# Homemade Arduino Heart Rate Monitor Project

In this day and age, keeping track of your vital parameters and biomarkers remains more critical than ever and due to many advances in medical technology, it is now more accessible than ever to do so. With portable blood pressure monitors, finger pulse oximeters and **heart rate monitors/tracker** built as standalone devices or into common consumer devices such as smartwatches, measuring such physiological parameters is certainly much more convenient than having to rely on manual/analogue devices. Here TechSparks will make a DIY Heart Rate Monitor via **Arduino**.

Table of Contents



Latest Post | Random Post | Popular Post

What is a rigid-flex circuit board?

July 27, 2023

What does SMT mean in the electronics industry?

July 27, 2023

Comprehensive Overview of PCB Additive Manufacturing Process

# Components to prepare for a DIY heart rate monitor

In the Arduino Heart Rate Monitoring Project TechSparks will be using a HW-502 heartbeat sensor, an OLED display (to display the heart rate readings), and an Arduino, which is a simple project. However, do keep in mind that the sensor used in this project should not be used as a replacement for more accurate, commercial medical equipment.

In terms of how the HW-502 sensor is built, it essentially is composed of a phototransistor and an infrared diode. When your finger is placed onto the phototransistor to measure heart rate, the phototransistor measures how much light is being detected (i.e. passed through your finger) as emitted from the photodiode. The sensor then outputs this information as an analogue signal and when blood flows through your finger (pulse), the amount of light detected by the photodiode will change and thus will be used to determine heart rate.

It is essential to note that external light can certainly alter the accuracy of the sensor and resultantly, there may be fluctuations and errors in the heart rate output. Therefore, I would recommend measuring your heart rate in a dimly lit area away from any mains-powered devices that may emit infrared radiation, disrupting the sensor.

In terms of the display used, a 0.96" 128×64 i2c OLED display will once again be used in this project as a visual display and due to its standard 4-wire i2c connection interface, connecting it up to the Arduino will not be difficult. In addition to the display of heart rate, within the serial plotter of the Arduino IDE, you will be able to visualize pulse readings as a continuous line graph

July 24, 2023

Through Hole PCB Assembly Guidelines

July 23, 2023

How to carry out the PCB manufacturing process step by step?

July 21, 2023

What is difference between PCB and PCBA

July 18, 2023

Arduino IDE, you will be able to visualize pulse readings as a continuous line graph.

Although this project is quite simple to understand, it definitely can be built upon and upgraded if you wish by combining this sensor with other health sensors to measure other health markers. Furthermore, by fabricating a custom PCB for this project and a case for this project, building a fully functional portable heart rate monitor as a next step should be effortless. The components needed to build this project are as follows:

- Arduino Nano (other Arduino-compatible boards will work)
- USB cable (compatible with the Arduino board)
- Breadboard
- Male-Male Jumper Wires (7)
- 0.96" 128×64 i2c OLED Display
- HW-502 Heartbeat Sensor

## Heart Rate Monitor Circuit Diagram

Depending on your Arduino board, you may or may not require a breadboard to plug your board in. In this example, an Arduino Nano is used, thus requiring a breadboard but if you are using an Arduino Uno, for example, the jumper wires can be plugged in from the components on the breadboard directly to the board pins. However, the wiring from the HW-502 sensor module and the OLED to the Arduino board remains the same. A circuit diagram is also featured below.

1. HW-502 Sensor: Connect the signal (S) pin to A0, the positive (+) pin to +5v and the negative (-) pin to GND.
2. OLED: Connect SDA (serial data) to A4, SCL/SCK (serial clock) to A5, VDD/VCC (supply voltage) to +5v and GND to GND.
3. Now, you can plug in your Arduino board via the USB cable to the computer.

# Arduino Heart Rate Monitor Code

## Code Segment

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define screen_width 128
#define screen_height 64
#define OLED_RESET 4
Adafruit_SSD1306 display(screen_width, screen_height);

#define samp_siz 4
#define rise_threshold 4

int sensorPin = 0;

void setup() {
Serial.begin(9600);
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
display.clearDisplay();
}

void  loop ()
{
   float reads[samp_siz], sum;
   long int now, ptr;
   float  last, reader, start;
   float first, second, third, before, print_value;
   bool rising;
   int rise_count;
```

```
    int rise_count;
    int n;
    long int last_beat;
    for (int i = 0; i < samp_siz; i++)
      reads[i] = 0;
    sum = 0;
    ptr = 0;
    while(1)
    {
      n = 0;
      start = millis();
      reader = 0.;
      do
      {
        reader += analogRead (sensorPin);
        n++;
        now = millis();
      }
      while (now < start +  20);
      reader /= n;
      sum -= reads[ptr];
      sum += reader;
      reads[ptr] = reader;
      last = sum / samp_siz;

      if (last > before)
      {
        rise_count++;
        if (!rising && rise_count > rise_threshold)
        {
          rising  = true;
          first = millis() - last_beat;
          last_beat = millis();
          print_value = 60000. / (0.4 * first + 0.3 *  second + 0.3

          Serial.print(print_value);
```

```
            Serial.print(print_value);
            Serial.print('\n');

            display.clearDisplay();
            display.setTextSize(1);
            display.setTextColor(SSD1306_WHITE);
            display.setCursor(0, 25);
            display.print("Heart Rate:");
            display.setCursor(70, 25);
            display.print(print_value);
            display.setCursor(110, 25);
            display.print("bpm");
            display.display();

            third = second;
            second  = first;

          }
        }
        else
        {
          rising = false;
          rise_count = 0;
        }

        before = last;
        ptr++;
        ptr  %= samp_siz;

    }
}
```

## Code Explanation

- This code is adapted from Johan Ha's work on calculating an accurate, usable heart rate

reading (in beats per minute) from the analogue output of the HW-502 heartbeat sensor. Due to the sensor construction consisting of a photodiode that is emitting infrared light and a phototransistor detecting that light (that passes through the finger), the data that is fed into the Arduino is an analogue value between 1 to 1023. With this project code, that analogue value is converted from integer values into a viable heart rate reading using some calculations.

- To start with, the first seven lines should be familiar if you have seen previous projects as they simply define libraries and parameters that are used for the OLED display. The next two lines starting with the #define component are involved in the process of translating the analogue value from the sensor to a heart rate reading by firstly defining a sample size of 4 (to average out readings to improve data accuracy) and a rise threshold of 4 to limit the variability between heart rate readings (any outliers or errors). The pin (A0) that the HW-502 sensor is connected to is then defined.
- The void setup section sets up serial communication by declaring a baud rate of 9600 bauds, defining the OLED display parameters and clearing the display before moving on to the void loop section.
- The first part of the void loop is a series of lines declaring various float, integer and bool data types that will be used to store data from the sensor and assist in the calculation of heart rate. Float data types are useful in storing continuous and analogue values which include decimal numbers whereas the int (integer) data type can only store positive or negative whole numbers. On the other hand, the bool (boolean) data type stores either a true or false value.
- Now, we move onto quite a large for loop. The first part of the for loop essentially declares an integer variable which starts at 0, and every time the loop is run, that variable adds one to its value, counting up. It will count up to 4 (as defined by the samp_size line that we have seen earlier) and then reset, and this is how we will generate the average of our output values.
- Moving onto the next do/while loop, it essentially commands the Arduino to read and add the analogue values coming out of the sensor (analogRead) and store it to the float variable called ‘reader’. At the same time, the float variables called ‘start’ and ‘now’ are used alongside the millis() function to essentially start a stopwatch to mark different parts of the loop sequence which will be referred to later.
- For the next while loop, the average over four data points is now calculated and we

implement ways to refine the data values for the most accurate readings. The value from the float variable ‘reader’, which was used to store our incoming data from the sensor, is first divided (/=) over the sample size (of 4). Then, we minus the oldest reading from the sample size from the sum of the values, followed by adding the newest reading that we collect to the sum of the values. The new data value is then saved in the float variable ‘reader’ once again and a brand new average is calculated and saved to the float variable ‘last’.

- The subsequent if statements serve the purpose of detecting a pattern in the curve generated by the serial plotter. By using a rise_threshold constant and counting the number of times the curve crosses over that threshold value, the Arduino will know what point of the curve it is currently at since there is a constant sine wave pattern to the curve. This then helps us to know when a finger is placed on the sensor and a heartbeat is detected, rather than any other patterns generated from external noise, for example. If the threshold value is not crossed four times in a row, the sensor essentially is not detecting a heartbeat and the Arduino is programmed to reset this procedure and scan the pattern again until it picks up that heartbeat.
- The last few sections of the code essentially print the filtered values calculated to the serial monitor (for debugging purposes) and the OLED display using the same SSD1306 library functions as seen in previous projects.

# Summary of Heart Rate Tracker Projects

A project like this demonstrates a very practical application for the Arduino in combination with the HW-502 heartbeat sensor where a portable heart rate monitor can be made. Once again, TechSparks reiterates that this project should not be used as a replacement for industrial-grade medical equipment which offers considerably higher reliability and accuracy of results.

However, although the hardware setup was quite simple, the code involved more intermediate concepts that focused on not only reading out analogue values from the sensor but instead going a step further and refining the data to produce a usable, heart rate reading in beats per minute. The code involved several float variables that were used to store data, in addition to several do/while loops, if statements and mathematical equations to calculate an average from a pre-defined sample size. There were also lines of code implemented to detect a heartbeat trace based on when

values crossed a specific threshold. These are all important concepts that can be picked up from this project and applied to many other data collection projects where values need to be accurately processed in some way.

With that said, this project can definitely be improved upon by potentially displaying the heart rate curve trace onto the OLED, building a standalone unit with custom PCBA (alongside a custom case), or simply adding more sensors to record other health metrics.