Social Distancing Sheild for COVID-19 – 2.0

Modeling and Designing

Manthan Pawar
New York University,
Tandon School of Engineering
Brooklyn, New York
mvp321@nyu.edu

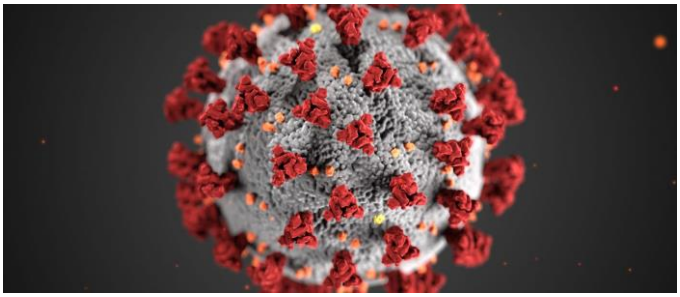
Zewen Wu
New York University
Tandon School of Engineering
Brooklyn, New York
zw2420@nyu.edu

***Abstract*— Novel Coronavirus disease 2019, known as COVID-19 hit the humanity om global scale and was announced as a global pandemic by World Health Organization (WHO). The coronavirus is thought to spread mainly from person to person. This can happen between people who are in close contact with one another. Droplets that are produced when an infected person coughs or sneezes may land in the mouths or noses of people who are nearby, or possibly be inhaled into their lungs. Understanding how the virus spreads reinforces the importance of social distancing and other health-promoting behaviors. Because COVID-19 spreads from person to person, reducing the ways people come in close contact with each other is essential. According to Centers for Disease Control and Prevention (CDC) guidelines, to practice social or physical distancing, you should stay at least 6 feet (2 meters) from other people. In these dark times, everyone needs wants to get out, and the key to start the economy back again is keeping those social distancing norms and work simultaneously. Therefore, we are proposing a prototype of a device which will help people keeping these social distancing norms. The device will notify the user if there is someone inside the 6ft radius from the person wearing it. That way the user can know that he/she needs to get distant from someone. The device is cap which can convert itself into a face shield, one of the most important Personal Protective Equipment (PPE) used against the fight of coronavirus, when a person is detected inside the radius of 6 ft from the person wearing it.**

Keywords— COVID-19, Social Distancing, Centers for Disease Control and Prevention (CDC), Pandemic, Restart Economy, Personal Protective Equipment (PPE), Face Sheild, Cap.

I. INTRODUCTION

Coronaviruses are an extremely common cause of colds and other upper respiratory infections. COVID-19, short for "coronavirus disease 2019," is the official name given by the World Health Organization to the disease caused by this newly identified coronavirus. People of any age should take preventive health measures like frequent hand washing, physical distancing, and wearing a mask when going out in public, to help protect themselves and to reduce the chances of spreading the infection to others. Recently published research found that on average, the time from exposure to symptom onset (known as the incubation period) is about five to six days. However, studies have shown that symptoms could appear as soon as three days after exposure to as long as 13 days later. These findings continue to support the CDC recommendation of self-quarantine and monitoring of symptoms for 14 days post exposure. The coronavirus is thought to spread mainly from person to person. This can happen between people who are in close contact with one another. Droplets that are produced when an infected person coughs or sneezes may land in the mouths or noses of people who are nearby, or possibly be inhaled into their lungs.



A person infected with coronavirus — even one with no symptoms — may emit aerosols when they talk or breathe. Aerosols are infectious viral particles that can float or drift

around in the air for up to three hours. Another person can breathe in these aerosols and become infected with the coronavirus. Therefore, everyone should cover their nose and mouth when they go out in public.
Coronavirus can also spread from contact with infected surfaces or objects. For example, a person can get COVID-19 by touching a surface or object that has the virus on it and then touching their own mouth, nose, or possibly their eyes.

Need of Face Shield for everyone:

Study done by National Institute of Allergy and Infectious Diseases' Laboratory of Virology in the Division of Intramural Research in Hamilton; Montana helps to answer this question. The researchers used a nebulizer to blow coronaviruses into the air. They found that infectious viruses could remain in the air for up to three hours. The results of the study were published in the *New England Journal of Medicine* on March 17, 2020.

Face shield offer more effective protection against coronavirus than masks and should be worn by the public whenever they leave home, according to US physician and epidemiologist Michael Edmond.
Edmond, an infectious diseases physician and hospital epidemiologist in Iowa City, USA, has been advocating the use of face shields on his blog.

He believes the simple devices are more effective than masks at protecting the eyes, nose and mouth from Covid-19 infection and praised efforts by architects and designers to manufacture the devices as "one of the silver linings of this pandemic". According to him, every person should have the face shield.

II. Proposed Solution

Our main idea is a sun cap whose shade converts into a face shield when it detects a presence of human within 6ft radius from the user and notifies the user. The device has two main components, the Shield Cap and the remote. We are using 3 microcontrollers, details of which will be discussed further.

The Remote:
The remote comprises of an Arduino Uno, a few interactive components such as LEDs and switches to control the shield cap.

Shield Cap:
The cap comprises of Arduino Nano and Raspberry Pi, shield actuator, Pi cam, PIR and some other components.

The remote controls the Shield cap. Switch on switch off commands and Shield calibration command can be given through the remote to the cap. The device detects if there is someone within 6ft behind the user. If there is someone within 6ft, the device flaps the shield down and warns the user about the presence of someone 6ft behind him/her. The device has a Pi cam and Passive Infrared Sensor (PIR Sensor). Both detect the human motion. Sometimes Pi cam detects unwanted motion. Therefore, we are using data from both the sensors to decide if the system. ether it was human motion or not. The data from both the sensors is used in AND relation so that it increases the reliability of the system. Once the motion is detected, the Pi cam detected actual distance of the subject from the user is used to decide whether the distance is safe of no. The vibration motor in the pocket and the LED on the cap tells the user that the distance is lower than 6 ft. Based on the intensity of vibration and LED brightness, criticality of the distance is represented. The LCD shows the information of motion detection and the distance. A servo motor is used to flap down the shield. Also, the user has a switch in his hand which he can use to directly trigger the shield. A potentiometer is used to Set up the user-defined rotation angle of servo motor of the shield. Currently we are implementing only the sensing of human from behind as lack of resources. Our actual proposal for the final project is that we will have a servo-actuated platform over which he Pi cam will be mounted that continuously rotated and scans for motion in 360 degree.

Bill of Material of Major Components:

Components	**Price**
Arduino UNO	12.99

Arduino NANO	7
Raspberry Pi	61.7
Pi CAM	8.77
Passive Infrared Sensor (PIR Sensor)	9.95
Servo Motor	12.95
Potentiometer	0
Radio Transmitter nRF24L	6.69
HC05	8.99
Switches	0
LEDs	0
Cap	9.99
Breadboard	4.95
Total	**143.98**

III. HARWARE PROTOTYPE

Following components are used in the prototype:

The Remote:

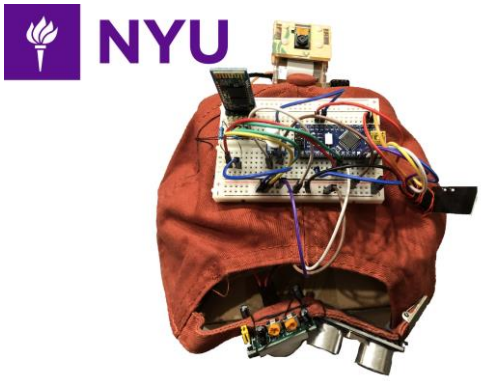
1. Arduino UNO
2. Switches
3. LEDs
4. Radio Transmitter nRF24L

The remote controller is based on an Arduino UNO. It is the only interface between the device and the user. There are two switches, button A and button B. Buttons A and B control the functionality of the device as mentioned in the User Interface and Control section. Button B is the one that can terminate the device after the current progress and back to the initial state of the device. Green LED on the remote indicates the time that user can set the rotation angle of the shield depending on personal preference. These two LEDs can simply be replaced by a two-color LED for the future development. RGB LED represents the state once the device starts working. Theoretically, this LED can be removed, as for the final product, it is unnecessary, while it is able to reflect if the communication between two microcontrollers has been set up properly.

Radio Transmitter nRF24L is used on the remote side to communicate between the Arduino UNO of the remote controller and Arduino nano of the Cap Module 1 where the second Radio Transmitter nRF24L module is used.
Depending on which button and when the button is pressed, the UNO send command to NANO to turn the system on/off, go into calibration mode so on and so forth.

Cap Module 1:

1. Arduino NANO
2. Servo Motor
3. Potentiometer
4. Radio Transmitter nRF24L
5. HC05

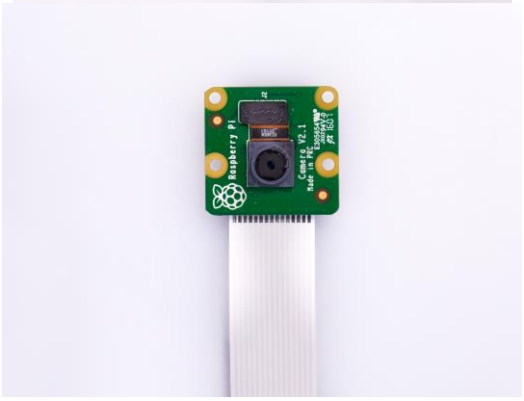


Arduino NANO on the Cap module acts as a slave which takes command from the Raspberry Pi and Arduino Uno. The solution to the communication between Arduino Uno and Arduino Nano has been introduced in the previous section. For communication between NANO and Raspberry Pi, we have used HC-05 Bluetooth module. According to the command from Raspberry Pi, NANO triggers the servo which actuates the shield and then send the current state back to Arduino UNO to control the LED indicator.
Also, according to the command from UNO, NANO starts performing the calibration process. For calibration we are using a potentiometer to manipulate the value of angle between the shield and horizonal. Which means the shield has user defined actuation for the flap.

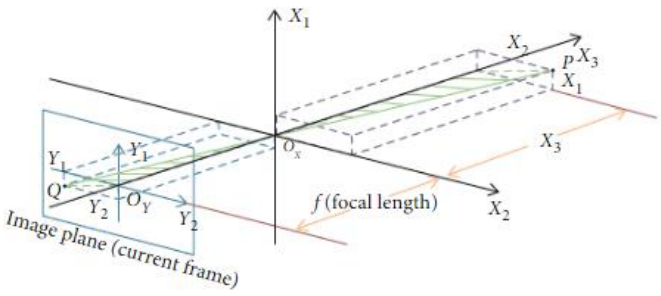
Cap Module 2:

1. Raspberry Pi
2. Pi CAM
3. Passive Infrared Sensor (PIR Sensor)

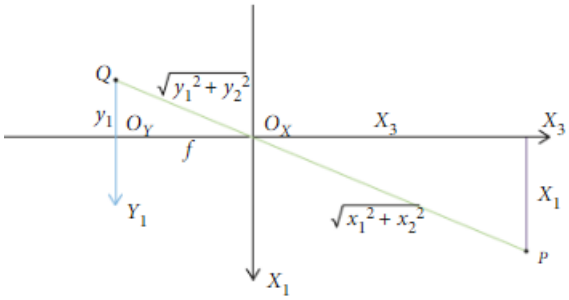
Raspberry Pi, which is integrate with a Pi Camera, is the master device giving the controlling commands to NANO. Pi Camera detects the human motion using image processing. Since there is no RGB-D camera designed for the current prototype, the distance measuring algorithm adapts pinhole camera model, which allows Pi camera to estimate the distance of the human from the camera. However, machine learning what we used is not accurate enough. Therefore, we decided to implement PIR sensor along with the Pi Camera. Date from both used in manipulating the decision.



Pinhole Camera Model



(a)



$$object\ distance = focal\ length \times physical\ size \div measured\ size$$

The triangle similarity goes something like this: Let's say we have a marker or object with a known width *W*. We then place this marker some distance *D* from our camera. We take a picture of our object using our camera and then measure the apparent width in pixels *P*. This allows us to derive the perceived focal length *F* of our camera:

$F = (P \times D) / W$

For example, let's say we place a standard piece of *8.5 x 11in* piece of paper (horizontally; $W = 11$) $D = 24$ *inches* in front of my camera and take a photo. When we measure the width of the piece of paper in the image, we notice that the perceived width of the paper is $P = 248$ *pixels*.

My focal length *F* is then:

$F = (248px \times 24in) / 11in = 543.45$

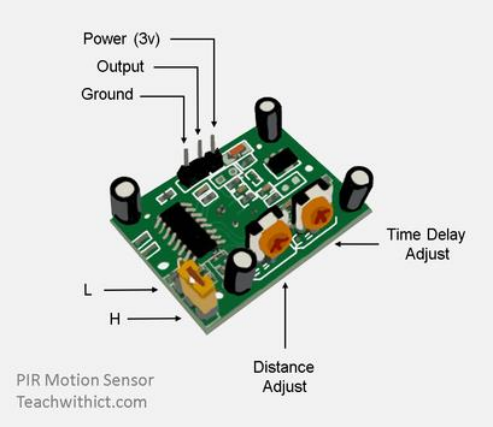
As we continue to move my camera both closer and farther away from the object/marker, we can apply the triangle similarity to determine the distance of the object to the camera:

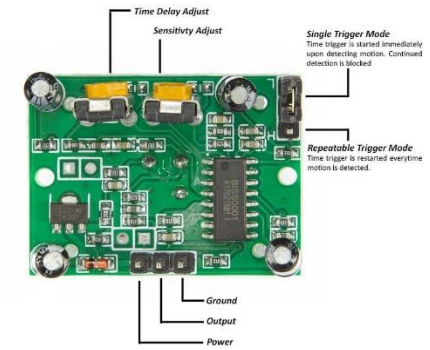
$D' = (W \times F) / P$

Again, to make this more concrete, let's say we move my camera *3 ft* (or 36 inches) away from my marker and take a photo of the same piece of paper. Through automatic image processing we are able to determine that the perceived width of the piece of paper is now *170 pixels*. Plugging this into the equation we now get:

D' = (11in x 543.45) / 170 = 35in
Or roughly 36 inches, which is 3 feet.

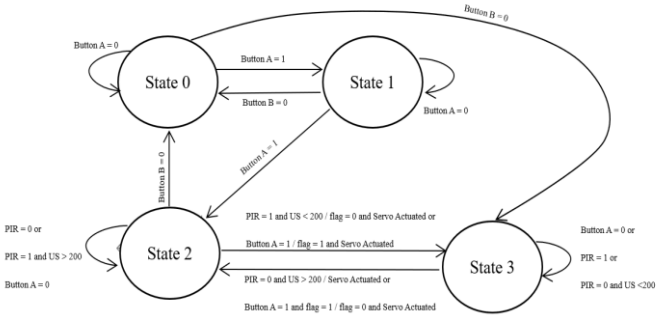
About PIR sensor:
A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. PIR sensors are commonly used in security alarms and automatic lighting applications.
PIR sensors detect general movement, but do not give information on who or what moved. For that purpose, an active IR sensor is required.





IV. User Interface and Control

At the current stage, there is not a communication constructed between the prototype and mobile device, which means all components have to be assembled on the cap. In this case, to optimize the usability, User Interface has to be designed as simple as possible. Therefore, only two buttons and a rotary encoder has been designed for users to activate the system and adjust the setting. The overall procedure of user interface can be specified as: a) turn on the power; b) press button A (green LED will be on, which indicates that the user can start set the rotation angle of the mask based on personal preferences); c) rotate the rotary encoder to adjust the setting (the default rotation angle is 90 degree); d) press button A to activate the system; e) press button B so that the system will return to its initial state (Green LED will be off and button B can be pressed anytime during the progress). Here, referring to the first figure in Section IV, button A is the push button on the top while button B is the push button at the bottom.
Other than UI designed for the product, a closed-loop control system has been designed for the prototype. When the system is in progress, it receives signals from button A, ultrasonic sensor and PIR sensor continuously. The bus contains these three signals will be pushed to the controller which is designed to generate the input signal for the servo motor. The position of the servo motor will be integrated to the bus which is the input for the controller. The control system, while it was being implemented can be considered as a finite state machine, referring to the design of digital logic circuit.
To be specific, except procedures that has been introduced above, if button A is either be pushed or a motion within 6 feet has been detected, the mask is expected to be actuated down. Otherwise, the system will remain in the current state. Similar strategy has been applied to state 3. One thing should be noted that, if the state switch is caused by the state of button A, moving back to state 2 can only be triggered by the state of button A, which has been realized by introducing a flag, which is either an input or an output in the state machine.



V. Program

Remote: Arduino UNO Program:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
```

```
#define led_pin 2
#define led_pin_1 3
#define button_pin 5
#define button_pin_1 4
#define red_pin 8
#define green_pin 7
#define blue_pin 6


RF24 radio(9, 10); // CE, CSN
const byte address[][6] = {"00001",
"00002"};     //Byte of array
representing the address. This is the
address where we will send the data. This
should be same on the receiving side.


int count = 0;
int count1 = 0;
boolean button_state = 0;
boolean button_state_1 = 0;
boolean last_button = 0;
boolean last_button_1 = 0;

void setup() {
  pinMode(led_pin, OUTPUT);
  pinMode(led_pin_1, OUTPUT);
  pinMode(blue_pin, OUTPUT);
  pinMode(green_pin, OUTPUT);
  pinMode(red_pin, OUTPUT);
  pinMode(button_pin, INPUT);
  pinMode(button_pin_1, INPUT);
  Serial.begin(9600);
  radio.begin();
//Starting the Wireless communication
  radio.openWritingPipe(address[1]);
//Setting the address at which we will
receive the data
  radio.openReadingPipe(1, address[0]);
  radio.setPALevel(RF24_PA_MIN);  //You
can set it as minimum or maximum
depending on the distance between the
transmitter and receiver.
  Serial.println("test");
}

void loop()
{
  delay(5);
  radio.stopListening();
  button_edge_detection();
  delay(5);
  radio.startListening();
  if(radio.available())
  {
    char test;
    radio.read(&test, sizeof(test));
    Serial.println(test);
    if(test == '0')
    {
      digitalWrite(led_pin, HIGH);
      digitalWrite(led_pin_1, LOW);
      digitalWrite(blue_pin, 0);
      digitalWrite(red_pin, 0);
      digitalWrite(green_pin, 0);
    }
    else if(test == '1')
    {
      digitalWrite(led_pin_1, HIGH);
      digitalWrite(led_pin, LOW);
    }
    else if(test == '2')
    {
      digitalWrite(led_pin_1, LOW);
      digitalWrite(blue_pin, 0);
      digitalWrite (green_pin, 255);
      digitalWrite (red_pin, 0);
    }
    else if(test == '3')
    {
      //digitalWrite(led_pin_1, LOW);
      digitalWrite(blue_pin, 255);
      digitalWrite (red_pin, 255);
      digitalWrite (green_pin, 0);
    }
  }
}


void button_edge_detection()
{
  button_state = digitalRead(button_pin);
  button_state_1 =
digitalRead(button_pin_1);
  //
  if(button_state == HIGH && last_button
== LOW)
  {
    count ++;
  }
  if(button_state_1 == HIGH &&
last_button_1 == LOW)
  {
    count1 ++;
  }

  if (count % 2 !=0 )
  {
    if (count1 % 2 !=0 )
    {
      const char text[] = "11";
      //Serial.println(text);
      radio.write(&text, sizeof(text));
//Sending the message to receiver
```

```
    }
    else
    {
      const char text[] = "10";
      //Serial.println(text);
      radio.write(&text, sizeof(text));
    }
  }
  else
  {
    if (count1 % 2 !=0 )
    {
      const char text[] = "01";
      //Serial.println(text);
      radio.write(&text, sizeof(text));
//Sending the message to receiver
    }
    else
    {
      const char text[] = "00";
      //Serial.println(text);
      radio.write(&text, sizeof(text));
    }                    //Sending the
message to receiver
  }
  last_button = button_state;
  last_button_1 = button_state_1;
    //Sending the message to receiver
}
```

Cap Module 1: Arduino NANO Program:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Servo.h>
#include "SoftwareSerial.h"

#define outputA 2
#define outputB 3
#define Servopin 6
#define potpin 7

SoftwareSerial BTserial(5, 4);
RF24 radio(9, 10); // CE, CSN


Servo myservo;
const byte address[][6] = {"00001",
"00002"};
boolean button_state = 0;
int button_last = 1;
int button_1_last = 1;
int button = 0;
int button_1 = 0;
int button_A = 0;
int button_B = 0;
int state = 0;
int count = 0;
int count1 = 0;
int count2 = 0;
int count3 = 0;
int count4 = 0;
int counter = 0;
int val;
int servo_angle = 90;
int flag = 0;
char c = ' ';
unsigned long previousMillis = 0;

void setup() {
  pinMode (outputA,INPUT);
  pinMode (outputB,INPUT);
  Serial.begin(9600);
  radio.begin();
  //Setting the address at which we will
receive the data
  radio.openWritingPipe(address[0]);
  radio.openReadingPipe(1, address[1]);
  radio.setPALevel(RF24_PA_MIN);
//You can set this as minimum or maximum
depending on the distance between the
transmitter and receiver.
  BTserial.begin(38400);
  Serial.println("BTserial started at
38400");
  myservo.attach(Servopin);
  delay(1000);
}
void loop()
{
  delay(5);
  radio.startListening();
  if (radio.available())
//Looking for the data from two buttons
  {
    serialcommunication();
    char text[2] = "";
//Saving the incoming data
    radio.read(&text, sizeof(text));
//Reading the data
    radio.read(&button_state,
sizeof(button_state));    //Reading the
data
    if(text[0] == '1')
    {
      button = 1;
    }
    else if(text[0] == '0')
    {
      button = 0;
    }
    if(text[1] == '1')
    {
      button_1 = 1;
    }
    else if(text[1] == '0')
    {
```

```
      button_1 = 0;
    }
  }
  if (button != button_last)
  {
    button_A = 1;
    count++;
  }
  else
  {
    button_A = 0;
  }
  if (button_1 != button_1_last)
  {
    myservo.write(0);
    Serial.println("State Change");
    count1 ++;
    button_B = 1;
    state = 0;
    delay(1000);
  }
  else
  {
    button_B = 0;
  }
  button_last = button;
  button_1_last = button_1;
  switch(state){
    case 0: // initial state
    {
      Serial.println("Initial State\n");
      servo_angle = 0;
      myservo.detach();
      if (button_A == 1)
      {
        Serial.println("Set the position
of the mask...\n");
        myservo.attach(Servopin);
        if (count1 == 1)
        {
          count1 --;
          state = 0;
          Serial.println(count1);
        }
        else
        {
          count1 = 10;
          state = 1;
          count = 1;
        }
      }
      Listen(state);
      break;
    }
    case 1:
    {
      Serial.println("State 1 \n");
      if (count == 1)
      {
        val = analogRead(potpin);
// reads the value of the potentiometer
(value between 0 and 1023)
        val = map(val, 0, 1023, 0, 90);
// scale it to use it with the servo
(value between 0 and 180)
        myservo.write(val);
// sets the servo position according to
the scaled value
        delay(15);
        servo_angle = val;
      }
      else
      {
        state = 6;
        myservo.write(0);
      }
      Listen(state);
      break;
    }
    case 2:
    {
      Serial.print("Case 2");
      if (button_A == 1)
      {
        Serial.println("Mask On...");
        myservo.attach(Servopin);
        myservo.write(servo_angle);
        flag = 1;
        state = 4;
      }
      else if(c == '1')
     {
        Serial.println("Person Behind
Detected!");
        myservo.attach(Servopin);
        myservo.write(servo_angle);
        state = 4;
     }
     Listen(state);
     break;
    }
    case 3:
    {
      if (button_A == 1 && c != '1')
      {

        Serial.println("Button pressed to
lift the mask!");
        myservo.attach(Servopin);
        myservo.write(0);
        flag = 0;
        state = 5;
        previousMillis = millis();
      }
      else if(c == '0' && flag == 0)
      {
```

```
        Serial.println("No Person is
Detected Behind!");
        myservo.attach(Servopin);
        myservo.write(0);
        state = 5;
        previousMillis = millis();
      }
      Listen(state);
      break;
    }
    case 4:
    {
      count2 = count2 + 1;
      Serial.println(count2);
      if (count2 > 100)
      {
        state = 3;
        myservo.detach();
        count2 = 0;
      }
      Listen(3);
      break;
    }
    case 5:
    {
      count3 = count3 + 1;
      if (count3 > 100)
      {
        state = 2;
        myservo.detach();
        count3 = 0;
      }
      Listen(2);
      break;
    }
    case 6:
    {
      count4 = count4 + 1;
      if (count4 > 100)
      {
        state = 2;
        myservo.detach();
        count4 = 0;
      }
      Listen(2);
      break;
    }
  }
}


void Listen(int state)
{
  delay(5);
  radio.stopListening();
  Serial.println("state = ");
  Serial.println(state);
  if (state == 0)
  {
    //Serial.println("test = 0");
    const char test = '0';
    radio.write(&test, sizeof(test));
  }
  else if (state == 1)
  {
    const char test = '1';
    //Serial.println("test = 1");
    radio.write(&test, sizeof(test));
  }
  else if (state == 2)
  {
    const char test = '2';
    //Serial.println("test = 1");
    radio.write(&test, sizeof(test));
  }
  else if (state == 3)
  {
    const char test = '3';
    //Serial.println("test = 1");
    radio.write(&test, sizeof(test));
  }
}

void serialcommunication()
{
  if (BTserial.available())
  {
    c = BTserial.read();
  }
  if (c == '1')
  {
    Serial.write("Person Detected\n");
  }
  else
  {
    Serial.write("No Person Detected\n");
  }
}
```

Cap Module 2: Raspberry Pi program:

```
from __future__ import print_function
from imutils.object_detection import
non_max_suppression
from imutils import paths
import cv2
import numpy as np
from imutils.video import VideoStream
import imutils
import time
import sys
import serial
import RPi.GPIO as GPIO

print("Start")
```

```
bluetooth =
serial.Serial("/dev/rfcomm0", 38400)
print("Connected")
bluetooth.flushInput()
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN)
usingPiCamera = True
# Set initial frame size.
frameSize = (320, 240)
cascPath =
"haarcascade_frontalface_default.xml"
faceCascade =
cv2.CascadeClassifier(cascPath)
# Initialize mutithreading the video
stream.
vs = VideoStream(src=0,
usePiCamera=usingPiCamera,
resolution=frameSize,
        framerate=32).start()
# Allow the camera to warm up.
time.sleep(2.0)
timeCheck = time.time()

while True:
    # Get the next frame.
    motion = GPIO.input(24)
    frame = vs.read()

    gray = cv2.cvtColor(frame,
cv2.COLOR_BGR2GRAY)

    faces =
faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )
    if motion == 0 and len(faces)==0:
        print ("No Person Detected")

bluetooth.write(str.encode(str(0)))
    else:
        print ("Person Detected")

bluetooth.write(str.encode(str(1)))
   # else:
    #
bluetooth.write(str.encode(str(0)))

    # Draw a rectangle around the faces
#    for (x, y, w, h) in faces:
#        cv2.rectangle(frame, (x, y),
(x+w, y+h), (0, 255, 0), 2)

    # Display the resulting frame
    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF ==
ord('q'):
        break

# When everything is done, release the
capture
video_capture.release()
cv2.destroyAllWindows()
```


ACKNOWLEDGMENT

We thank professor Kapila and the class of Advanced Mechatronics, NYU Tandon school of Engineering for their guidance and suggestions for the project.

FUTURE SCOPE

We hope to get our hands on some more hardware resources. We propose to have a better hardware prototype and robust algorithm based on image processing to scan the human motion and assist social distancing.

REFERENCES

[1] https://www.health.harvard.edu/diseases-and-conditions/covid-19-basics

[2] https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html

[3] https://www.who.int/health-topics/coronavirus#tab=tab_1

[4] https://coronavirus.jhu.edu/map.html