# Pulse sensor

November 6, 2021

**Introduction**

We use the pulse sensor to get the live heart rate data. The pulse sensor uses a method of pulse detection called Photoplethysmogram. With every heart beat we will have the associated waveform and by counting the number of peaks per minute results in the heart rate.

# Contents

# 1 How does a pulse sensor work

The pulse sensor has a green led light and and a photo sensor, when the green led light shines bright in to your finger, the oxygenated haemoglobin in the blood absorbs the amount of light and the photo-sensor can sense the amount of light reflected back. So when ever the blood gets pumped out the the heart for circulation, the concentration of the blood is higher that is more green light is absorbed and then there is a drop in the pulse that is when there is no heart beat, the amount of light absorbed is smaller. Continuously recording this highs and lows will result us in a pulse wave form as shown in the Fig. 1 and we can calculate the number of beats per minute.

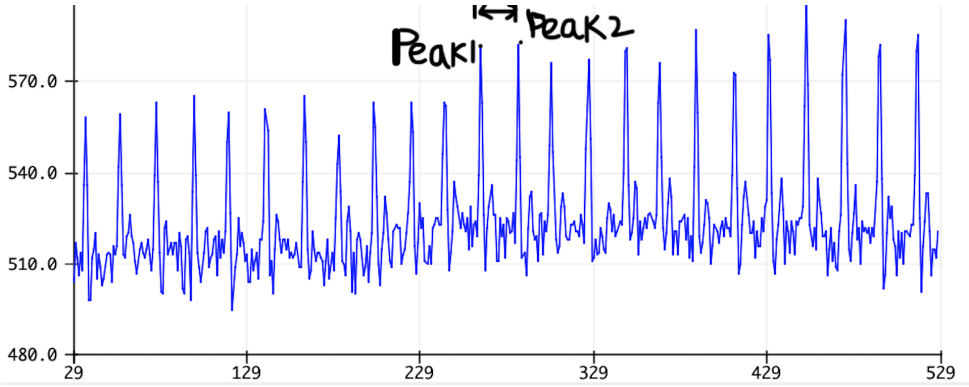


Figure 1: Raw pulse Waveform output from arduino serial plotter

# 2 Project Implementation

The hardware components required are the Arduino Nano 33 BLE Sense, a pulse sensor, connecting wires and a power supply. You can directly power the Arduino by plugging it to your laptop using a USB cable. And the software required will be an Arduino integrated development environment (IDE) which can be downloaded free from the site of Arduino.cc

## 2.1 *Wiring Pulse Sensor with Arduino*

The wiring is quite simple, the positive pin(red) of the pulse sensor is connected to the +5v of the Arduino, negative pin(black) is connected to ground and the data pin(yellow) is connected to the analog pin A0 of the Arduino board. The wiring is shown in the Fig. 2.

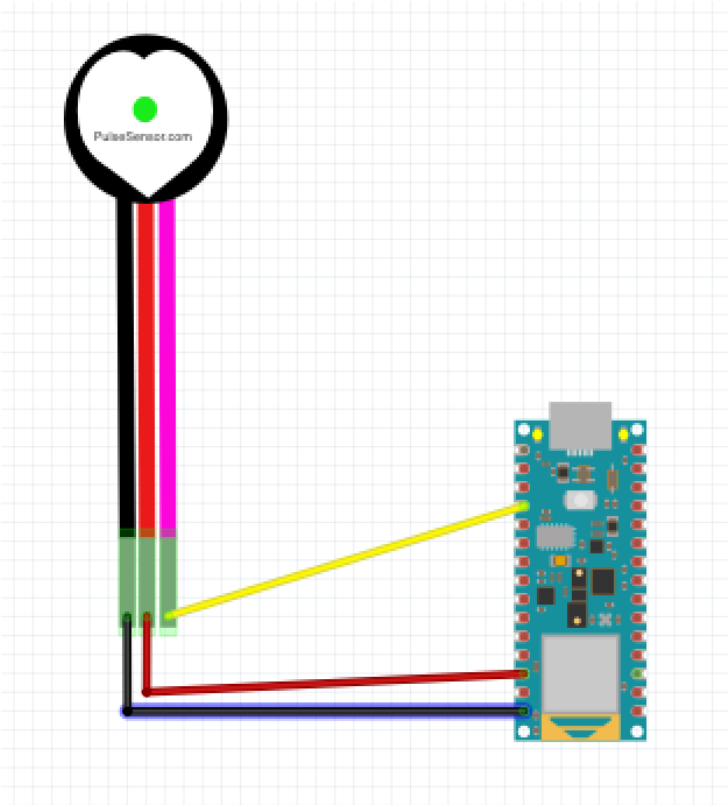

Figure 2: Wiring diagram of the system

### 2.1.1 *Arduino Nano 33 BLE Sense*

The Arduino nano 33 BLE sense is the successor of the typical nano board. There are two variants of the board the Nano BLE and nano BLE sense. The pin out diagram of the Arduino Nano 33 BLE Sense is shown in the Fig. 4 It is Arduino's 3.3v AI enabled board in the smallers available form factor: 45x18mm. They are not based on a microchip processor but an ARM Cortex-M4F processor running at 64 MHz which will allow you to make larger programs than with the Arduino Uno (it has 1MB of program memory, 32 times bigger), and with a lot more variables (the RAM is 128 times bigger), they are also equipped with a u-blox NINA B306 module for BLE and Bluetooth 5 communication which built on top of Nordic nRF52840. Nano 33 BLE sense comes with a much larger set of sensors in addition to the 9-axis IMU on board. It has

- 9 axis interial sensor
- humidity and temperature sensor
- barometric sensor,
- microphone,
- gesture, proximity, light color and light intensity sensor.

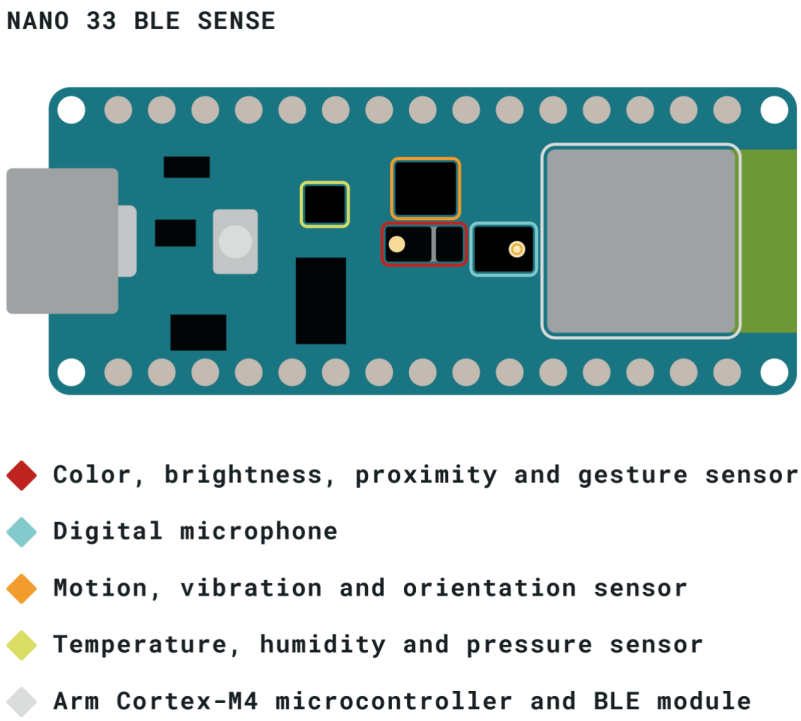


Figure 3: Nano BLE Sensor's description and location

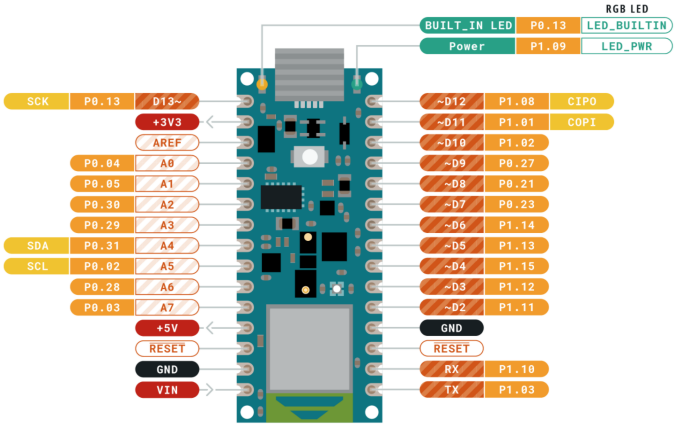


Figure 4: Pinout diagram of Arduino Nano 33 BLE Sense

### 2.1.2 *Code*

```
1
/*
3   Pulse Sensor

5   This code reads the data from the pulse sensor and then
     gives us the heart rate based on the number of peaks in
     the signal with in a minute.
```

```
*/


#define pulsePin A0 // Analog input pin.


//  VARIABLES
//int rate[10];
int BPM = 0;
int Signal = 0;
int threshold = 600;
int current_value = 0;
int previous_value = 0;
int peak = 512;
int trough = 550;
bool peak_flag = false;
bool trough_flag = true;
int peak_time = 0;
int past_peak_time = 0;
int IBI = 0;
unsigned long previousMillis = 0;

const long interval_1 = 6000;
bool Start_flag = false;
bool past_peak_time_flag = false;

// Circular Buffer implementation.
struct CircularBuffer{
  CircularBuffer(size_t size);
  void addElement(float newElement);
  float average();

  float* array;
  float* next;
  size_t size;
  int index = 0;
  int num_elements = 0;
};

CircularBuffer::CircularBuffer(size_t arraySize)
{
```

```
  size = arraySize;
  array = new float[arraySize];
  next = &array[0];
}

void CircularBuffer::addElement(float newElement)
{
  *next = newElement;
  index++;
  if (index >= size)
  {
    index = 0;
  }
  next = &array[index];
  if(++num_elements > size)
  {
    num_elements = size;
  }
}

float CircularBuffer::average()
{
  float average = 0;
  for(int i = 0; i < num_elements; i++)
  {
    average += array[i];
  }
  return average/num_elements;
}

// circular buffer of size 10 that holds peak values which
   is averaged to calculate heart rate  implementation

CircularBuffer myBuffer(10);  // create a 10 element (float
   ) circular buffer


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

}
```

```
87
void loop() {
89              Signal = analogRead(pulsePin);              //
    Read the analog pin
                if (Signal < 250) {                         //
    whenever there is a finger on the sensor the signal
    drops, this case is to recognise when the finger is
    placed on the sensor
91                Serial.println("signal is < 250");
                  Start_flag = true;                        //
    set the start_flag
93              }
                if (Start_flag == true){
95                Peak_trough();                            //
    calculate the peaks and eventually compute Heart rate
                }
97
                if (Start_flag == false){                   //
    this flag is set when there no peak for certain interval
99                past_peak_time_flag = false;              //
    this flag is reset and set to true when there is a
    finger on the sensor.
                  Serial.println("no pulse");
101             }
                delay(50);
103 }


105
void Peak_trough() {
107   if (Signal > threshold){
        // checks if the signal is greater than threshold
      if (Signal < previous_value && peak_flag == false){
        // checks if the signal value is less than previous
    peak.
109       peak_flag = true;
          // the peak is found
          trough_flag = false;
          // trough flag is set to false
111       peak = previous_value ;
          // the peak value is set to previous value
          peak_time = millis();
```

```
      IBI = peak_time - past_peak_time;
      // calculate the interval between two peaks.
      if (IBI < 1500){
        Serial.print("IBI: ");
        Serial.println(IBI);
        myBuffer.addElement(IBI);
      // Add the element to the circular buffer
        BPM = myBuffer.average();
      // average the circular buffer

        BPM = 60000/BPM;
      // calculate the Beats per minute
        Serial.println();
        Serial.print("BPM: ");
        Serial.println(BPM);
        past_peak_time_flag = true;
      // Keep this flag true as long as the finger is on
 the sensor.
      }

    }
  }
  past_peak_time = peak_time;
      // keep track of the past peak value

  if (Signal < threshold){
      // check if the signal is less than threshold to look
   for trough
    if (Signal > previous_value && trough_flag == false) {
      // if the signal at present time point is greater
 than past time point then the value at the past
 timepoint is the trough value.
      peak_flag = false;
      // set the peak flag to false
      trough_flag = true;
      // set the trough flag to true
      trough = previous_value;
      // track the trough value
    }
  }
  previous_value = Signal;
```

```
139
141
    unsigned long currentMillis_1 = millis();
143 if (currentMillis_1 - past_peak_time >= interval_1 &&
     past_peak_time_flag == true) { // if the time interval
     is greater than threshold, there is no finger on the
     sensor
      Serial.println("Start_flag is < false");
145   Start_flag = false;
      threshold = 600;
                                  // reset the threshold.
147
    }
149 }
```

Listing 1: PulseSensor.ino

## 3 Setup

- connect the pulse sensor to analog pin A0 of the arduino board.
- Connect the arduino board to the computer via USB.
- open the PulseSensor.ino code and upload the code to the board.
- open the serial monitor and select the baud rate to 9600.

## 4 Result

This arduino console prints the interval between two heart beats, and the average beats per minute.

## 5 Troubleshooting

If you encounter dramatic results like increased heart rate or larger interval between beats, try lifting your finger and placing it again. There are a couple of other things for obtaining proper heart rate

- kindly not hold the sensor too hard or too soft as it might result in no blood flow through the finger or invent noise, gentle press of the finger over the sensor results in the best results.
- Try it with different threshold values on line 146 in the arduino code for the signal as it can vary from person to person, the range of the signal is from 0 to 1023, lowering the threshold value increases the sensitivity of the pulse.

- applying different pressure to the sensor results in distorted signal, so it's advised to apply constant pressure to the sensor.
- It is advised to stick the sensor on the velcro dot and make a finger strap with the velcro tape as it helps in applying constant pressure.