David Adams 216110104

Home Security System

Requirements

YOUTUBE TUTORIAL

https://youtu.be/jk9Drh5gK_A

use this link to download: https://y2mate.guru/en5/

GitHub Code

https://github.com/Davym8/HomeSecurityProject.git

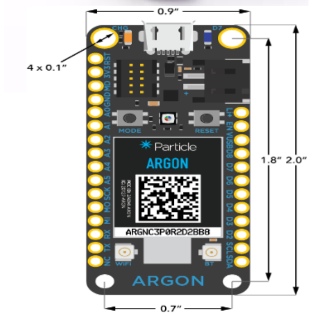
DUE TO THE FILE SIZE LIMIT YOU MUST CREATE YOUR OWN TRAINER AND LABELS, FOLLOW THE GUIDE TO FIGURE OUT HOW.

The requirements for this system are

A Buzzer:

This will send a sound when an unknown person is detected.

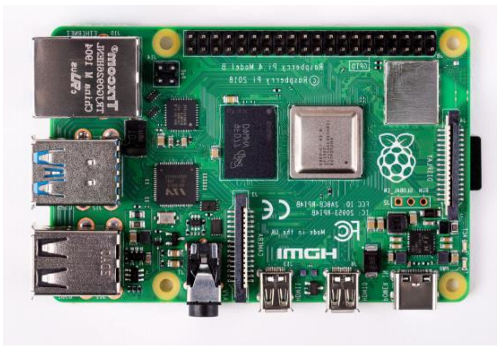
Argon device:



This is a wireless device connected to the cloud.

This will also connect to the Raspberry Pi through I2c and be a slave device.

Raspberry Pi:



This will act as a Master device.

And connect to the Argon device through I2C.

Led lights:

Led's which will be connected through the argon device.

PIR sensor:

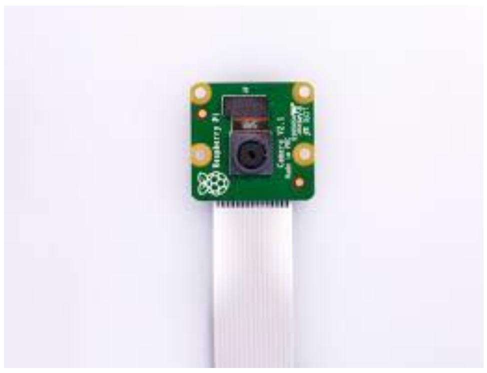
PIR sensor detects objects using infrared light

this will detect humans and initiate the program.

Raspberry Pi camera:

This camera will be connected to the raspberry PI

And will stream video and be used to detect faces.



Jumper Wires:

We need male to male and female to male connecters.

These will connect the sensors and devices together.

Power Cable USB-C:

Requirements for software is to install OpenCV by following this link

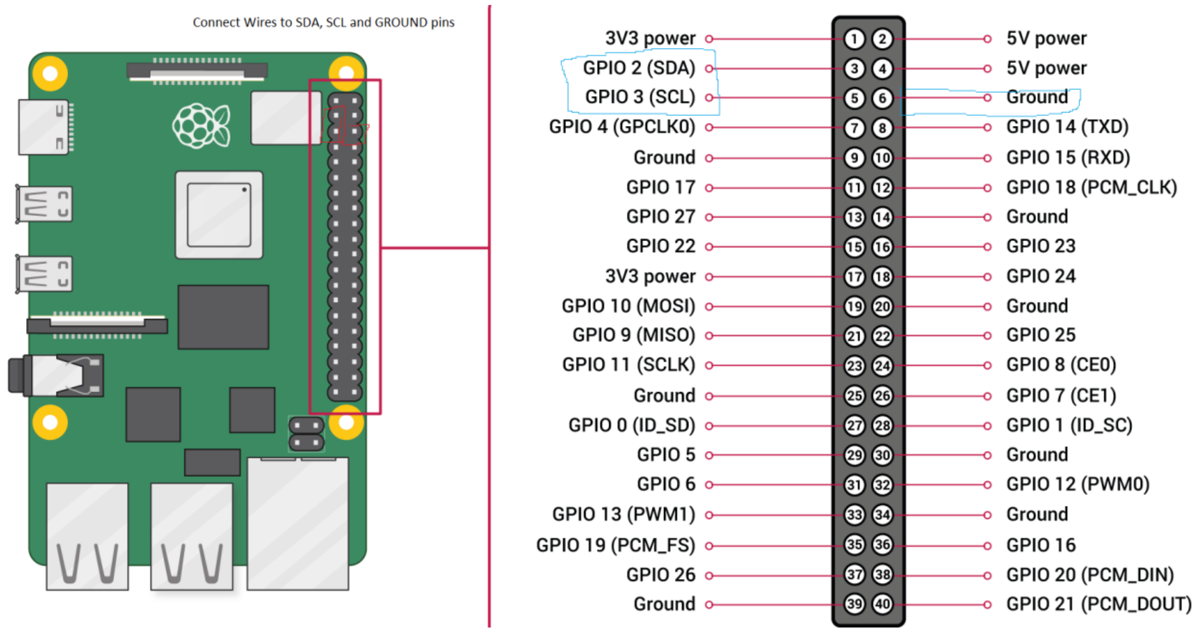
https://pimylifeup.com/raspberry-pi-opencv/

https://www.hackster.io/hackershack/smart-security-camera-90d7bd#toc-code-6

https://linuxize.com/post/how-to-install-opencv-on-raspberry-pi/

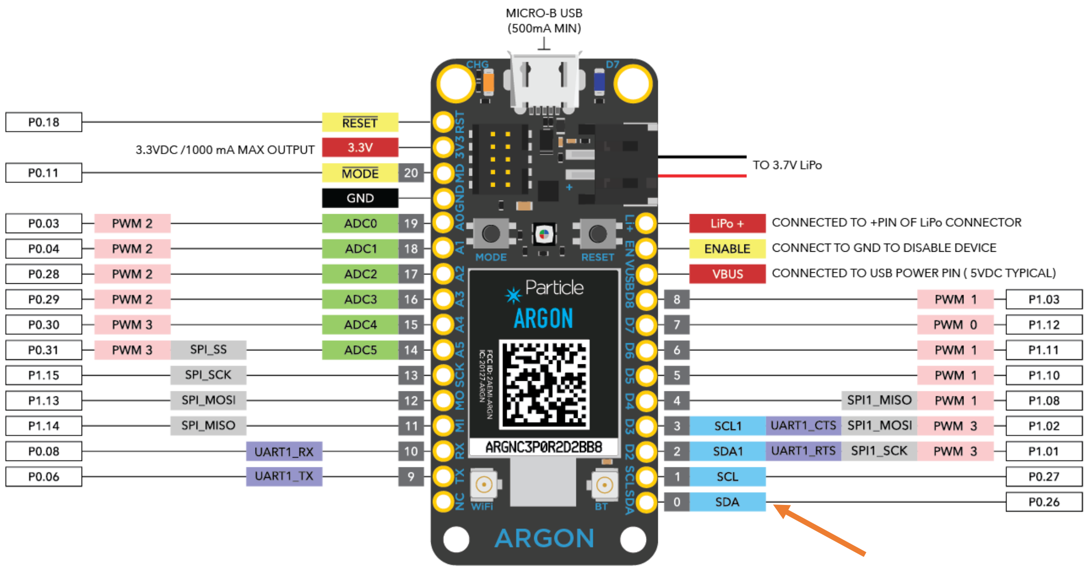
Prototype Architecture

Prototype architecture


Connect Wires to SDA, SCL and GROUND pins
3V3 power 1 2 5V power
GPIO 2 (SDA) 3 4 5V power
GPIO 3 (SCL) 5 6 Ground
GPIO 4 (GPCLK0) 7 8 GPIO 14 (TXD)
Ground 9 10 GPIO 15 (RXD)
GPIO 17 11 12 GPIO 18 (PCM_CLK)
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3V3 power 17 18 GPIO 24
GPIO 10 (MOSI) 19 20 Ground
GPIO 9 (MISO) 21 22 GPIO 25
GPIO 11 (SCLK) 23 24 GPIO 8 (CE0)
Ground 25 26 GPIO 7 (CE1)
GPIO 0 (ID_SD) 27 28 GPIO 1 (ID_SC)
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 (PWM0)
GPIO 13 (PWM1) 33 34 Ground
GPIO 19 (PCM_FS) 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 (PCM_DIN)
Ground 39 40 GPIO 21 (PCM_DOUT)

Connect SDA and SCL cables from Raspberry Pi to the Argon Device

David Adams 216110104

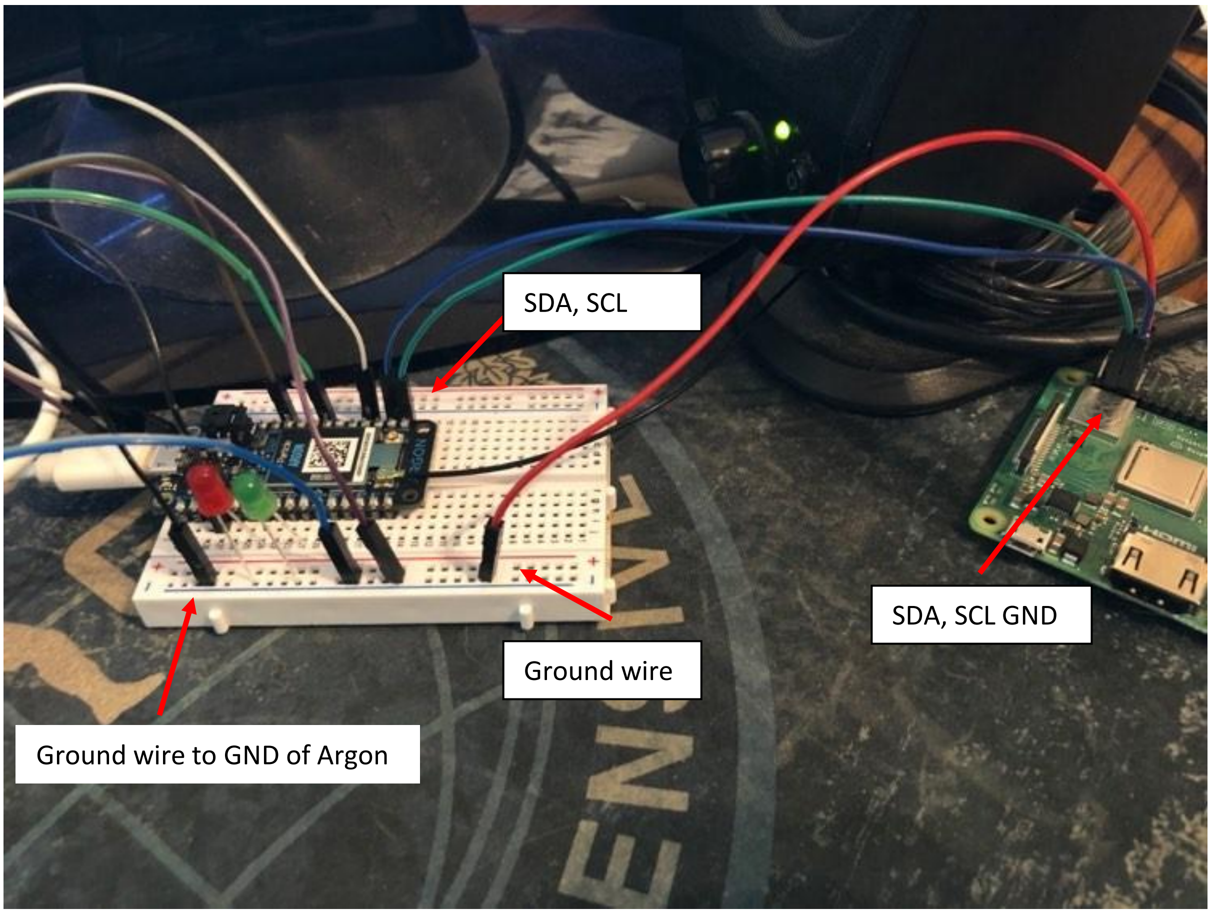


Connect Ground Wire to breadboard int the negative column blue line.

First step: Connecting the two devices

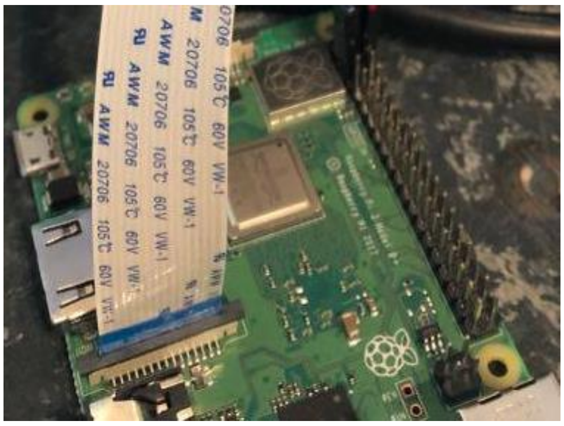
Connect wire from argon GND to the negative blue column

And connect ground cable from raspberry pi to the breadboards negative column as well making a connection.



Second Step: Connect Pi camera on Raspberry Device.

Connect the strip into the port next to the HDMI PORT by pulling the latch up then insert and push down.

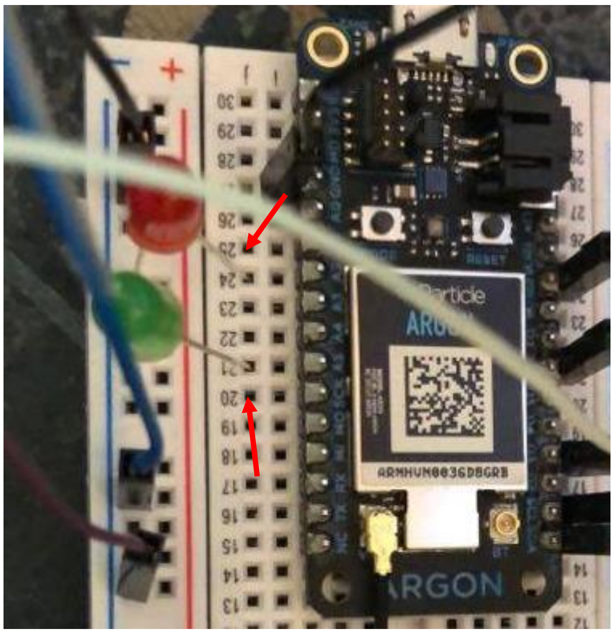


This will allow the camera to work.

Third Step: connect green and red LED’s to argon device

Connect long leg of LED to desired pin. In this case red pin to A2 and green to A5

Then connect the short leg to the negative column so that they are grounded



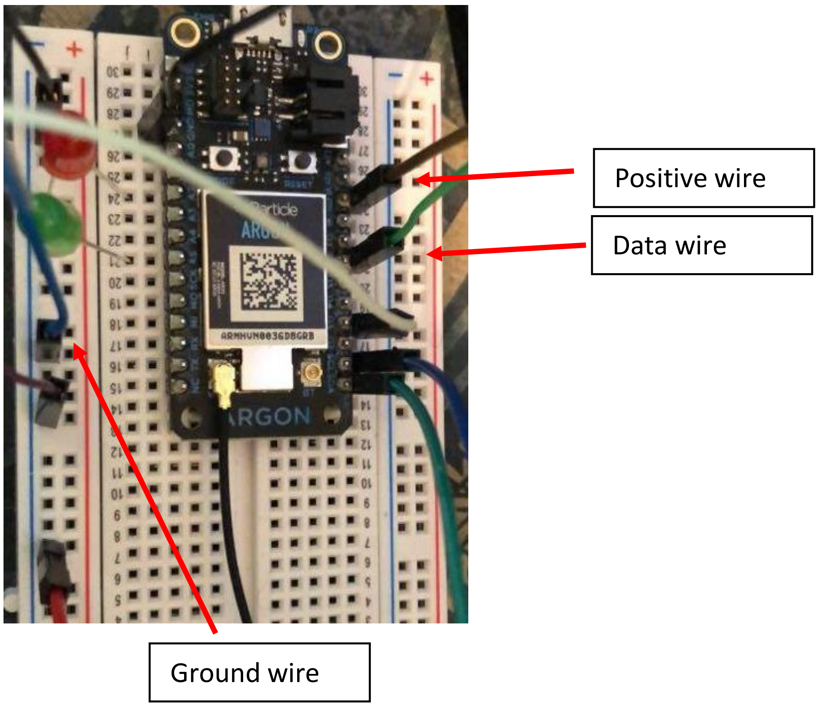
Fourth Step: Connect PIR sensor to Argon

Connect the data pin with cable to the pin you want to use, in this case D6, Green wire.

Connect Positive pin with cable to the VUSB pin on the argon device.

IMPORTANT: MUST BE VBUS as it is 5V required.

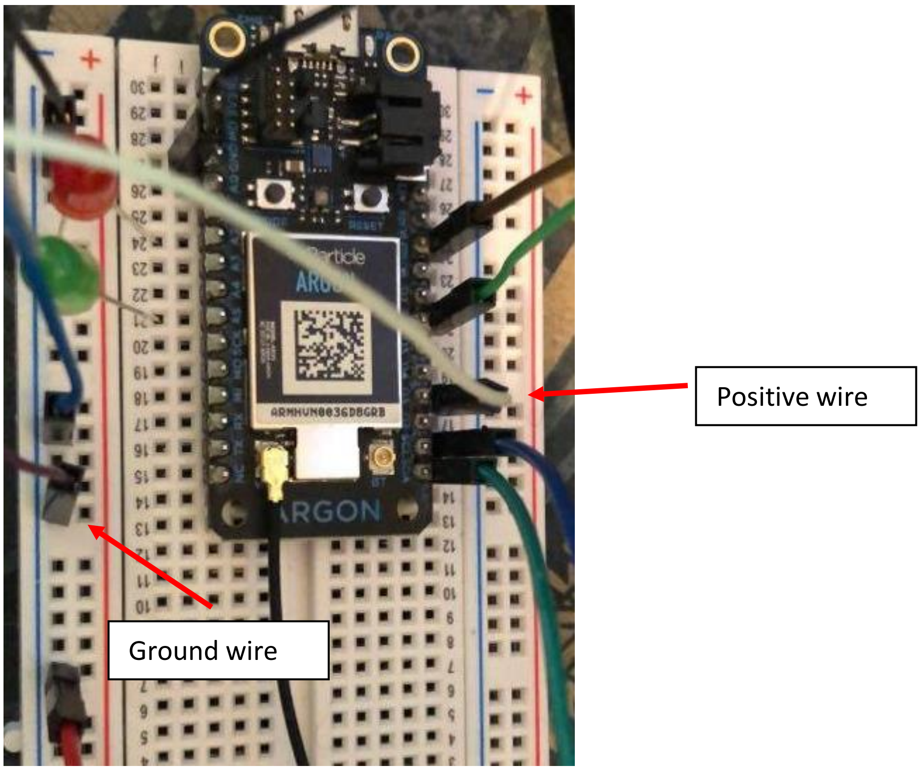
Connect ground wire to the negative blue side column



Fifth Step: Connect Buzzer

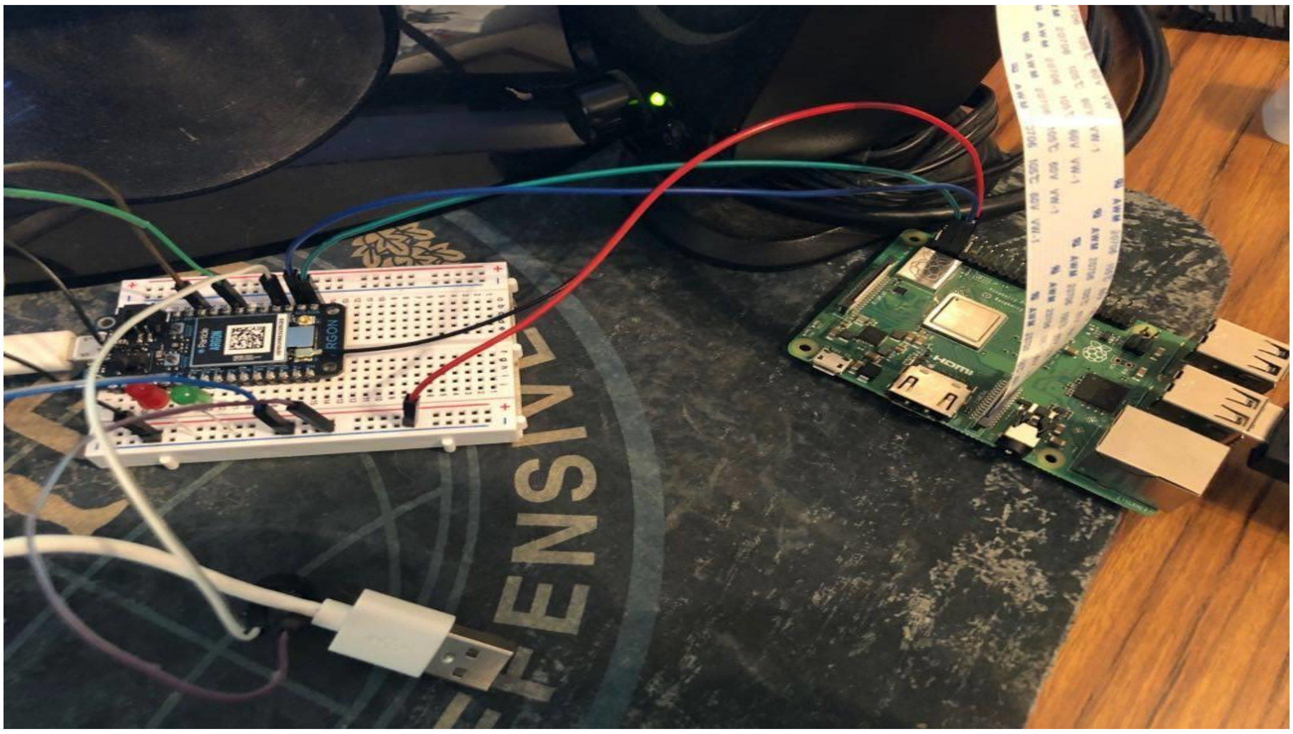
Connect positive pin with wire to the pin of your choice in this case D3.

Connect Ground pin with wire to the negative blue column like the rest.



And we have completed the system.

The prototype architecture is complete



This system uses I2C which is a master/slave system and needs to be connected with wires from the raspberry Pi to the Argon device. However, you can make this wireless by using the Argon device to publish when a variable occurs, this will then send an event through Wi-Fi which can then be detected by the raspberry pi and act accordingly.

We have finished the hardware setup and design now let's move to the software side of things.

Software Design

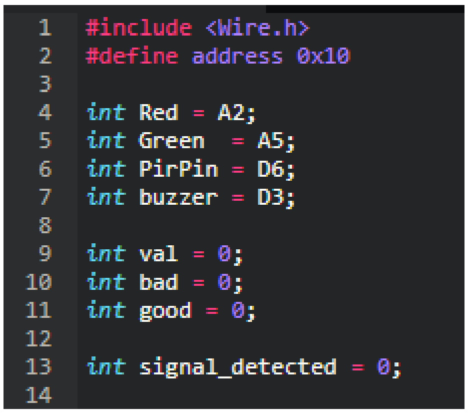
In this section we will be exploring the software setup and design of the system so that It works correctly, the software uses OpenCV module, Python, C++ and HTML to create the system.

First step: Argon software

Within the argon device we need to configure the system so that the system can send data and connect to the Raspberry Pi as well as get data from sensors. Include the wire.h this library is used for the signal communication

Define the address of the argon device. This is only for the slave device

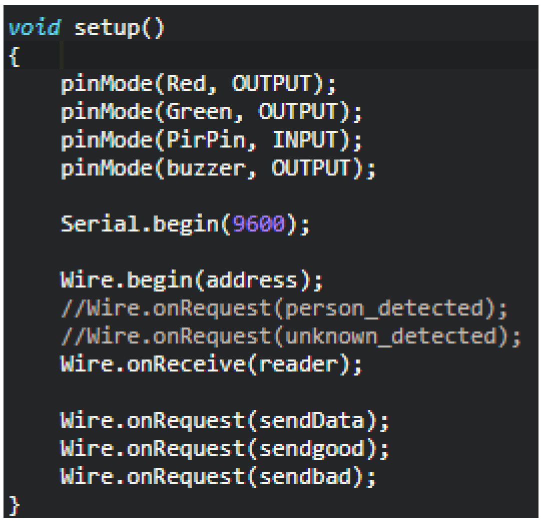
David Adams 216110104



```
#include <Wire.h>
#define address 0x10

int Red = A2;
int Green  = A5;
int PirPin = D6;
int buzzer = D3;

int val = 0;
int bad = 0;
int good = 0;

int signal_detected = 0;
```

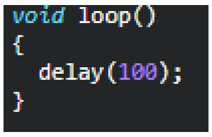
Define pins and variables being used.

Setup initialises the pins and connection



```
void setup()
{
    pinMode(Red, OUTPUT);
    pinMode(Green, OUTPUT);
    pinMode(PirPin, INPUT);
    pinMode(buzzer, OUTPUT);

    Serial.begin(9600);

    Wire.begin(address);
    //Wire.onRequest(person_detected);
    //Wire.onRequest(unknown_detected);
    Wire.onReceive(reader);

    Wire.onRequest(sendData);
    Wire.onRequest(sendgood);
    Wire.onRequest(sendbad);
}
```

This loops the program

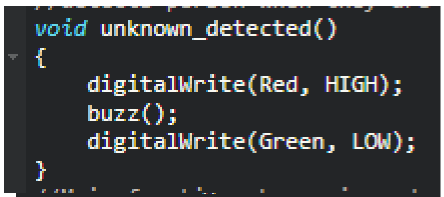


```
void loop()
{
  delay(100);
}
```

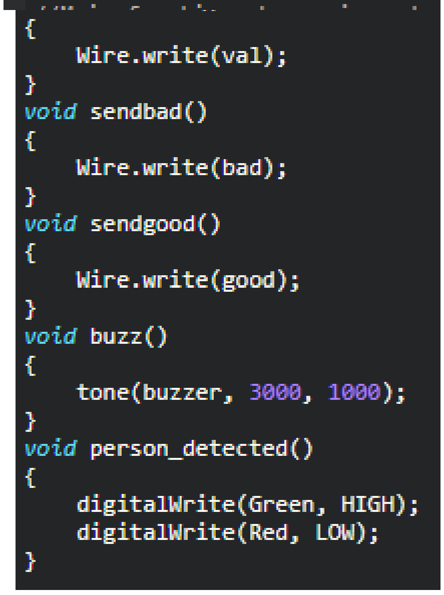
Functions that will be used in the main function.

These functions are created so that code compactness and complexity is easily distinguishable and understandable.

Doing this will allow calling functions and make the main function easier to understand.

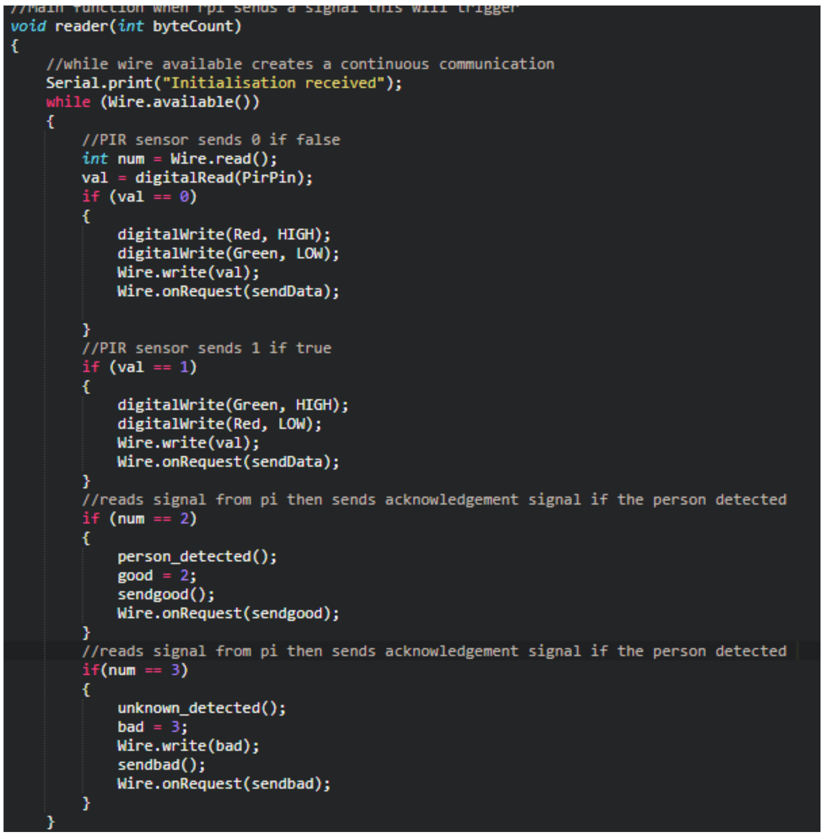


```
void unknown_detected()
{
    digitalWrite(Red, HIGH);
    buzz();
    digitalWrite(Green, LOW);
}
```



```
{
    Wire.write(val);
}
void sendbad()
{
    Wire.write(bad);
}
void sendgood()
{
    Wire.write(good);
}
void buzz()
{
    tone(buzzer, 3000, 1000);
}
void person_detected()
{
    digitalWrite(Green, HIGH);
    digitalWrite(Red, LOW);
}
```

The main function being initiated when a signal is sent is the reader, this will allow a continuous communication stream between the raspberry Pi and argon device.



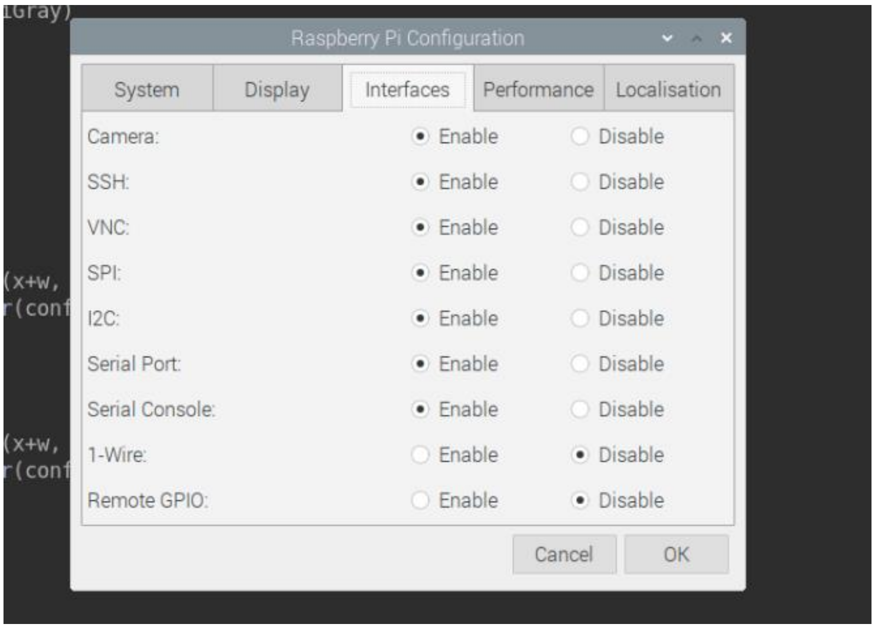
```
//main function when rpi sends a signal this will trigger
void reader(int byteCount)
{
    //while wire available creates a continuous communication
    Serial.print("Initialisation received");
    while (Wire.available())
    {
        //PIR sensor sends 0 if false
        int num = Wire.read();
        val = digitalRead(PirPin);
        if (val == 0)
        {
            digitalWrite(Red, HIGH);
            digitalWrite(Green, LOW);
            Wire.write(val);
            Wire.onRequest(sendData);

        }
        //PIR sensor sends 1 if true
        if (val == 1)
        {
            digitalWrite(Green, HIGH);
            digitalWrite(Red, LOW);
            Wire.write(val);
            Wire.onRequest(sendData);
        }
        //reads signal from pi then sends acknowledgement signal if the person detected
        if (num == 2)
        {
            person_detected();
            good = 2;
            sendgood();
            Wire.onRequest(sendgood);
        }
        //reads signal from pi then sends acknowledgement signal if the person detected
        if(num == 3)
        {
            unknown_detected();
            bad = 3;
            Wire.write(bad);
            sendbad();
            Wire.onRequest(sendbad);
        }
    }
```

Setting up the Raspberry Pi

Step 1: Installation of required files

Follow the steps provided above to install OpenCV and other files.

Next go to your preferences and enable I2C and Camera.

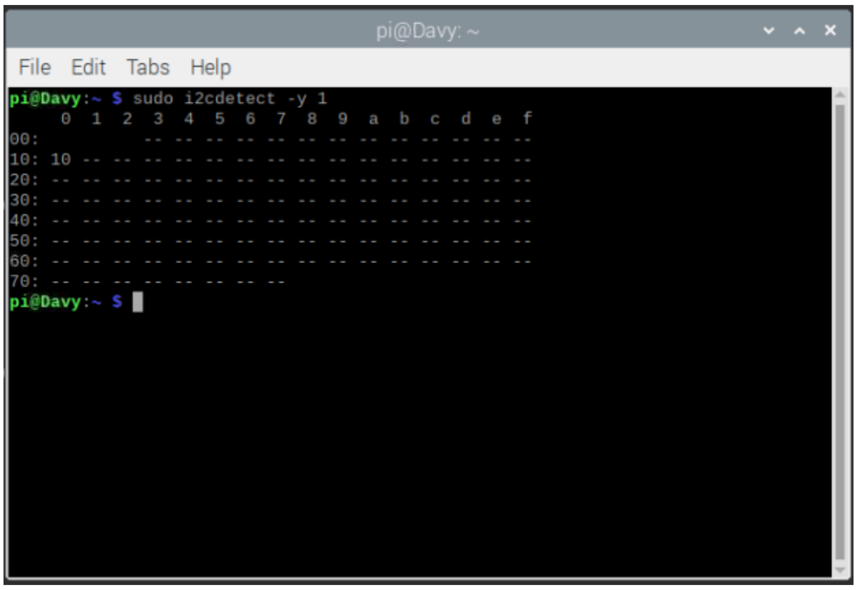


Camera:	[✓ Enable]	Disable
SSH:	[✓ Enable]	Disable
VNC:	[✓ Enable]	Disable
SPI:	[✓ Enable]	Disable
I2C:	[✓ Enable]	Disable
Serial Port:	[✓ Enable]	Disable
Serial Console:	[✓ Enable]	Disable
1-Wire:	Enable	[✓ Disable]
Remote GPIO:	Enable	[✓ Disable]

Step 2: I2C connection

Make sure your devices are connected through I2C and to check if you are

Type in sudo i2cdetect -y 1

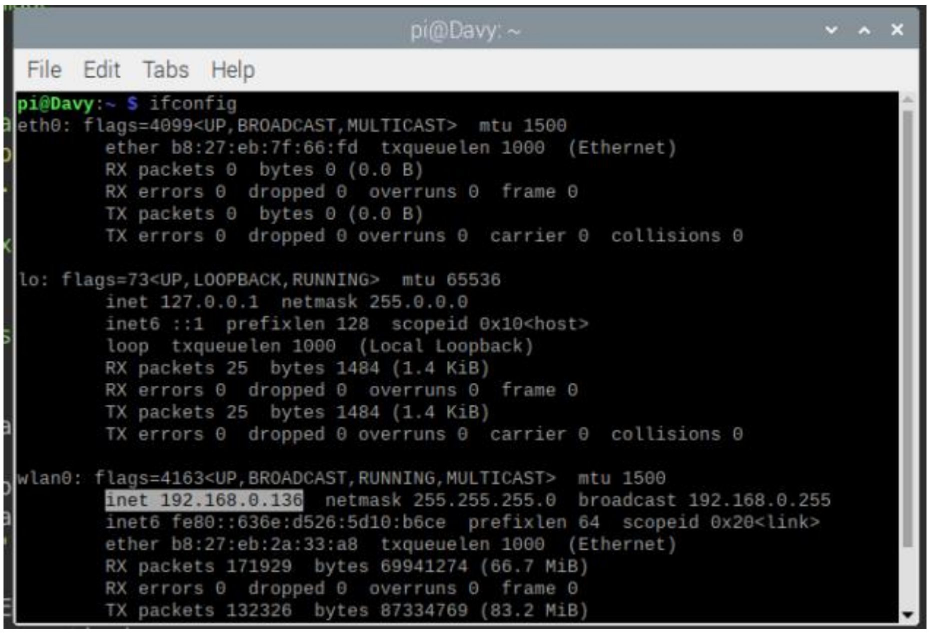


```
pi@Davy:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: 10 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@Davy:~ $
```

You will see the address location on the bus.

Next is to find your IP address for the device you are hosting off

Type ifconfig



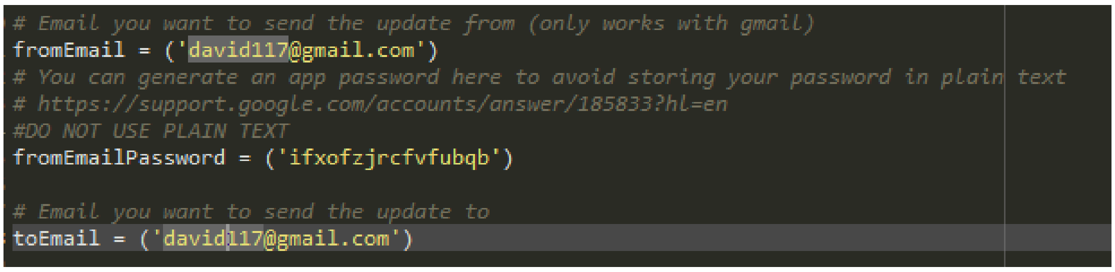
```
pi@Davy:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:7f:66:fd  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 25  bytes 1484 (1.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25  bytes 1484 (1.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.136  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::636e:d526:5d10:b6ce  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:2a:33:a8  txqueuelen 1000  (Ethernet)
        RX packets 171929  bytes 69941274 (66.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 132326  bytes 87334769 (83.2 MiB)
```

This will be used to access the video stream.

Step 3: Implementing the code

First, we will start by adding our details within mail python program.

All we need to do is add our email address and password, DO NOT USE PLAIN TEXT go to the URL listed to generate a key.

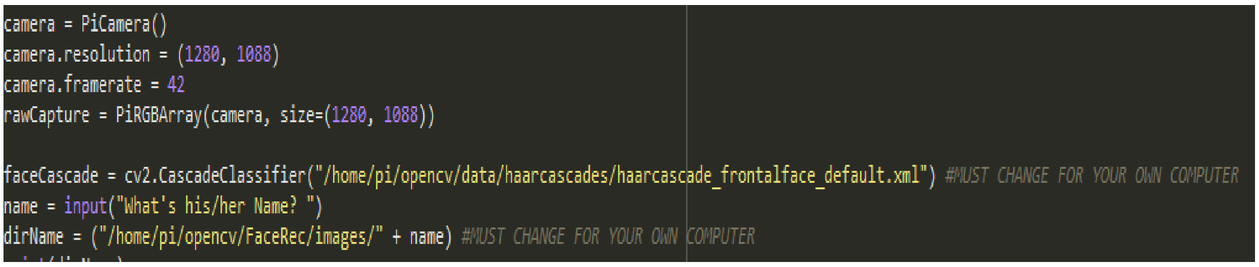


```
# Email you want to send the update from (only works with gmail)
fromEmail = ('david117@gmail.com')
# You can generate an app password here to avoid storing your password in plain text
# https://support.google.com/accounts/answer/185833?hl=en
#DO NOT USE PLAIN TEXT
fromEmailPassword = ('ifxofzjrcfvfubqb')

# Email you want to send the update to
toEmail = ('david117@gmail.com')
```

FaceRecognition

Next, we will go to the face recognition file and change the resolution and framerate if you want, but if you do so they must match with the other files so make sure you update them.

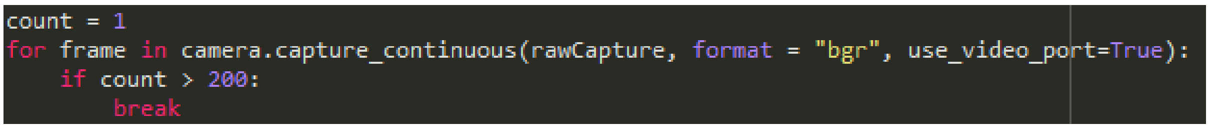
Then get the classifiers xml file where you store it by giving it the address



```
camera = PiCamera()
camera.resolution = (1280, 1088)
camera.framerate = 42
rawCapture = PiRGBArray(camera, size=(1280, 1088))

faceCascade = cv2.CascadeClassifier("/home/pi/opencv/data/haarcascades/haarcascade_frontalface_default.xml") #MUST CHANGE FOR YOUR OWN COMPUTER
name = input("What's his/her Name? ")
dirName = ("/home/pi/opencv/FaceRec/images/" + name) #MUST CHANGE FOR YOUR OWN COMPUTER
```

Then we can change the accuracy/photo count, simply choose your number.

The higher it is the more accurate it is.



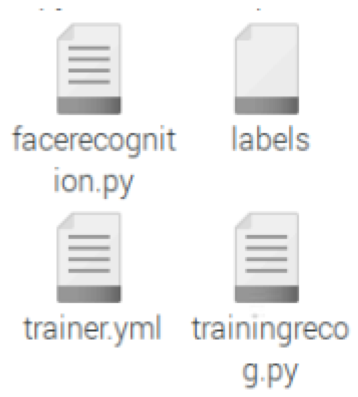
```
count = 1
for frame in camera.capture_continuous(rawCapture, format = "bgr", use_video_port=True):
    if count > 200:
        break
```

And done onto the next step.

Training recognition

We move onto the next file and run it

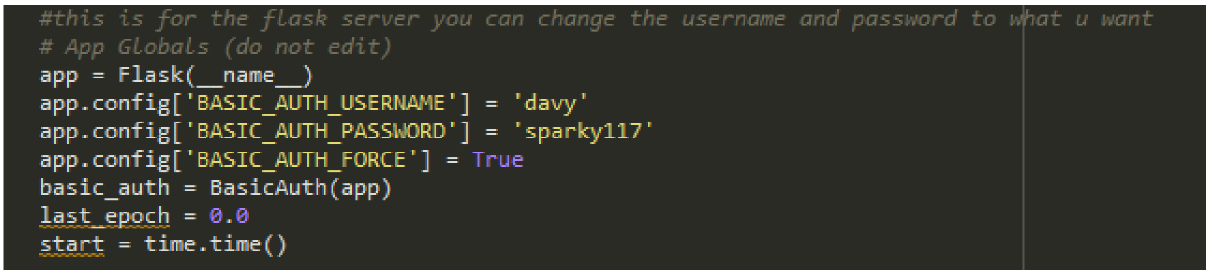
This will create two files LABELS and trainer



Now that this is done, we can go to our main program.

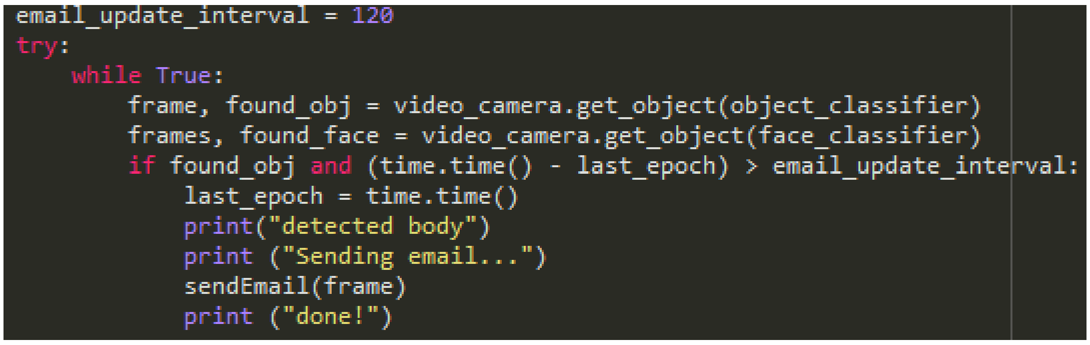
FaceCamMain

In this file we will be modifying the username and password for your own benefit and change the file locations for your OpenCV classifiers



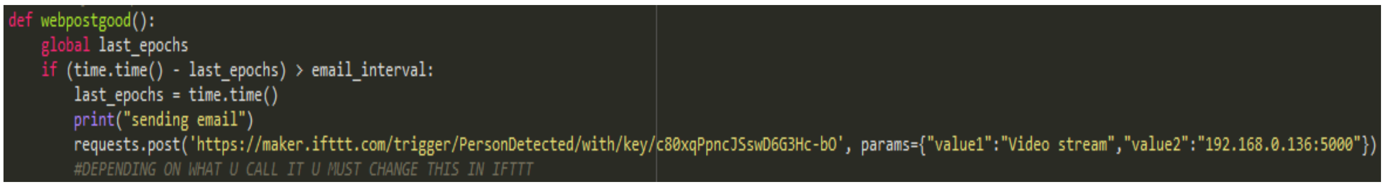
```
#this is for the flask server you can change the username and password to what u want
# App Globals (do not edit)
app = Flask(__name__)
app.config['BASIC_AUTH_USERNAME'] = 'davy'
app.config['BASIC_AUTH_PASSWORD'] = 'sparky117'
app.config['BASIC_AUTH_FORCE'] = True
basic_auth = BasicAuth(app)
last_epoch = 0.0
start = time.time()
```

You can also change the interval in which you want to receive emails



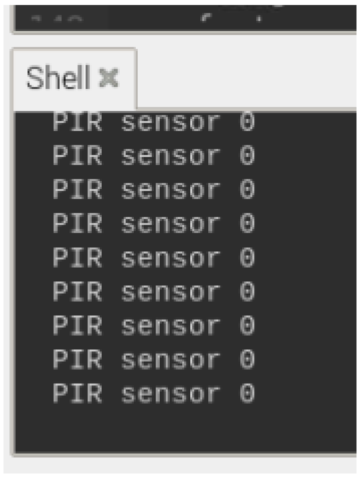
```
email_update_interval = 120
try:
    while True:
        frame, found_obj = video_camera.get_object(object_classifier)
        frames, found_face = video_camera.get_object(face_classifier)
        if found_obj and (time.time() - last_epoch) > email_update_interval:
            last_epoch = time.time()
            print("detected body")
            print ("Sending email...")
            sendEmail(frame)
            print ("done!")
```

Next we want to change the ifttt trigger by creating an applet.

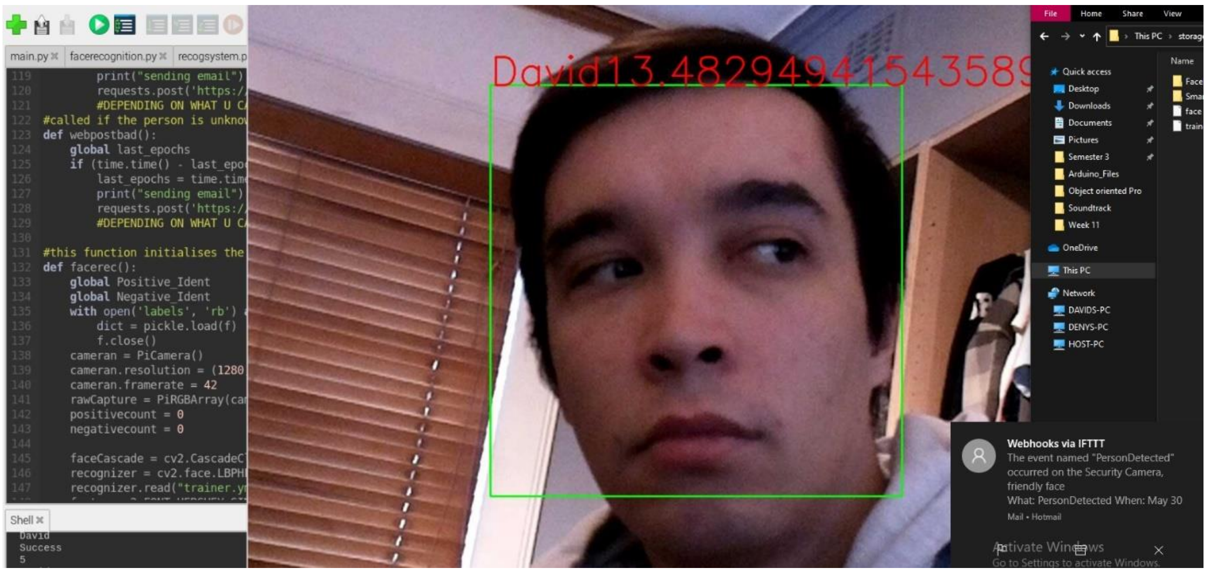


```
def webpostgood():
    global last_epochs
    if (time.time() - last_epochs) > email_interval:
        last_epochs = time.time()
        print("sending email")
        requests.post('https://maker.ifttt.com/trigger/PersonDetected/with/key/c80xqPpncJSswD6G3Hc-b0', params={"value1":"Video stream","value2":"192.168.0.136:5000"})
        #DEPENDING ON WHAT U CALL IT U MUST CHANGE THIS IN IFTTT
```

After this our program is ready to start



```
PIR sensor 0
PIR sensor 0
PIR sensor 0
PIR sensor 0
PIR sensor 0
PIR sensor 0
PIR sensor 0
PIR sensor 0
PIR sensor 0
```



```
            print("sending email")
            requests.post('https:/
            #DEPENDING ON WHAT U C
#called if the person is unknow
def webpostbad():
    global last_epochs
    if (time.time() - last_epo
        last_epochs = time.tim
        print("sending email")
        requests.post('https:/
        #DEPENDING ON WHAT U C

#this function initialises the
def facerec():
    global Positive_Ident
    global Negative_Ident
    with open('labels', 'rb')
        dict = pickle.load(f)
        f.close()
    cameran = PiCamera()
    cameran.resolution = (1280
    cameran.framerate = 42
    rawCapture = PiRGBArray(ca
    positivecount = 0
    negativecount = 0

    faceCascade = cv2.CascadeC
    recognizer = cv2.face.LBPH
    recognizer.read("trainer.y
```

```
David
Success
5
```

Webhooks via IFTTT
The event named "PersonDetected" occurred on the Security Camera, friendly face
What: PersonDetected When: May 30
Mail • Hotmail

And here you will see the program working, it will send an email saying it detected a face

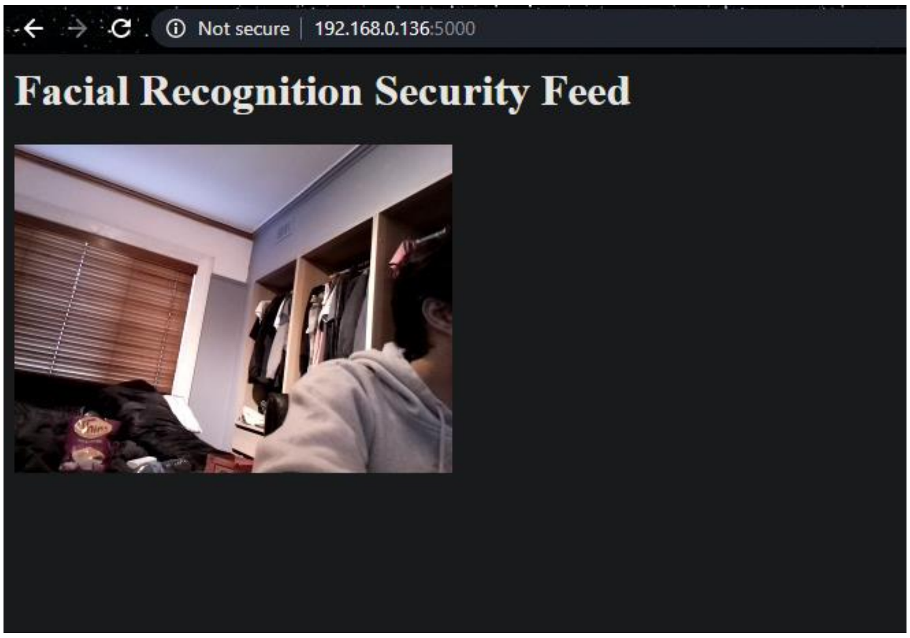
And if it's an unknown face the live feed will activate and buzzer and led will change colour and hence send an email of the screen.

Website Live feed

In order to access the website, you will need to enter the devices IP address with the port number being 5000

Eg. http://192.168.0.136:5000/

Then enter username details.



Not secure | 192.168.0.136:5000

Facial Recognition Security Feed

And thus, we see our solution to our problem complete.

Testing Approach

With testing the system, I have created contingencies and methods to increase accuracy and robustness of the solution.

First, what if other objects are detected this is when our facial recognition system starts and during this time there will be a timer where if nothing is detected the system will go back to the start.

What if there’s a false positive?

In this instance I have created a system that tallies the positive and negative identifications and whichever comes first will trigger the response. For example, if there’s a negative identification for 10 times the live stream and security alarm will sound and send an email. If there’s a positive identification for ten times then the device will go green, and loop back.

This solution aims to get an accuracy of 80% above, furthermore I have tested using multiple people within the frame and it accurately detects the person.

Testing Phone images, this is the one flaw I found when using an image of a face the accuracy rate drops significantly but if its in the range it will still have a positive identification. In order to fix this, change the accuracy rate.

What if they get through the recognition system? Well it automatically sends an email and within it is an image. So, you can verify the information sent.

Testing the system

For you to test the system you can wait out the facial recognition timer and see if it reboots, you can also change the resolution and frame rate to alter the accuracy.

You may also use photos or images to determine accuracy and fault detection. You may also use different and multiple facial classifiers to make a better system.