mechanism and it is displayed on a dial. With the advent of modern electronics, the tachometers have changed a lot. This article is about a contactless digital tachometer using arduino. The speed of the motor can be also controlled using the same circuit. The RPM and all the other informations are displayed on a 16×2 LCD screen. The circuit diagram of the digital tachometer using arduino is shown below.

**Circuit diagram.**

**[](http://www.circuitstoday.com/wp-content/uploads/2014/07/tachometer-using-arduino.png)RPM Sensor.**

 An IR photo transistor and IR LED forms the sensor. IR photo transistor is a type of photo transistor which responds to infra-red waves only. The use of IR phototransistor avoids other light interferences from the environment. The photo transistor and IR diode are aligned side by side. Resistor R2 limits the current through the IR diode. A reflective strip is glued on the rotating object (shaft, disc or fan) in line with the sensor. I used a 9V/100mA cooling fan. The clearence between the sensor and reflective strip has to be less than 1cm. When the reflective strip passes in front of the sensor, IR waves are reflected back to the photo transistor. The photo transistor conducts more at this moment and as a result the voltage across R3(68K resistor) shoots up at this moment. The result will be a waveform like what shown below at the emitter of the photo transistor. RPM can be determined by counting the number of upward shoots in a given interval of time.

**[](http://www.circuitstoday.com/wp-content/uploads/2014/07/tachometer-waveform.jpg)**

**Counting the RPM.**

Arduino is used for counting the RPM and displaying it on the LCD screen. Emitter of the photo transistor is connected to the Interrupt 0 (digital pin 2) of the arduino. The arduino interrupt is configured to be rising edge triggered. As a result the will be an interrupt for every upward shoot in the emitter waveform. The number of interrupts occurred in a given time is counted by incrementing a varible using the interrupt service routine. The time elapsed during te counting cycle is determined using the millis() function. The millis() function returns the number of milli seconds passed since the arduino board is switched ON. Calling the millis() function before and after the counting cycle and the taking their difference gives the times passed during the counting cycle. The (number of interrupts/time in milliseconds)\*60000 will give the revolutions per minute (RPM).

**Controlling the speed of motor.**

A provision for controlling the motor speed using a potentiometer is also included in the circuit. Transistor Q1 is used for driving the motor. Its base is connected to pwm pin 9 of the arduino through the current limiting resistor R1. Wiper of the speed control POT R4 is connected to anlog pin A0 of the arduino. The voltage at this pin is converted into a value between 0 and 1023 using the anlogRead function. Then this value is divided by four to fit it into the 0 to 255 range. Then this value is written to the PWM pin 9 using the anlogWrite function. The result will be a square wave at pin 9 whose duty cycle is proportional to the value written using the analogWrite function. For example if the value is 255, the duty cycle will be 100% and if the value is 127, the duty cycle will be around 50%. D1 is  a free wheeling diode and C1 is a noise by-pass capacitor(de coupler).  The rpm and duty cycle are displayed on the LCD screen using the standard LiquidCrystal library. Read this article: [Interfacing LCD to Arduino](http://www.circuitstoday.com/interfacing-lcd-to-arduino). Full program for the digital tachometer using arduino is shown below.

**Program.**

#include<LiquidCrystal.h>

LiquidCrystal lcd(12,11,6,5,4,3);

int pwm=9;

int pot=A0;

float value=0;

int percent;

float rev=0;

int rpm;

int oldtime=0;

int time;

void isr() //interrupt service routine

{

rev++;

}

void setup()

{

lcd.begin(16,2); //initialize LCD

attachInterrupt(0,isr,RISING); //attaching the interrupt

}

void loop()

{

delay(1000);

detachInterrupt(0); //detaches the interrupt

time=millis()-oldtime; //finds the time

rpm=(rev/time)\*60000; //calculates rpm

oldtime=millis(); //saves the current time

rev=0;

value=analogRead(pot); //reads the speed control POT

value=value/4;

analogWrite(pwm,value); //sets the desired speed

percent=(value/255)\*100; //finds the duty cycle %

lcd.clear();

lcd.setCursor(0,0);

lcd.print("\_\_\_TACHOMETER\_\_\_");

lcd.setCursor(0,1);

lcd.print(rpm);

lcd.print(" RPM");

lcd.print("   ");

lcd.print(percent);

lcd.print("%");

attachInterrupt(0,isr,RISING);

}

**Notes.**

* The arduino board can be powered using a 9V supply through the external power jack.
* The 5V needed at some parts of the circuit can be tapped from the 5V source on the arduino board.
* The fan I used was rated 9V/100mA. The transistor 2N2222 can handle only upto 800mA. Keepthis in mindwhileselecting the load.
* The LCD module used was JHD162A.
* POT R5 can be used to adjust the contrast of the LCD display. When connected first, the LCD may not show up anything. Adjust the R5 until you get the display. The optimum voltage at the wiper of R5 is between 0.4 to 1V.
* The IR photo transistor and the IR diode both were taken from an LTH-1550 photo interrupter module.
* The lateral surface of the photo transistor must be masked using a tape.
* The sensor arrangement is shown in the figure below.

<http://www.circuitstoday.com/tachometer-using-arduino>

Pour detector la frequences cardiaques et affichee sur LCD 2\*16

code :

#include<LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int in = 8;

int Reset=6;

int start=7;

int count=0,i=0,k=0,rate=0;

unsigned long time2,time1;

unsigned long time;

byte heart[8] =

{

0b00000,

0b01010,

0b11111,

0b11111,

0b11111,

0b01110,

0b00100,

0b00000

};

void setup()

{

lcd.createChar(1, heart);

lcd.begin(16,2);

lcd.print("Heart Beat ");

lcd.write(1);

lcd.setCursor(0,1);

lcd.print("Monitering");

pinMode(in, INPUT);

pinMode(Reset, INPUT);

pinMode(start, INPUT);

digitalWrite(Reset, HIGH);

digitalWrite(start, HIGH);

delay(1000);

}

void loop()

{

if(!(digitalRead(start)))

{

k=0;

lcd.clear();

lcd.print("Please wait.......");

while(k<5)

{

if(digitalRead(in))

{

if(k==0)

time1=millis();

k++;

while(digitalRead(in));

}

}

time2=millis();

rate=time2-time1;

rate=rate/5;

rate=60000/rate;

lcd.clear();

lcd.print("Heart Beat Rate:");

lcd.setCursor(0,1);

lcd.print(rate);

lcd.print(" ");

lcd.write(1);

k=0;

rate=0;

}

if(!digitalRead(Reset))

{

rate=0;

lcd.clear();

lcd.print("Heart Beat Rate:");

lcd.setCursor(0,1);

lcd.write(1);

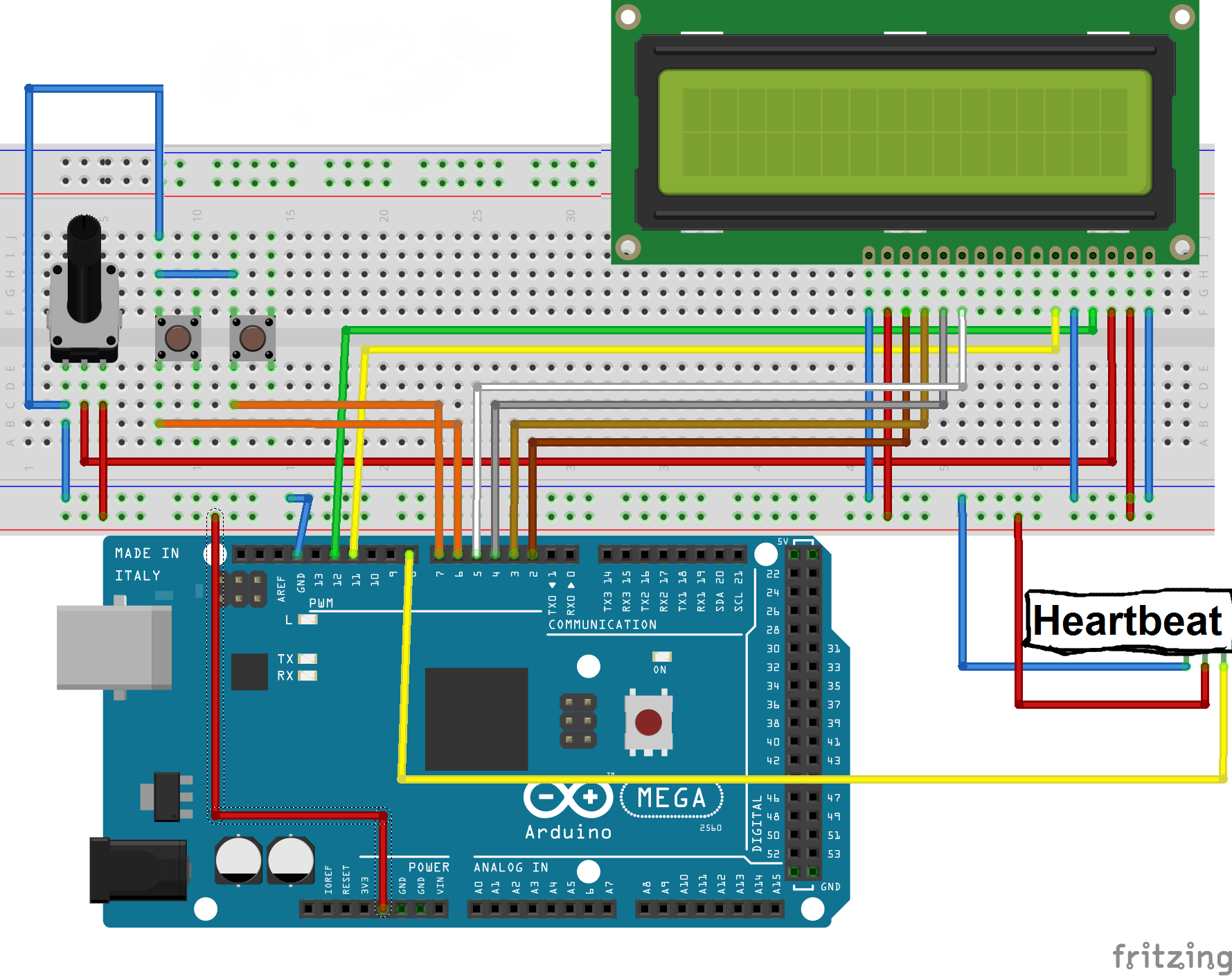
lcd.print(rate);

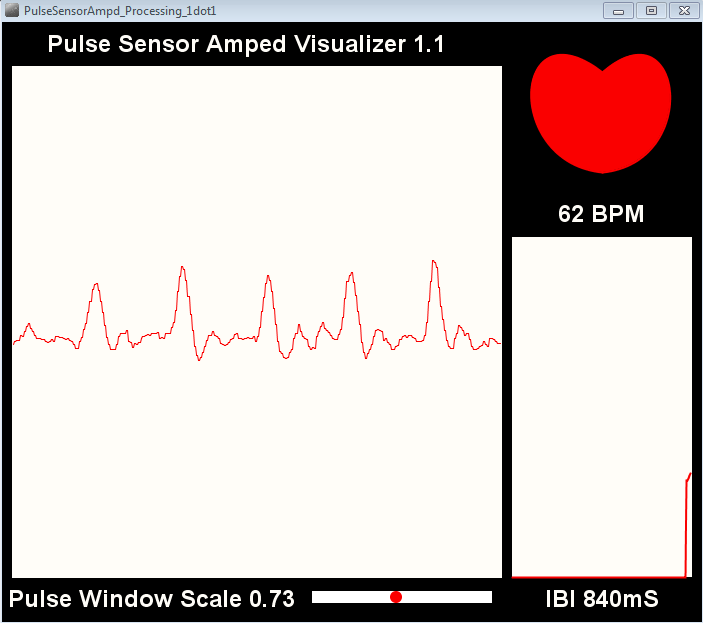
k=0;

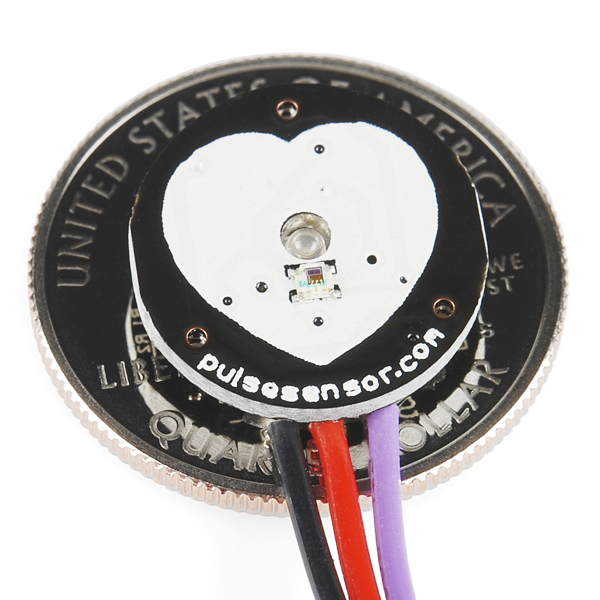
}

}

Le shema pour cette code :







Ministère de l'Enseignement supérieur et de la Recherche scientifique

# Université Mentouri de Constantine

# Faculté des Sciences de l'Ingenieur

# Département d'Electronique

Licence en Electronique Médicale

(Académique)

**Rapport de mini**

**projet**

Rédigé par : Encadré par :

1. Aldhehabi samer
2. Alraawi yousef sadeq
3. BOUGHIDA Maroua Ysmine
4. MOUALKIA Djamila