



## Your Pulse Displayed with NeoPixels

Created by Mike Barela



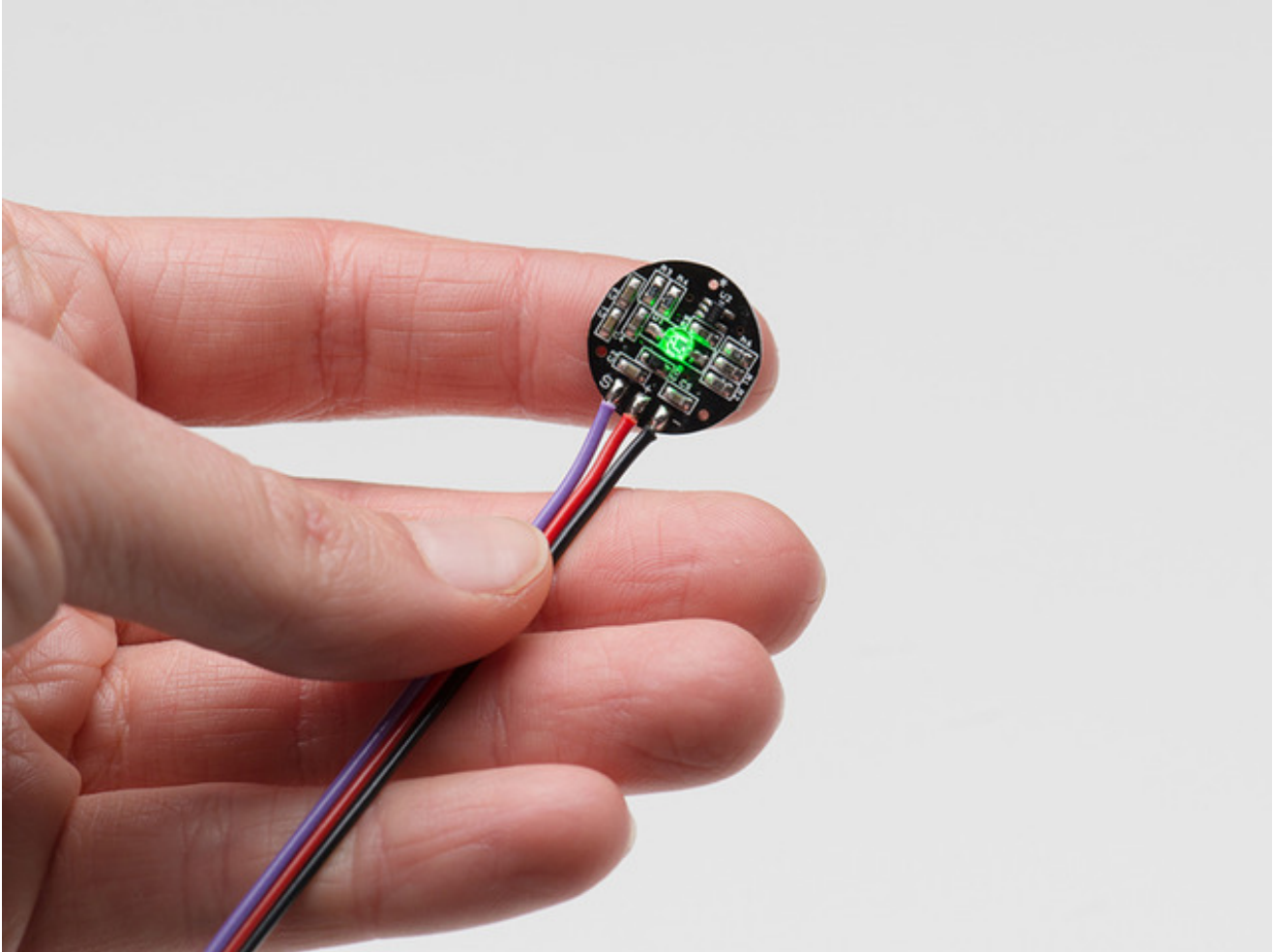
Last updated on 2015-09-29 06:17:11 PM EDT

## Guide Contents

Guide Contents	2
Overview	3
Breadboarding the Circuit	5
A Wearable Circuit	7
The Code	9
Programming	13
Use	14
The Next Level	15

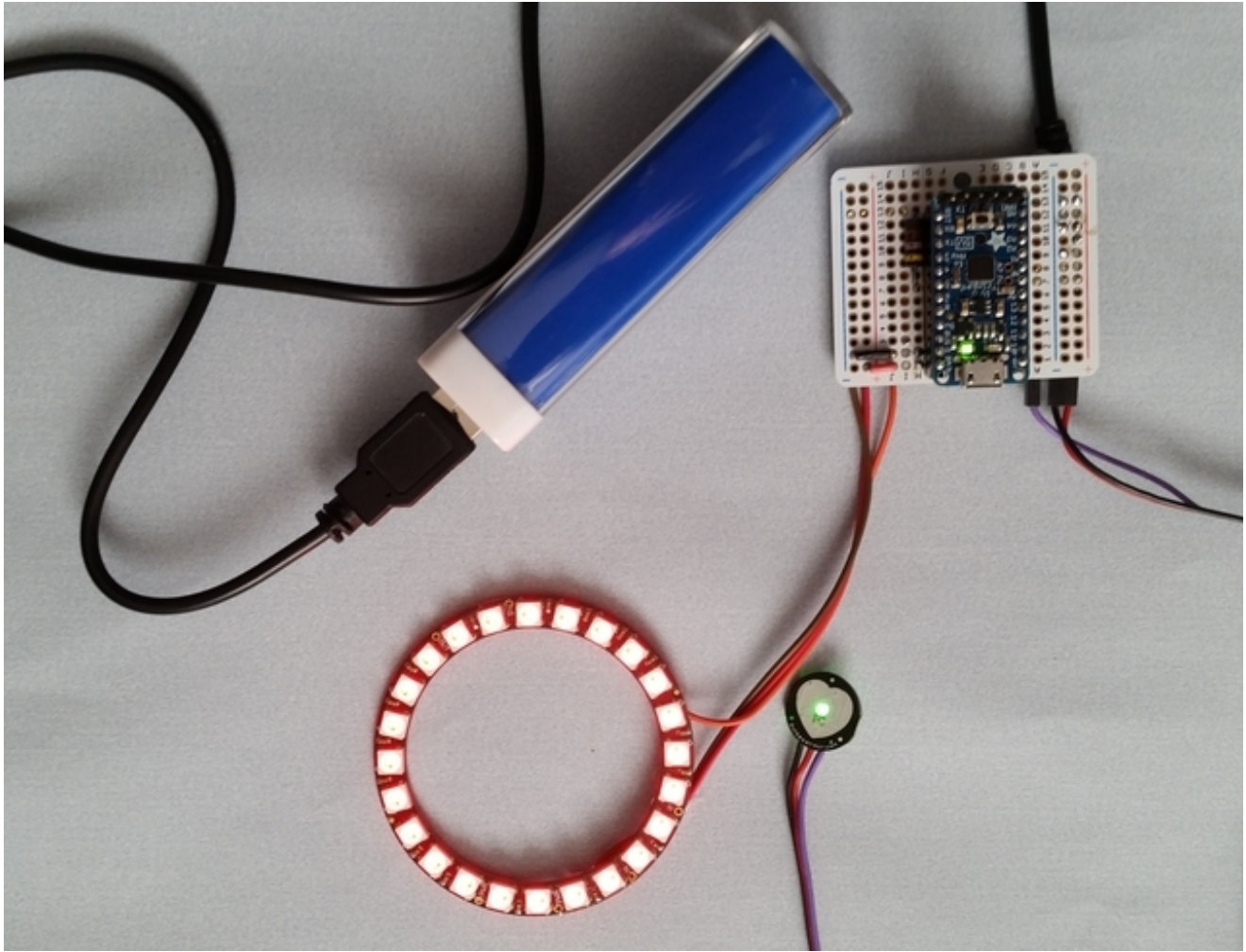
## Overview

The [Pulse Sensor Amped](http://adafru.it/1093) (<http://adafru.it/1093>), made by [pulsesensor.com](http://pulsesensor.com) (<http://adafru.it/ihE>) and sold at [Adafruit.com](http://adafruit.com) (<http://adafru.it/aK0>), is a wonderful bio sensor. With the sensor and an Arduino, some very interesting projects can be made.



The [examples of using this sensor](http://adafru.it/ihF) (<http://adafru.it/ihF>) (on the [pulsesensor.com](http://pulsesensor.com) website) start with a blinking single color LED and rapidly go into examples interacting with Processing and other computers.

This leaves room for a very interesting application: using the pulse sensor values to vary addressable LEDs such as Adafruit's NeoPixels. From a single pixel to circles (or a heart shaped design), to a costume full of pixels, people will know when your heart is racing!



This project is inspired by and designed with Laura Barela.

# Breadboarding the Circuit

---

If you would like to experiment with the parts before building a final project, using a breadboard is the way to go. This allows you to check the circuit out and think about using different components.

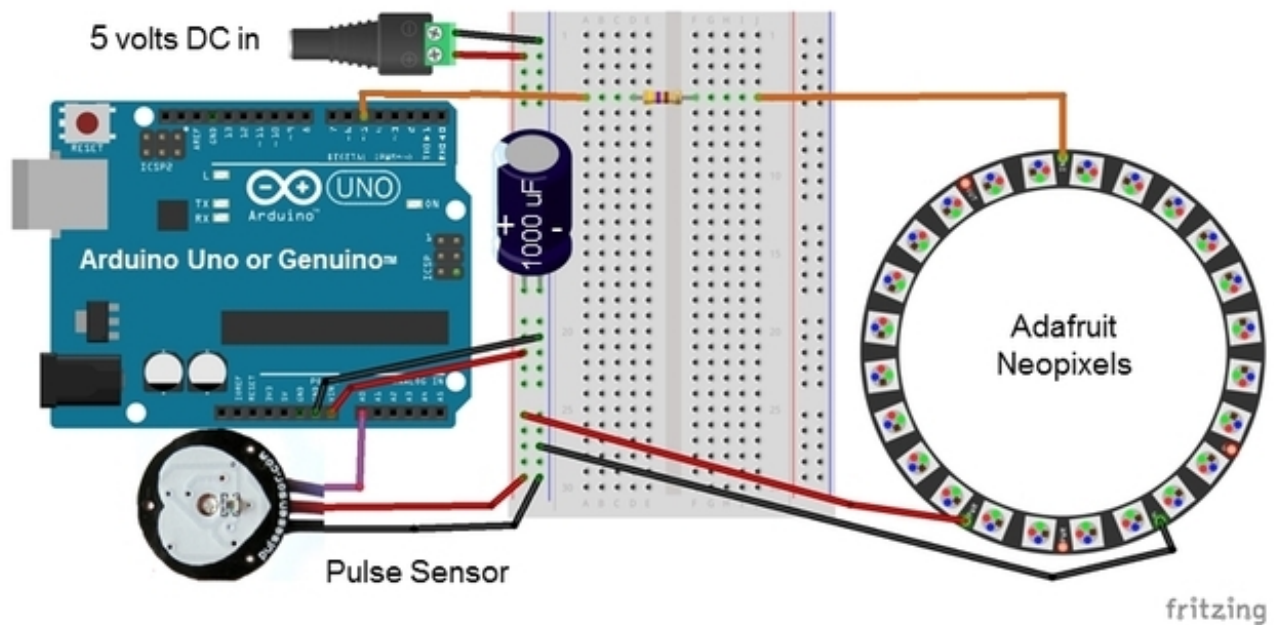
If you want to try things out, you can start with the first circuit below:

Parts List:

- [Pulse Sensor Amped \(http://adafru.it/1093\)](http://adafru.it/1093)
- [Adafruit Pro Trinket \(http://adafru.it/2000\)](http://adafru.it/2000) or Arduino Uno, Genuino, or Adafruit Metro
- 5 volt power supply (cell phone charger or wall to 5 volt supply)
- One or more Adafruit NeoPixel products, shown is the [24 NeoPixel Ring \(http://adafru.it/1586\)](http://adafru.it/1586)
- [Half Breadboard \(http://adafru.it/64\)](http://adafru.it/64)
- [2.1 mm power supply adapter \(http://adafru.it/368\)](http://adafru.it/368)
- Large Capacitor and 470 ohm resistor

Some people use a great number of NeoPixels in their projects. Each pixel draws current and the current needs can add up. The current can vary depending on the colors and brightness you choose. The values in the code set the red to about 25% brightness on red with no blue or green LED use so the current needed is much less than setting all colors to maximum brightness.

The use of a large capacitor and a data line resistor are recommended in the [Adafruit NeoPixel Überguide \(http://adafru.it/dYa\)](http://adafru.it/dYa) which I recommend you read before designing with NeoPixels. As the design below uses 24 NeoPixels, I added the components knowing that other designs might use many more pixels. The 24 pixels with the code settings selected draws 80 milliamps max, not too bad. For a small project (less than 30 pixels), you may forego the extra parts.



The breadboard circuit shows an Arduino Uno (or Genuino or Adafruit Metro, they have the same features) as the microcontroller. The pulse sensor code will work with any of the five volt ATmega 328 based AVR processor board with an analog input and a digital output. If you plan to work on the wearable version on the next page, you may wish to use the Pro Trinket 5 volt - it works the same as the Uno but is much smaller.

Due to the pulsesensor.com code, use of other microcontrollers other than a ATmega 328 would need recoding to set a 2 microsecond timer needed by the sensor. Recoding would include the Teensy line, popular with LED project designers.

For breadboarding, select a wall supply with a steady 5 volts. For up to 30 or so NeoPixels, the [Adafruit 5 Volt, 1 amp USB supply \(http://adafru.it/501\)](http://adafru.it/501) works well. For many more pixels, consider the [Adafruit 5V 10A switching power supply \(http://adafru.it/658\)](http://adafru.it/658). If you plan to go portable, the wearable version on the next page shows how to do that.



# A Wearable Circuit

---

A redesigned circuit shows a more mobile installation. A 1/4 perma-proto circuit board is used like a breadboard. A Pro Trinket 5 volt is soldered on. The pulse sensor is connected to analog pin A0 and power. The NeoPixels are connected to Digital Pin 5 through a 470 ohm resistor.

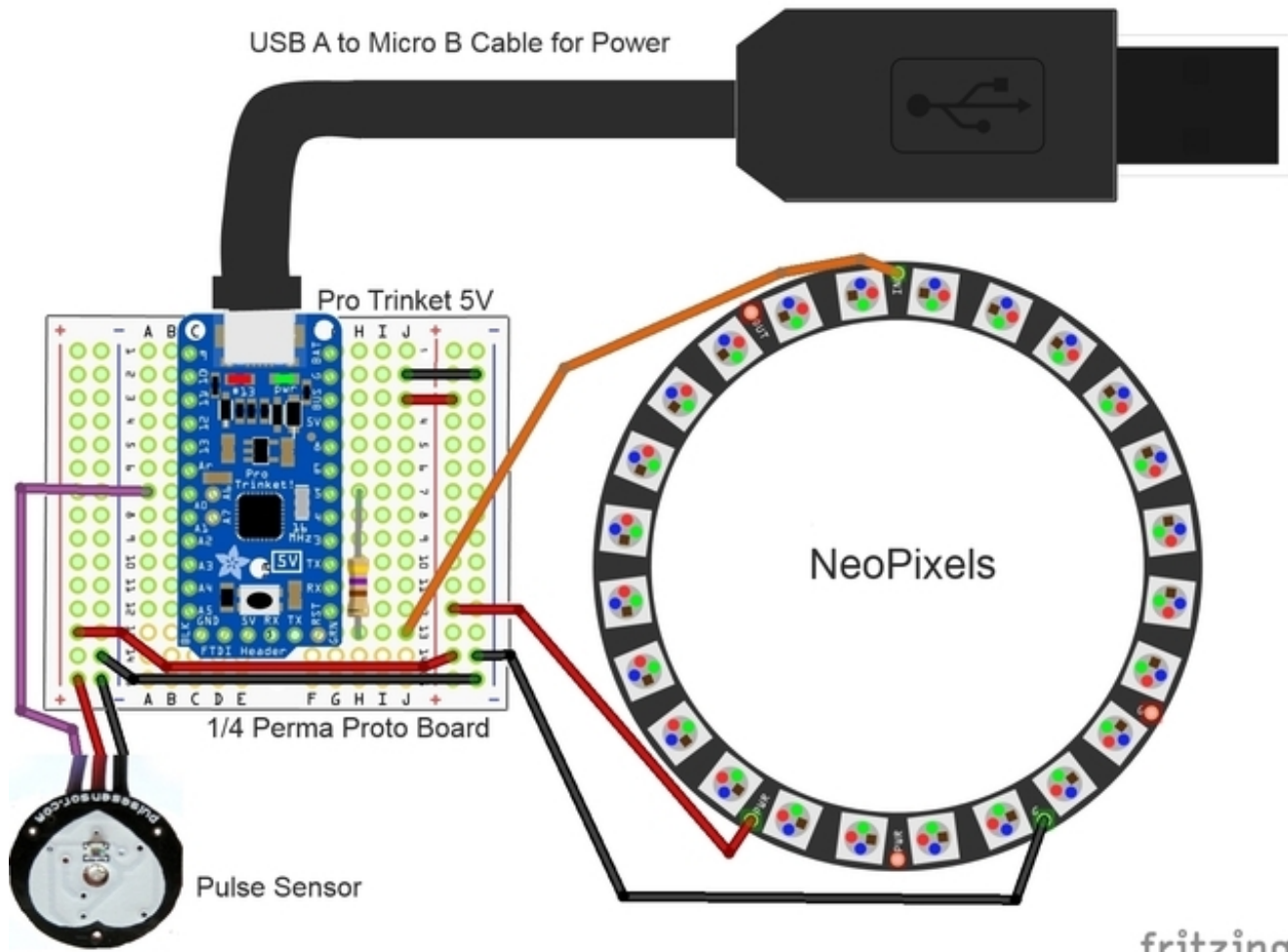
## Parts List:

- [Pulse Sensor Amped \(http://adafru.it/1093\)](http://adafru.it/1093)
- [Adafruit Pro Trinket \(http://adafru.it/2000\)](http://adafru.it/2000)
- [1/4 Perma Proto Board \(http://adafru.it/1608\)](http://adafru.it/1608)
- [USB Cable USB-A to Micro B \(http://adafru.it/iiia\)](http://adafru.it/iiia)
- One or more Adafruit NeoPixel products, shown [24 NeoPixel Ring \(http://adafru.it/1586\)](http://adafru.it/1586)
- 470 ohm resistor (value between 220 and 690 ohms should be fine, no resistor needed for a few NeoPixels).
- USB "Phone Recharge" battery pack. Size depends on the number of NeoPixels and how long the circuit should run. Adafruit has [2200 mAh \(http://adafru.it/1959\)](http://adafru.it/1959), [4400 mAh \(http://adafru.it/1565\)](http://adafru.it/1565), and [10000 mAh \(http://adafru.it/1566\)](http://adafru.it/1566) versions.
- Hookup wire (Adafruit sells many different types [like this \(http://adafru.it/1311\)](http://adafru.it/1311))

The Pro Trinket 5V may be connected to the board via the male headers pins included with the Pro Trinket. If you want a removable Pro Trinket, you can attach female header to the perma proto and male header to the Pro Trinket. The USB A side plugs into the battery pack, the Micro B side plugs into the Pro Trinket to provide power. Tape or secure the USB to Pro Trinket end if you plan to flex the USB cable, as may happen in a wearable.

Powering the Pro Trinket via the USB connector, you can take USB power out of the BUS pin for the rest of the circuit. If you intend to power alot of NeoPixels and want to use a separate power for the 'Pixels, connect a separate 5 volt supply to the NeoPixels Pwr and Gnd pins AND connect the NeoPixel Gnd to the circuit ground (-) so the signal from the board (orange) has a ground path. If you use another source of 5 volt power other than the USB cable, connect to the Perma Proto + and - lines AND put the red wire on the BAT pin on the Pro Trinket instead of the BUS line.

Connect the pulse sensor the data line (purple) to Pro Trinket pin A0 (analog pin zero). Connect the NeoPixel data in connection (be sure it is "data in" or "Din") to the resistor. Connect the other end of the resistor to the Pro Trinket digital pin 5. Do not use digital pin 3 or digital pin 11 on the Pro Trinket as the sensor code disables the pulse width timer used for those pins along with the Tone function.





# The Code

There are two pieces to the code. First is code from [pulsesensor.com](http://pulsesensor.com) which they have written to interface with the Pulse Sensor. Second is the code to read the sensor values and translate heartbeat information into brightness values for the NeoPixels.

The code for the sensor, `interrupt.ino`, is shown below. A detailed description on how it works is on the [pulsesensor.com](http://pulsesensor.com) (<http://adafru.it/iib>) website. Copy the code and place it in the same file directory as the main code, calling the file `interrupt.ino`.

```
/*
This code uses the Pulse Sensor Amped by Joel Murphy and Yury Gitman
www.pulsesensor.com
>>> Pulse Sensor purple wire goes to Analog Pin 0 <<<<
Pulse Sensor sample acquisition and processing happens in the background via Timer 2 interrupt. 2mS sample
PWM on pins 3 and 11 will not work when using this code, because we are using Timer 2!
The following variables are automatically updated:
Signal : int that holds the analog signal data straight from the sensor. updated every 2mS.
IBI : int that holds the time interval between beats. 2mS resolution.
BPM : int that holds the heart rate value, derived every beat, from averaging previous 10 IBI values.
QS : boolean that is made true whenever Pulse is found and BPM is updated. User must reset.
Pulse : boolean that is true when a heartbeat is sensed then false in time with pin13 LED going out.

NOTE: This code works with Arduino UNO or Arduino PRO or Arduino Pro Mini 5V or any Arduino running with
ATmega328 and 16MHz clock. This will disable PWM output on pin 3 and 11. Also, it will disable the tone() co

All the work to find the heartbeat and determine the heartrate happens in the code below.
For using the Pulse Sensor code, see the link below for a code walkthrough:
http://pulsesensor.myshopify.com/pages/pulse-sensor-amped-arduino-v1dot1

Code Version 02 by Joel Murphy & Yury Gitman Fall 2012
*/

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

void interruptSetup(){
// Initializes Timer2 to throw an interrupt every 2mS.
// DDRCRA = 0x00; // DISABLE PWM ON DIGITAL PINS 3 AND 11 AND GO INTO CTC MODE
```

```

TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER
OCR2A = 0x7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}

// THIS IS THE TIMER 2 INTERRUPT SERVICE ROUTINE.
// Timer 2 makes sure that we take a reading every 2 milliseconds
ISR(TIMER2_COMPA_vect){ // triggered when Timer2 counts to 124
  cli(); // disable interrupts while we do this
  Signal = analogRead(pulsePin); // read the Pulse Sensor
  sampleCounter += 2; // keep track of the time in mS with this variable
  int N = sampleCounter - lastBeatTime; // monitor the time since the last beat to avoid noise

  // find the peak and trough of the pulse wave
  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 track of lowest point in pulse wave
    }
  }

  if(Signal > thresh && Signal > P){ // thresh condition helps avoid noise
    P = Signal; // P is the peak
  } // keep track of highest point in pulse wave

  // NOW IT'S TIME TO LOOK FOR THE HEART BEAT
  // signal surges up in value every time there is a pulse
  if (N > 250){ // avoid high frequency noise
    if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
      Pulse = true; // set the Pulse flag when we think there is a pulse
      digitalWrite(blinkPin,HIGH); // turn on pin 13 LED
      IBI = sampleCounter - lastBeatTime; // measure time between beats in mS
      lastBeatTime = sampleCounter; // keep track of time for next pulse

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

  // keep a running total of the last 10 IBI values

```

```

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];         // and drop the oldest IBI value
    runningTotal += rate[i];     // add up the 9 oldest IBI values
}

rate[9] = IBI;                  // add the latest IBI to the rate array
runningTotal += rate[9];        // add the latest IBI to runningTotal
runningTotal /= 10;            // average the last 10 IBI values
BPM = 60000/runningTotal;      // how many beats can fit into a minute? that's BPM!
QS = true;                      // set Quantified Self flag
// QS FLAG IS NOT CLEARED INSIDE THIS ISR
}
}

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

if (N > 2500){                     // if 2.5 seconds go by 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 = true;                // when we get the heartbeat back
}

sei();                              // enable interrupts when youre done!
} // end isr

```

The main program melds the pulse sensor reading with NeoPixel output.

```
/* Pulse Sensor Amped with NeoPixels
```

```
Pulse an arbitrary number of Adafruit NeoPixels based on a heartbeat sensor
```

```
The pulsesensor.com code needs to be in module interrupt.ino in the sketch directory
```

```

http://pulsesensor.com/pages/pulse-sensor-amped-arduino-v1dot1
Code also uses the Adafruit NeoPixel library code discussed at
https://learn.adafruit.com/adafruit-neopixel-uberguide

Version 1.0 by Mike Barela for Adafruit Industries, Fall 2015
*/
#include <Adafruit_NeoPixel.h> // Library containing

// Behavior setting variables
int pulsePin = 0; // Pulse Sensor purple wire connected to analog pin 0
int blinkPin = 13; // Digital pin to blink led at each beat
int fadePin = 5; // pin to do fancy neopixel effects at each beat
int fadeRate = 0; // used to fade LED on with PWM on fadePin

// these variables are volatile because they are used during the interrupt service routine
volatile int BPM; // used to hold the pulse rate
volatile int Signal; // holds the incoming raw data
volatile int IBI = 600; // holds the time between beats, the Inter-Beat Interval
volatile boolean Pulse = false; // true when pulse wave is high, false when it's low
volatile boolean QS = false; // becomes true when Arduino finds a beat.

// Set up use of NeoPixels
const int NUMPIXELS = 24; // Put the number of NeoPixels you are using here
const int BRIGHTNESS = 60; // Set brightness of NeoPixels here
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUMPIXELS, fadePin, NEO_GRB + NEO_KHZ800);

void setup(){
  pinMode(blinkPin,OUTPUT); // pin that will blink to your heartbeat!
  // Serial.begin(115200); // Serial output data for debugging or external use
  strip.begin();
  strip.setBrightness(BRIGHTNESS);
  for (int x=0; x < NUMPIXELS; x++) { // Initialize all pixels to 'off'
    strip.setPixelColor(x, strip.Color(0, 0, 0));
  }
  strip.show(); // Ensure the pixels are off
  delay(1000); // Wait a second
  interruptSetup(); // sets up to read Pulse Sensor signal every 2mS
}

void loop(){
  // sendDataSerial('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
  // sendDataSerial('B',BPM); // send heart rate with a 'B' prefix
  // sendDataSerial('Q',IBI); // send time between beats with a 'Q' prefix
    QS = false; // reset the Quantified Self flag for next time
  }
}

```

```

}
ledFadeToBeat();          // Routine that fades color intensity to the beat
delay(20);                // take a break
}

void ledFadeToBeat() {
  fadeRate -= 15;          // Set LED fade value
  fadeRate = constrain(fadeRate,0,255); // Keep LED fade value from going into negative numbers
  setStrip(fadeRate);     // Write the value to the NeoPixels
  // sendDataSerial('R',fadeRate);
}

void sendDataSerial(char symbol, int data ) {
  // 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
}

void setStrip(int r) { // Set the strip to one color intensity (red)
  int g = 0;          // Green is set to zero (for non-red colors, change this)
  int b = 0;          // Blue is set to zero (for non-red colors, change this)
  for (int x=0; x < NUMPIXELS; x++) {
    strip.setPixelColor(x, strip.Color(r, g, b));
  }
  strip.show();
}
}

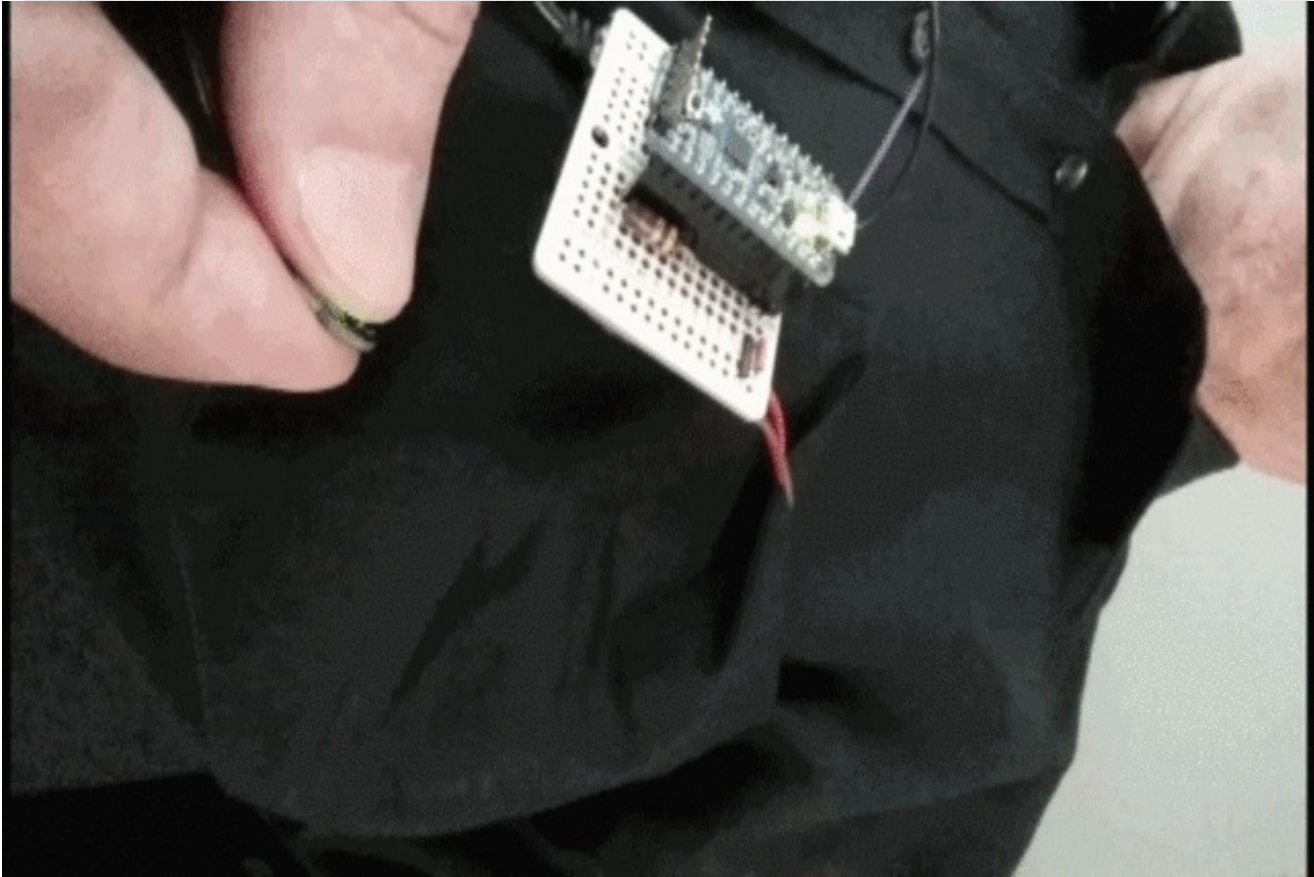
```

## Programming

With both `interrupt.ino` and `pulsesensor.ino` in your code directory on your computer, open the Arduino programming environment. Select Tools then your processor type (Like Pro Trinket 5V/16 MHz USB). Plug your Pro Trinket in, press the Pro Trinket reset button then select upload to place the code on the Pro Trinket. For detailed instructions on placing your code on the Pro Trinket via the Arduino IDE, see the Pro Trinket tutorial at <https://learn.adafruit.com/introducing-pro-trinket/starting-the-bootloader> (<http://adafru.it/iic>).

If you would like to debug the program or use the pulse sensor data on other processors, uncomment the lines outputting data to the Serial port.

## Use



The video above shows the NeoPixel ring in a pocket with the pulse sensor between two fingers. The sensor package comes with velcro for using the sensor on one finger and an ear clip for handless Bajorian pulse sensing on your earlobe.

The circuit board is out of the pocket for the demo above. This may be tucked in the pocket along with a battery for costuming. The USB cable connecting the circuit board and the battery may be routed to another pocket or other inconspicuous place.

Ensure the sensor is against the skin but not too tight. Adjust it for a steady beat. You can use the Pro Trinket/Arduino onboard LED to view the flashing in addition to seeing the NeoPixels flash.

If you do not get the NeoPixels to flash:

- If the Pro Trinket/Arduino LED 13 flashes, the processor is working along with the code. No flash? Be sure you are holding the sensor and that you loaded the code from the previous page onto the Pro Trinket/Arduino.
- The green LED in the center of the sensor should be on when it is connected correctly to the Pro Trinket/Arduino. Check connection wires and power to the project if you do not see the green sensor LED (in the center of the heart). If you see the green LED, gently press the



sensor between two fingers to detect your pulse.

## The Next Level

The obvious shape for the NeoPixels would be a heart. NeoPixel strips are not very flexible. [Individual NeoPixels \(http://adafru.it/1260\)](http://adafru.it/1260) may be soldered in shapes useful for costuming.

A [NeoPixel Tiara \(http://adafru.it/fR0\)](http://adafru.it/fR0) may be pulsed with the Pro Trinket and an ear clipped Pulse Sensor. A dress bodice or skirt could also pulse to a heartbeat. May this project be your start to using the Pulse Sensor and NeoPixels together.