Finger Measuring Heartbeat Module

To begin, I would not recommend buying this sensor. I could not get it to work, and I have not found anybody else who has. That said, if you already have this sensor and insist on messing with it, hopefully I can save you some time.

The sensor is made to shine an infrared led through your finger. The infrared sensor on the other side can then pick up slight changes in the light transmittance through your finger when blood is pumped in. When the sensor senses light, it becomes more resistant, causing the voltage reading in the serial monitor to go down. However, if your sensor is anything like mine, the LED is not bright enough to be detected by the sensor even when nothing is obstructing it, so the sensor constantly reads 5v. I tried replacing the LED with a stronger one and got slightly better results, but I don’t think the sensor itself is sensitive enough to be able to detect a heartbeat over random noise and movements. You could try replacing the sensor as well, but then you’ve just replaced the entire module. Continue as you see fit.

The sensor is a simple 3-pinned analog sensor. Plug the ground in the -, the 5v to the middle pin and A0 to the S pin.
The code to program the Arduino and measure the voltage output is below. However alone it does not do much good. It is difficult to see any patterns in the numbers that flash by. I highly recommend downloading the programing “Processing” and running it in tandem. Processing will take the readings from your serial monitor and graph them in an easy to look manor. Full instructions can be found here.

http://www.youtube.com/watch?v=2_c0yE9QHNI

Special thanks to that link for the great instructions and code.

Arduino Code

// Pulse Monitor Test Script
int ledPin = 13;
int sensorPin = 0;
double alpha = 0.75;
int period = 20;
double change = 0.0;
void setup ()
{
    pinMode (ledPin, OUTPUT);
    Serial.begin (9600);
}

void loop ()
{
    digitalWrite(13, HIGH);
    static double oldValue = 0;
    static double oldChange = 0;
    int rawValue = analogRead (sensorPin);
double value = alpha * oldValue + (1 - alpha) * rawValue;
Serial.print (((value*value)-100000)-942000)/10);
Serial.println (((value*value)-100000)-942000)/10);
oldValue = value;
delay (period);
}

Processing Code

import processing.serial.*;

Serial myPort;  // Create object from Serial class
int val, screen_increment, old_x=0, old_y=0;      // Data received from the serial port
String inString;  // Input string from serial port
int lf = 10;      // ASCII linefeed
void setup()
{
    size(displayWidth-100, 600);//screen size setup, display width is read into teh program, and I
    //clipped it a little bit. The screen height is set to be 600, which matches the scaled data,
    //the arduino will send over
    String portName = Serial.list()[0];//Set the Serial port COM or dev/tty.blah blah
    println(Serial.list());//look in the console below to determine which number you put in
    myPort = new Serial(this, portName, 9600);//Set up the serial port
    myPort.bufferUntil(lf);//read in data until a line feed, so the arduino must do a println
background(208,24,24);//make the background that cool blood red

}//setup

void draw()
{
//nothing in here, this is kind of like the void loop in arduino
}

void serialEvent(Serial myPort) { //this is called whenever data is sent over by the arduino

inString = myPort.readString();//read in the new data, and store in inString
inString = trim(inString);//get rid of any crap that isn't numbers, like the line feed
val = int(inString);//convert the string into a number we can use
strokeWeight(12);//beef up our white line
stroke(255, 255, 255);//make the line white

//here's where we draw the line on the screen
//we need to draw the line from one point to the next
//so we have the point we last drew, to the new point
//values are written as an x,y system, where x is left to right, left most being 0
//y is up and down, BUT 0 is the upmost point,
//so we subtract our value from the screen height to invert
//screen increment, is how we progress teh line through the screen
line(0, 0, screen_increment, 600-val);

//store the current x, y as the old x,y, so it is used next time
old_x = screen_increment;
old_y = 600-val;

//increment the x coordinate, you can play with this value to speed things up
screen_increment=screen_increment+2;

//this is needed to reset things when the line crashes into the end of the screen
if(screen_increment>(displayWidth-100)){
    background(208,24,24); //refresh the screen, erases everything
    screen_increment=-50; //make the increment back to 0,
    //but used 50, so it sweeps better into the screen
    //reset the old x,y values
    old_x = -50;
    old_y = 0;
}

}// if screen... blah blah

}//serial event