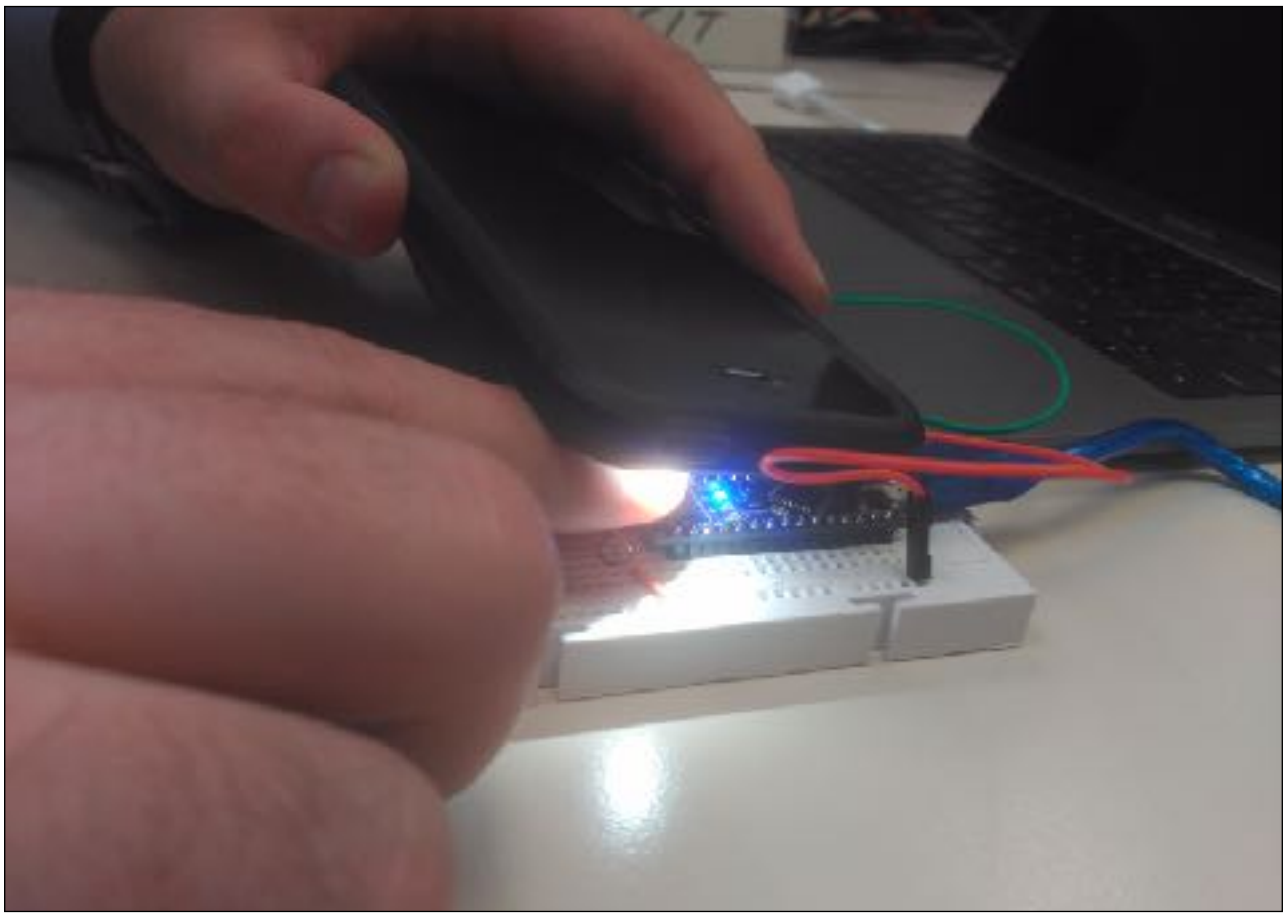
Physical Computing Based on Open Software and Hardware Platforms

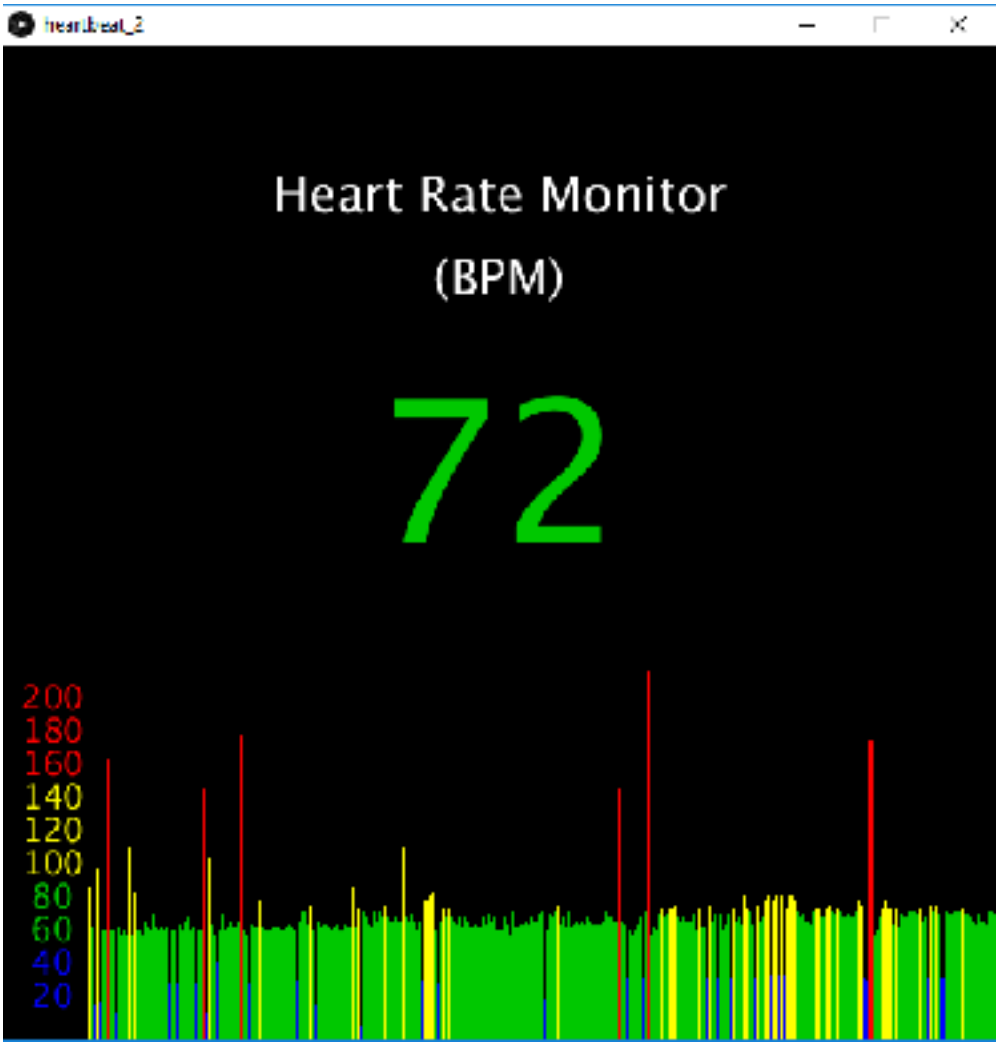
Heart Rate Monitor - Athens November 2017

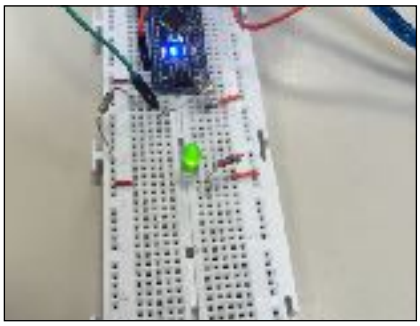
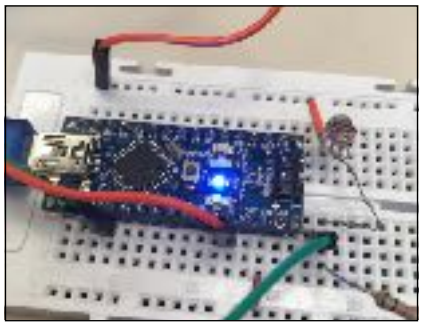
Tyler De Belder & Montserrat Aroca Roca - 17 November 2017



Introduction

Measuring a persons heart rate is a excellent way to monitor their health. The goal of this project was to measure a heart rate using a photoresistor to measure the change of light intensity through the persons finger. The voltage over this resistor changes according to the light intensity shining on it. The amount of light able to propagate through the finger changes with the amount of blood in the fingertip. The change in voltage can then be measured and sampled (20ms) using the Arduino. This signal is then filtered using a high-pass and low-pass filter at 0.833Hz and 3Hz respectively. After filtering the signal only contains the frequency range of a human heart rate. There is still noise in the signal but this is then reduced by making rolling averages of the samples. The heart rate in beats per minute (BPM) can then be extracted by measuring the time between the peaks of the signal. This BPM value is then transferred to the serial port of the Arduino which is then read by the GUI running in Processing. This displays the real time heart rate in BPM and also the value of the BPM over the last 550 samples on a graph at the bottom of the display. This graph gets moved to the left every time a new sample comes through the serial port.



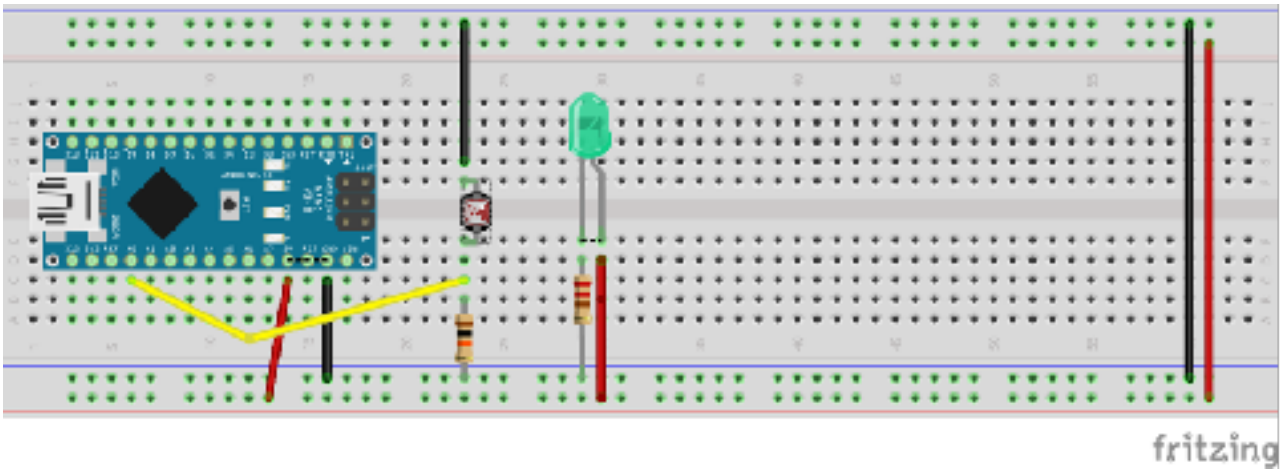



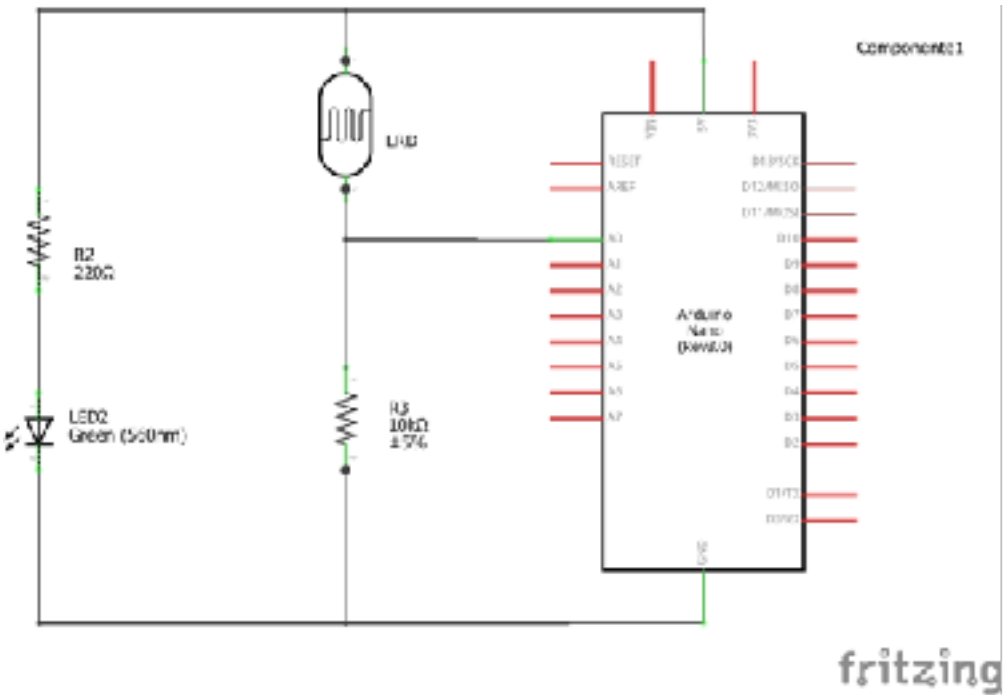

Materials and Setup

The materials used in the project are:

- Arduino Nano
- Green LED (used the iPhone flash as a light instead because it has more brightness)
- Photoresistor (LDR)
- 10k ohm resistor
- 220 ohm risistor
- Breadboard
- Cables
- USB cable

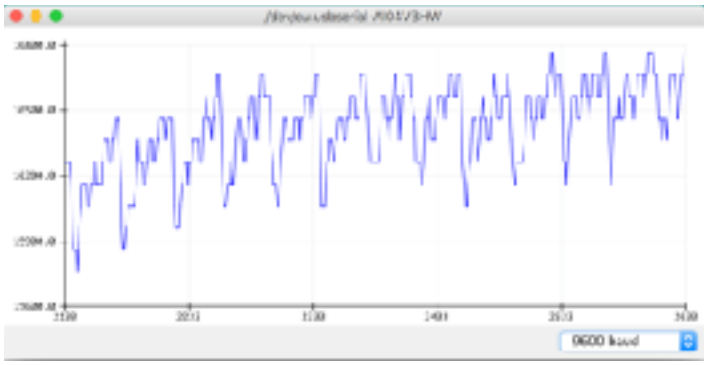
Schematic (made using Fritzing)



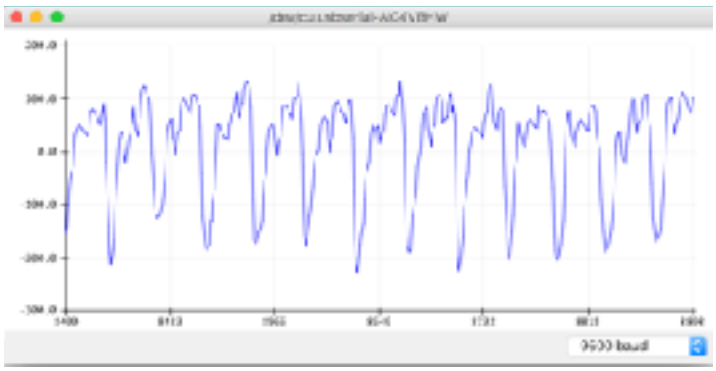


Signal processing

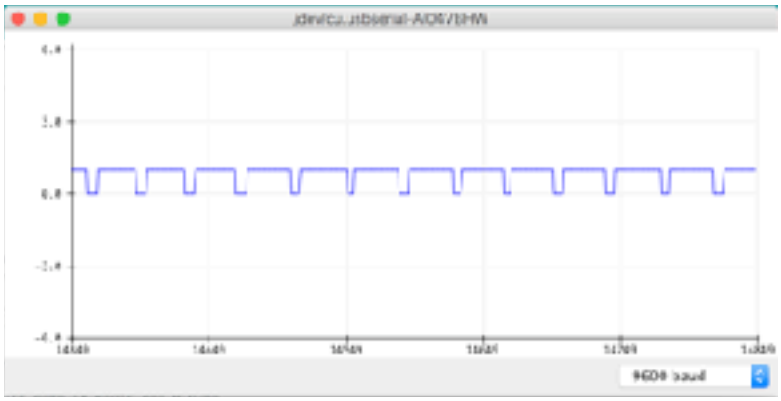
Unfiltered heart rate signal:



Filtered heart rate signal (high-pass and low-pass)



Averaged signal over 500 samples compared to the average of the last 10 samples. If the average of the last 10 is larger than X*the average value over 500 samples print a 1 otherwise it is 0. The parameter X needs to be experimentally determined for different fingers.



The time between the rising edges is then determined and then converted to a BPM signal.

Evaluation of the project

The heart rate monitor returned accurate values of for the persons heart rate. The value for the threshold value for comparing the averages needs to be experimentally tested to get accurate results. This value must be changed for different fingers. The value X=10 works well for male fingers and X=5 for female fingers. The light used to illuminate the finger was the flash of an iPhone because of the higher power output compared to the green LED powered by the Arduino. The signal from the green LED was too weak to obtain accurate results.

Future improvements

- Make a clip to attach to the finger
- Used a high power LED powered from the Arduino so the iPhone doesn't have to be used as the light
- Make it wireless
- Transit the heart rate signal to processing and plot this as well as the BPM plot.
- Improve the accuracy
- Determine the average heart rate over a period of time.

Arduino Code

```
#include <FilterDerivative.h>
#include <FilterOnePole.h>
#include <Filters.h>
#include <FilterTwoPole.h>
#include <FloatDefine.h>
#include <RunningStatistics.h>

float HeartbeatSignal;
float FilteredHeartbeatSignal;
float HighpassFilteredHeartbeatSignal;
int values[500];
int values10[10];
int values30[30];
int counter;
int i;
int j;
int counter10;
int counter30;
int sum;
int sum10;
int sum30;
int average;
int average10;
int average30;
int oneorzero;
int oneorzeroprevious;
int BPM;
unsigned long start, finished, elapsed;

void setup() {
//initializing the variable values
```

```
//values[500] can be changed to make the average signal average over a shorter or
  //longer period of time
int values[500];
int counter = 0;
int i = 0;
int j=500;
int sum=0;
int sum10=0;
int sum30=0;
int counter10 = 0;
int counter30 = 0;
int average10 = 0;
int average30 = 0;
int oneorzero = 0;
int oneorzeroprevious = 0;

Serial.begin(9600);
}
// filter out frequencies lower than 0.833 Hz.
float HighPassFilterFrequency = 0.833;

// highpass filter that only keeps frequencies above HighPassFilterFrequency
FilterOnePole filterOneHighpass( HIGHPASS, HighPassFilterFrequency );

// filter out frequencies greater than 3.33Hz
float lowFilterFrequency = 3.33;

// create a lowpass filter that only keeps frequencies below lowFilterFrequency
FilterOnePole filterOneLowpass(LOWPASS, lowFilterFrequency);

void loop() {

//The next line applies a band pass filter to the signal

  HeartbeatSignal = 100*analogRead(A0);
  HighpassFilteredHeartbeatSignal = filterOneHighpass.input(HeartbeatSignal);
```

```
        FilteredHeartbeatSignal =
filterOneLowpass.input(HighpassFilteredHeartbeatSignal);
        //delay to sample at aproximately 20ms
        delay(20);
        //Serial.println(FilteredHeartbeatSignal);
      //filling the array sequentially and when it gets to the end fill it in the 0 place.
      if (counter = j){
        counter = 0;
        values[0] = FilteredHeartbeatSignal;
        counter++;
         }
      else {
         values[counter] = FilteredHeartbeatSignal;
         counter++;
      }
      //make the sum of all the values in the array
      for (i = 0; i < j; i++) {
        sum = values[i] + sum;

      }
      //average value over the last j samples
      average = sum/j;

      //filling the array sequentially and when it gets to the end fill it in the 0 place.
      if (counter10 = 10){
        counter10 = 0;
        values10[0] = FilteredHeartbeatSignal;
        counter10++;
         }
      else {
         values10[counter10] = FilteredHeartbeatSignal;
         counter10++;
      }
      //make a sum of all the values in the array of the last 10 values
      i=0;
      sum10=0;
```

```
for (i = 0; i < 10; i++) {
  sum10 = values10[i] + sum10;

  }
 //average value of the last 10 samples
average10 = sum10/10;

 //Serial.println(average10);

 /*    //not used because the 10 gave better results
 //  average of the last 30 samples
if (counter30 = 30){
  counter30 = 0;
  values30[0] = FilteredHeartbeatSignal;
  counter30++;
    }
else {
   values30[counter30] = FilteredHeartbeatSignal;
   counter30++;
}

i=0;
sum30=0;
for (i = 0; i < 30; i++) {
  sum30 = values30[i] + sum30;
  }
average30 = sum30/30;
*/

 //comparing the averages of the last 10 samples compared to the last j samples multiplied by 10.
 //The value 10 can be changed if for different peoples fingers to get good results
 //This will set a value of 1 or 0 to the signal. if the signal is high compared to the average it is 1 otherwise it is 0
if (average10 > 10*average) {
  //Serial.println(1);
```

```
  //set the previous value of oneorzero
  oneorzeroprevious = oneorzero;
  //set oneorzero to 1
  oneorzero = 1;
 }
 else {
  //Serial.println(0);
  oneorzeroprevious = oneorzero;
  oneorzero = 0;

 }

 //determines if the change of the oneorzero and oneorzeroprevious has changed
with.
 //if there is a change then it starts the timer and calculates the time since the last
 //timer started. Then BPM is written to the serial port for data in the GUI
 if (oneorzero < oneorzeroprevious) {
  //finishes the timer that was started in the previous loop
  finished=millis();
  //time elapsed between the start of the timer and when it finished in this loop
  elapsed = finished - start;
  // calculates the heart rate in Beats Per Minute
  BPM = 60000/elapsed;
  // Serial.println(BPM); //for debugging

  //writes the BPM value to serial
  Serial.write(BPM);

  //start the timer
  start = millis();

 }
 else {
  // Serial.println(1); //for debugging
 }
 }
```

Processing Code

```
import processing.serial.*;


int val = 0; // To store data from serial port, used to color background
Serial port; // The serial port object
int xPos = 600;  //sets the width of the display
int oldvalue = 0;
int[] array = new int[550];
int i = 0;
int j = 0;

void setup() {
  //make the window
  size(600, 600);
  //read the serial
  port = new Serial(this, Serial.list()[0], 9600);
  i = 0; //initialize the counter variables
  j = 0;
}

void draw()

{
  background(0, 0, 0); //set the background black
  //make the title
  fill(255, 255, 255); //make the title color white
  textSize(30);
  textAlign(CENTER);
  text("Heart Rate Monitor", width/2, height/2-200);
  text("(BPM)", width/2, height/2-150);
```

```
//Heart rate: change the color in function of the current heart rate
if (array[j]>150) {
  fill(255, 0, 0);
} else if (array[j]<50) {
  fill(0, 0, 255);
} else if (array[j]>=50 && array[j]<80 ) {
  fill(0, 200, 0);
} else {
  fill(255, 255, 0);
}

//display the heart rate
textSize(120);
textAlign(CENTER);
text(val, width/2, height/2);        //shows the value at the center of the screen


//set the color of the values on the y axis on the graph
fill(255, 0, 0);
textSize(20);
text("200", 30, 400);

textSize(20);
text("180", 30, 420);

textSize(20);
text("160", 30, 440);

fill(255, 255, 0);
textSize(20);
text("140", 30, 460);

textSize(20);
text("120", 30, 480);

textSize(20);
```

```
  text("100", 30, 500);

  fill(0, 200, 0);
  textSize(20);
  text("80", 30, 520);

  textSize(20);
  text("60", 30, 540);

  fill(0, 0, 255);
  textSize(20);
  text("40", 30, 560);

  textSize(20);
  text("20", 30, 580);



  for (int j = 0; j<549; j = j+1) {
    if (array[j]>150) {
      stroke(255, 0, 0);
    } else if (array[j]<50) {
      stroke(0, 0, 255);
    } else if (array[j]>=50 && array[j]<80 ) {
      stroke(0, 200, 0);
    } else {
      stroke(255, 255, 0);
    }
    line(xPos -j, height, xPos -j, height - array[j]);
  }
}


// Called whenever there is something available to read
void serialEvent(Serial port) {
```

// Data from the Serial port is read in serialEvent() using the read() function and assigned to the global variable: val

```
  //move the values down in the array
  for (int i = 548; i>=0; i=i-1) {
    array[i+1] = array[i];
  }

  //assign the value from the serial
  val = port.read();
  //add the value to the first spot in the array
  array[0] = val;
}
```