COVID-19 Ventilator and Health Monitoring Device

Ashwini Kumar Sinha

May 7, 2021

At this time of COVID-19 outbreak, the demand for ventilator and health monitoring devices has increased with each passing day. To overcome the shortage of these devices, we don't have any other good solution yet that can help us in fulfilling this requirement. we can try to design own RaspberryPi COVID -19 Ventilator.

So, today we have decided to make a small ventilator prototype using Raspberry Pi. It will also be capable of monitoring our health and provide data about our heartbeat and SPO2 levels.

Our ventilator prototype uses a servo motor that applies pressure on an air sack (BVM bag), thus pushing oxygen-concentrated air into the lungs. When the servo motor comes back to its earlier position, it results in pressure being released from the air sack (BVM bag), making it retain its original shape. This helps to draw out CO2 from the lungs (similar to the process of breathing in and out). The entire ventilator mechanism of respiration should be in sync with a patient's normal respiratory rate. This can be achieved by changing the speed of the servo motor in the program. We have also used the MAX30100 sensor that gives us live data about the rise and fall of pulse rate and oxygen level in the blood of a patient. By implementing Raspberry Pi and any standard LCD display, we can observe the pulse rate and blood oxygen percentage as a graph on the display screen. (Refer Fig 2,3,4,5)

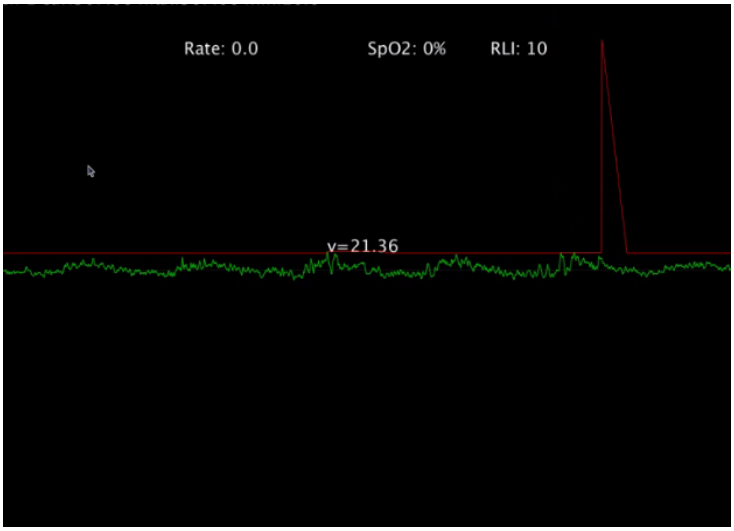


Fig 2. No beat

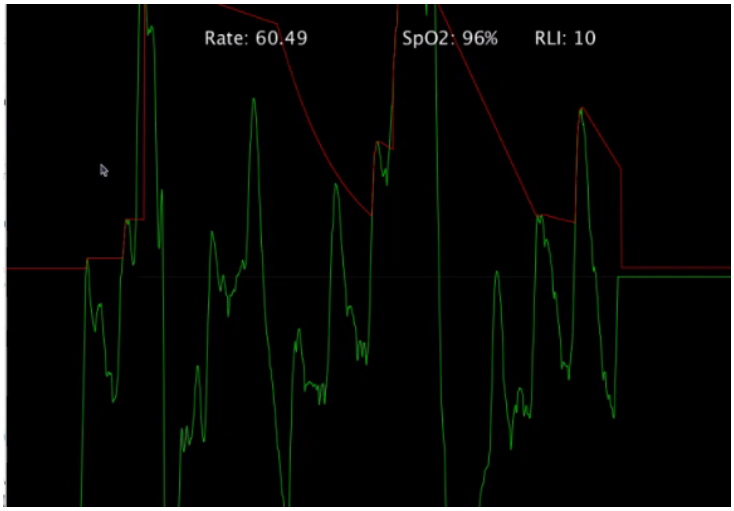


Fig 3. Showing Beat Rate

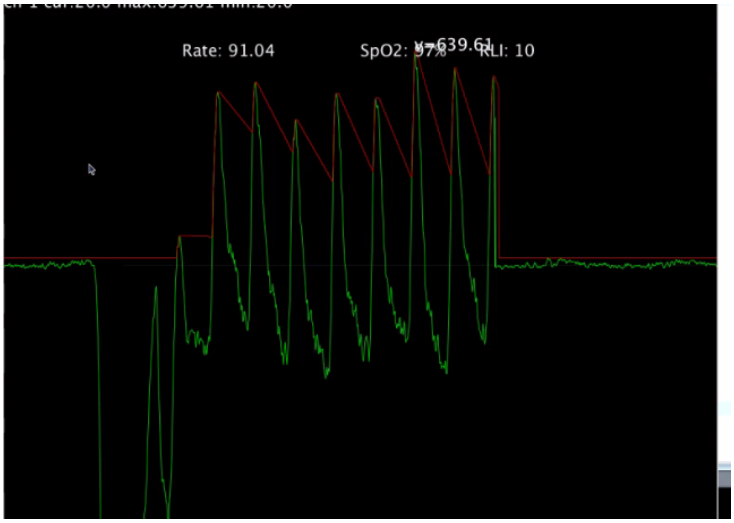


Fig 4.

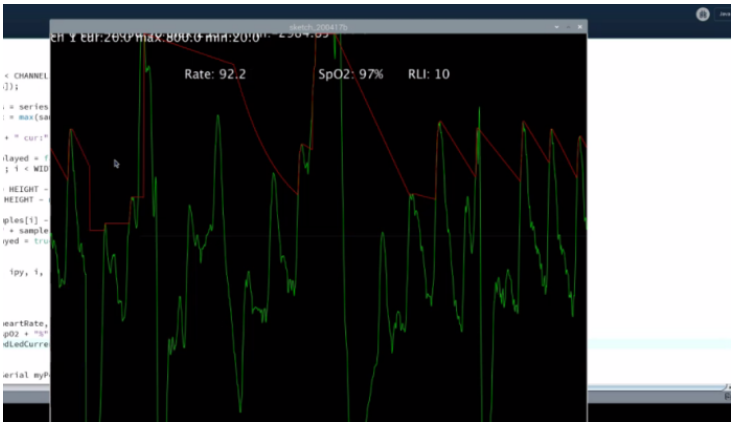


Fig 5.

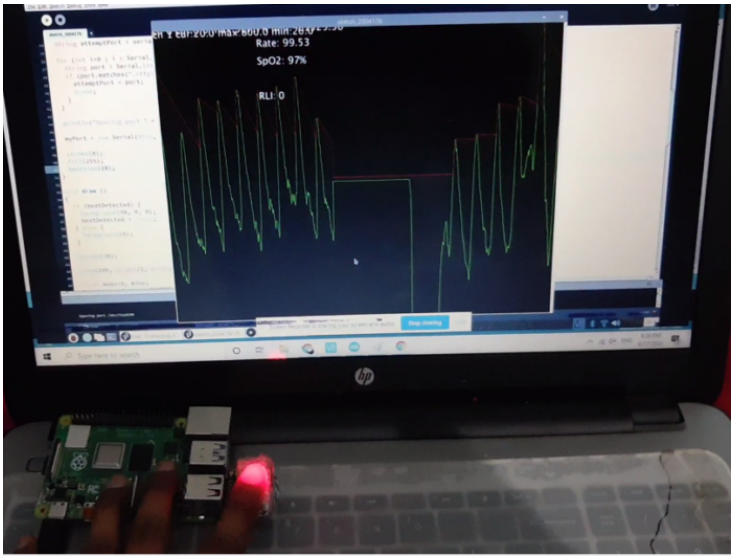


Fig 6.

So let's start our project by collecting the following components for RaspberryPi COVID -19 Ventilator:

Bill Of Material

Component Name	Quantity	Description	Cost Approx. In INR
Raspberry Pi 3/4,Zero	1	Any Model Of Raspberry Pi	1500
BVM BAG	1	Any Standard BVM BAG	1000
MAX30100	1	Sensor	300
Test Lung	1	Standard Test Lung	800
LCD Display	1	RCA or HDMI	1000
Servo Motor	1	2KG/cm Torque	300
Wires	depends		30
Arduino Pro Micro / any Other version	1		200
SSD1306(Optional)	1	OLED Display	300
Total Cost			**5430**

We also need extra cardboard and pipe for mechanical construction.

NOTE:-Here, I am using a balloon instead of a test lung and a BVM bag. But for good results, please use a BVM bag and standard test lung. You will also need to make some mechanical changes to the BVM bag for it to work properly.

Do keep in mind that this DIY project should be implemented only under strict medical supervision.

Now let's begin our project with some mechanical arrangement and construction.

Construction of ventilator

Here, we will take any cardboard and fix the BVM bag on its flat surface. (Refer Fig 7). Now we will take one end of a pipe and insert that into the opening of the BVM bag (here I have used a

flattened balloon). The other end of the pipe will be attached to the standard test lung (here also I have used a balloon). Now we will place another cardboard piece on top of the BVM bag so that one side of the BVM bag is fixed to one end of the cardboard and its other side with a servo motor shaft.(Refer Fig 7,8,9,10).

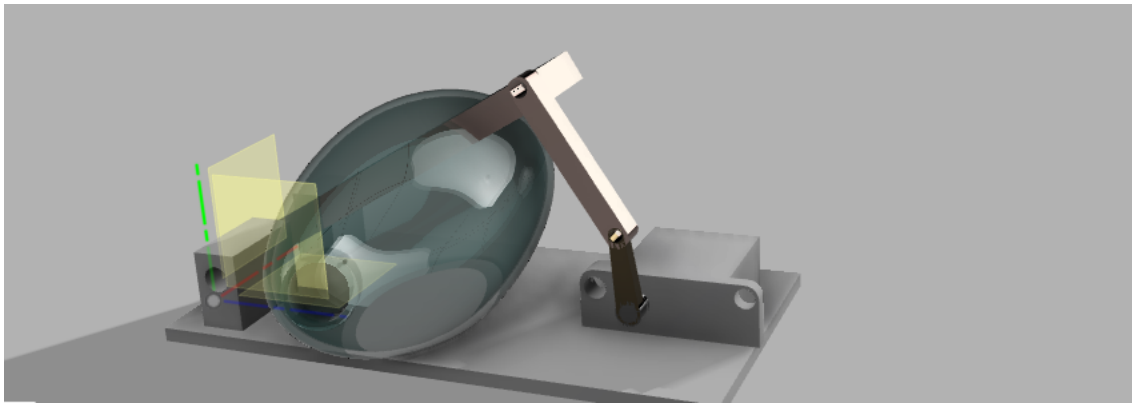


Fig 7

Fig 8.

Fig 8

Fig 9

Fig 10

After the mechanical construction, now let's go for the electronic construction and codingConnect the RPi with the servo motor in the following way:-

RPI	SERVO
GND	GND
5V	5V
GPIO 17 (Gpio zero pin number)	SIGNAL PIN (Yellow Wire)

The Health Monitoring System

For coding of the health monitoring system, open the Arduino IDE, go to Library manager and install the following required libraries

- SSD1306 Chart
- Max30100

After their successful installation, we can start coding.

First, we will initialize the libraries in the code and create two variables, val1 and val2.

Next, we will create a setup function that will start the OLED display and the MAX30100 sensor. It will also set the graph length and breadth.

Then we will create a loop function that checks data from the MAX sensor, displays it on the OLED and also sends that to the serial port.

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS     100
#include <Arduino.h>
#include <OLED_SSD1306_Chart.h>
float val1=0;
float val2=0;
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)

OLED_SSD1306_Chart display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

PulseOximeter pox;

uint32_t tsLastReport = 0;
```

Fig 11.

```
void setup()
{
    Serial.begin(115200);

    if (!pox.begin(PULSEOXIMETER_DEBUGGINGMODE_PULSEDETECT)) {
        Serial.println("ERROR: Failed to initialize pulse oximeter");
        for(;;);
    }
     Wire.begin();
   display.begin(SSD1306_SWITCHCAPVCC, 0x3c);
  display.clearDisplay();
  display.setChartCoordinates(0, 60);         //Chart lower left coordinates (X, Y)
  display.setChartWidthAndHeight(123, 60);    //Chart width = 123 and height = 60
  display.setXIncrement(5);                   //Distance between Y points will be 5px
  display.setYLimits(0, 100);                 //Ymin = 0 and Ymax = 100 for first chart
  display.setYLimits(0, 100, 1);              //Ymin = 0 and Ymax = 100 for second chart
  display.setPointGeometry(POINT_GEOMETRY_CIRCLE, 1); //Points will be circles for second chart
  display.setAxisDivisionsInc(12, 6);         //Each 12 px a division will be painted in X axis and each 6px inY axis
  display.setPlotMode(DOUBLE_PLOT_MODE);      //Set double plot mode
  display.drawChart();                        //Update the buffer to draw the cartesian chart
  display.display();
    // The default current for the IR LED is 50mA and it could be changed
    //   by uncommenting the following line. Check MAX30100_Registers.h for all the
    //   available options.
    // pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
```

Fig 12

```
void loop()
{
    // Make sure to call update as fast as possible
    pox.update();
     val1=pox.getHeartRate();
    val2=pox.getSpO2();
    // Asynchronously dump heart rate and oxidation levels to the serial
    // For both, a value of 0 means "invalid"
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        Serial.print("H:");
        Serial.println(pox.getHeartRate());

        Serial.print("O:");
        Serial.println(pox.getSpO2());

        tsLastReport = millis();
    }
      if(!display.updateChart(val1, val2))   //Value between Ymin and Ymax will be added to chart
      {
        display.clearDisplay();             //If chart is full, it is drawn again
        display.drawChart();
      }
}
```

Fig 13

Next, connect the components as shown in the circuit diagram.

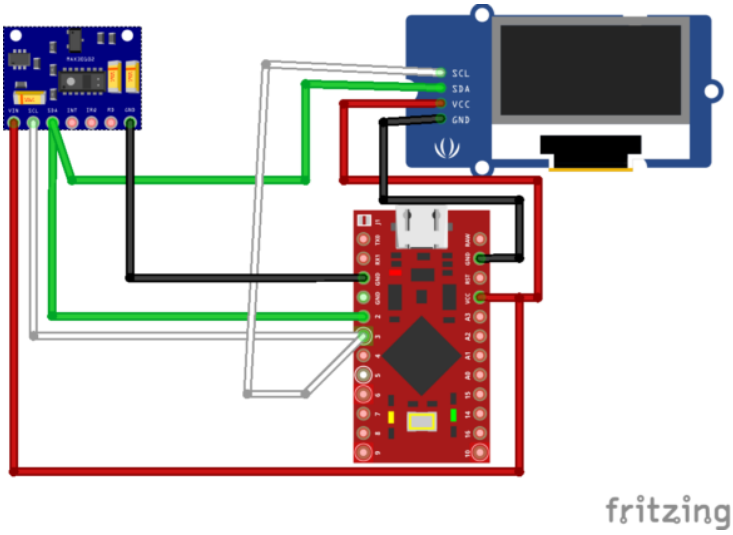


Fig 14.Connection

Now for visualizing the live graph and health data, we will setup the Processing 3 in Raspberry Pi. To do so, open the Raspberry Pi and run the given system in terminal.

curl https://processing.org/download/install-arm.sh | sudo sh

Next, open the Processing 3 (Refer Fig 15). Get the code named ”rolling graph.pde” from the Arduino Library extras folder and then paste that in the Processing IDE in Raspberry Pi.

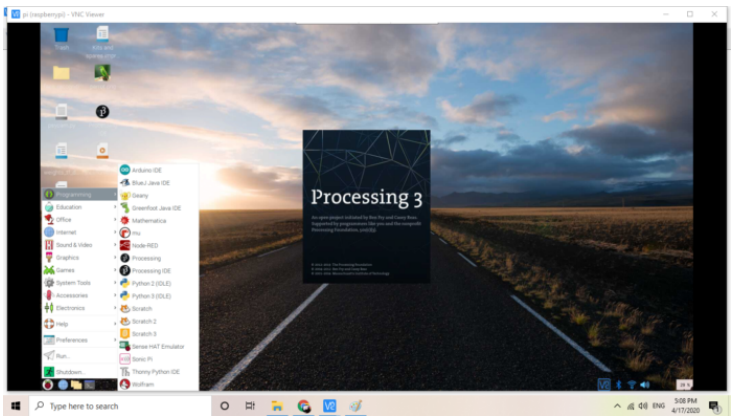
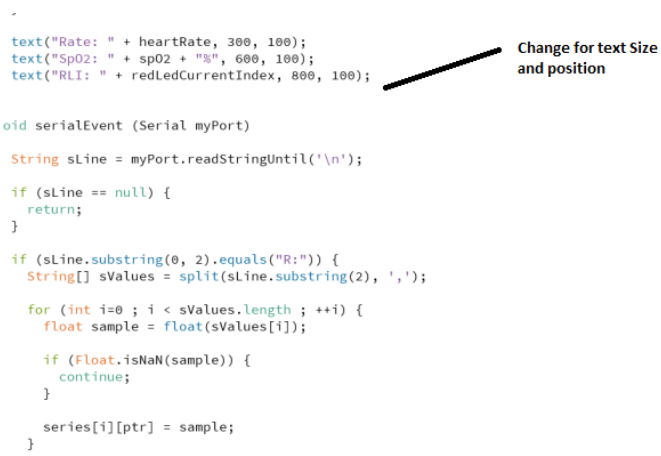


Fig 15.

Fig 16. Change the port name in code




```
text("Rate: " + heartRate, 300, 100);
text("SpO2: " + spO2 + "%", 600, 100);
text("RLI: " + redLedCurrentIndex, 800, 100);

oid serialEvent (Serial myPort)

 String sLine = myPort.readStringUntil('\n');

 if (sLine == null) {
   return;
 }

 if (sLine.substring(0, 2).equals("R:")) {
   String[] sValues = split(sLine.substring(2), ',');

   for (int i=0 ; i < sValues.length ; ++i) {
     float sample = float(sValues[i]);

     if (Float.isNaN(sample)) {
       continue;
     }

     series[i][ptr] = sample;
   }
```

Fig 17.

Testing

Now power the Raspberry Pi and then connect it with the display. Run the code for ventilator and then run the code for health data i.e. “rollinggraph.pde” in the Raspberry Pi Processing. Also, connect the Arduino with MAX30100 into the Raspberry Pi USB port.

```
from gpiozero import Servo
from gpiozero.tools import sin_values

servo = Servo(17)

servo.source = sin_values()
```

Fig 18.the code to move servo and press BVM bag of ventilator

When the code for the ventilator runs, the servo motor shaft starts moving and results in pressure being added on the BVM BAG. Hence, oxygen goes through the pipe and flattens the lung. Then the servo motor comes back to its original position, restoring the BVM bag to its previous stae. This continuous process simulates the contraction (breathing in) and relaxation (breathing out) of a human lung.

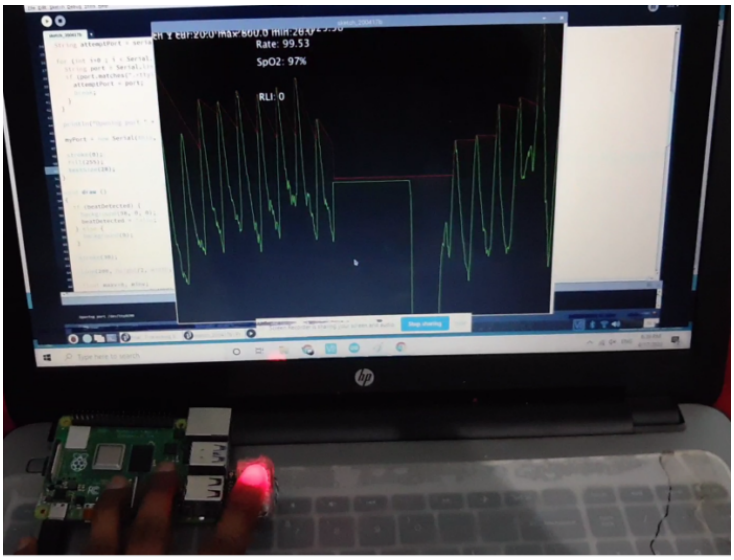
For health data visualization, simply put your finger on the MAX sensor. As soon as it will start to glow, the live graph of the data health will be obtained.

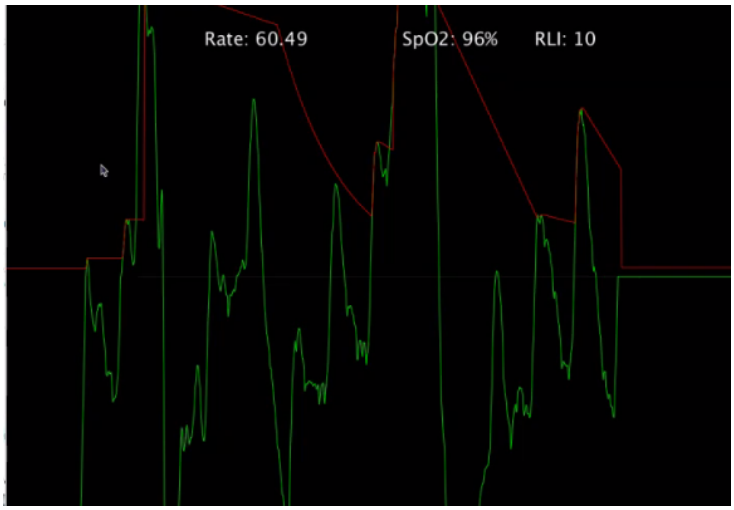
NOTE:- To change the breathing rate of the ventilator, add the time delay in the code .

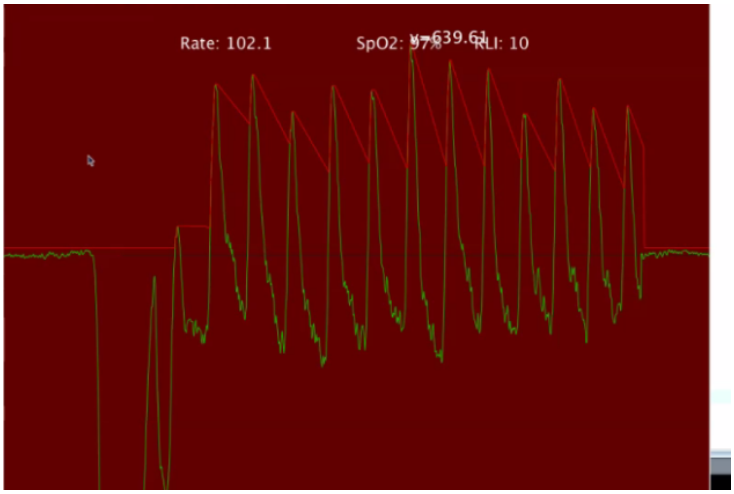
```
servo.source = sin_values()
servo.source_delay = 0.1 #change this according to breadth rate so servo move as per requirement
```

Also for health data monitoring, change the port name to the port name of Arduino. Here my Arduino port name is ttyACM0.]

Congrats, Our prototype of RaspberryPi COVID -19 Ventilator is ready.







Download Source Code

This article was first published on 24 April 2020 and was updated on 7 May 2021.