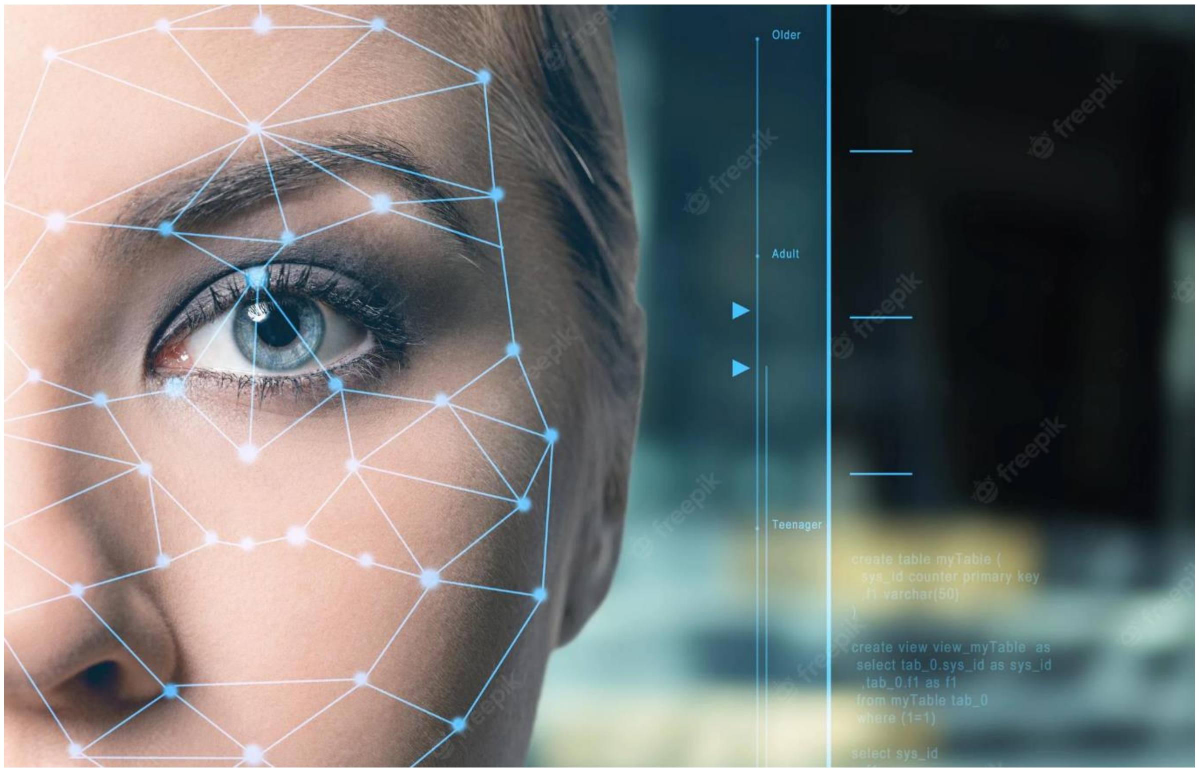
SMART DOOR UNLOCK SYSTEM USING FACE RECOGNITION



```
create table myTable (
  sys_id counter primary key
  f1 varchar(50)
)
create view view_myTable as
 select tab_0.sys_id as sys_id
  ,tab_0.f1 as f1
 from myTable tab_0
 where (1=1)
select sys_id
```

AS2019927
C.A.H.M.A.T. KARUNATHILAKA
MAT/PHY/ICT

Content

Abstract

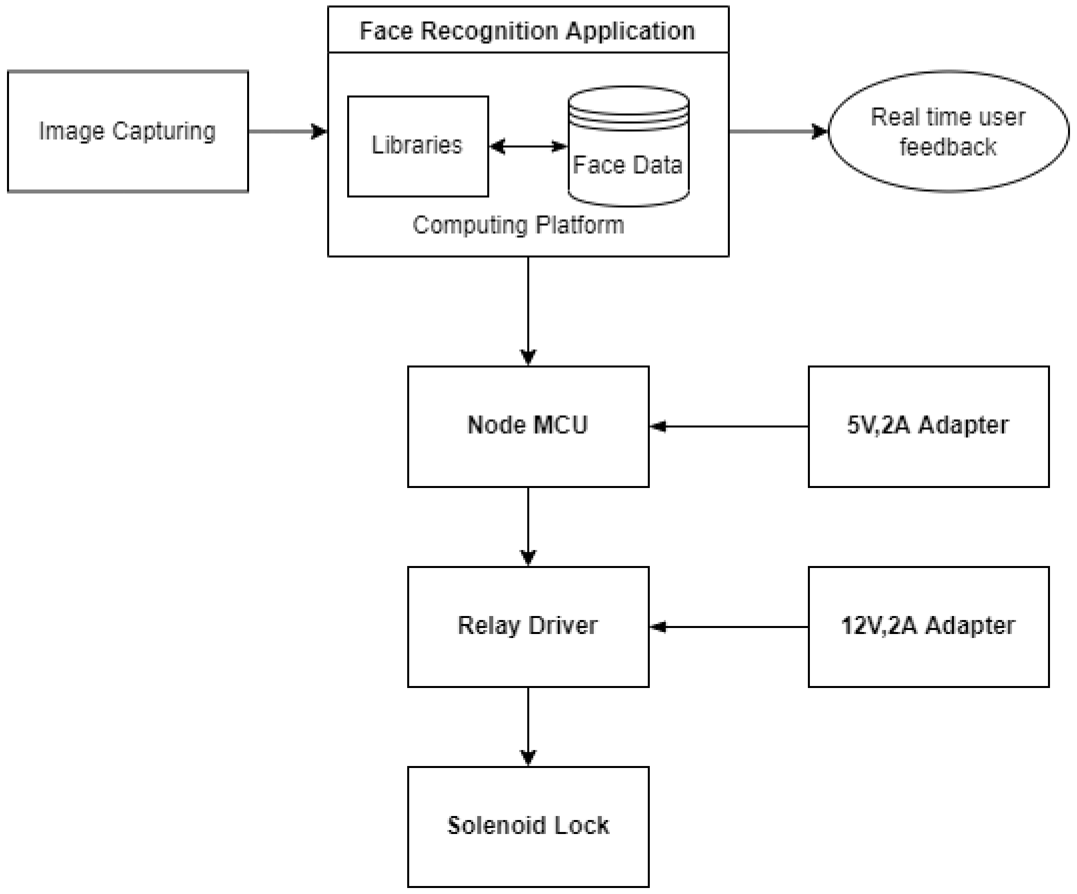
Face recognition system is widely used for human identification particularly for security functions. The scope of this project is to develop a security access control application based on face recognition. In this project, I am using the Face recognition module to capture human images and to compare with stored database images. The most important of characteristic of any home security system is to discover the people who enter or leave the house. Instead of monitoring that through passwords or pins, unique faces can be made use of as they're one's biometric characteristic. I aim to make a smart door, which secures the gateway on the base of who we are. Goal of this project is to help users improve the door security of sensitive places by using face detection and recognition

Introduction

Introduction Biometrics is unique to an individual and is used in numerous systems that involve security. In the face recognition approach, a given face is compared with the faces stored in the database to identify the person. The aim is to search out a face in the database, which has the most similarity with the given face. The field of smart home technology has grown by hops and bounds over the last several ages. As a result, there are several new products available that can add convenience and security to any home. Numerous people became aware of smart home devices with the introduction of the smart thermostat. Today, smart thermostats are only the beginning. Smart devices are helping people manage their schedules, their grocery lists, their home lighting, and indeed their home security. Ideally, all these devices will work together to make life a bit easier, and a bit safer, too. One group of smart home security devices that are gaining in popularity are smart door locks. Smart door locks are also seeing a bit of innovation lately, with some companies adding facial recognition capability. Facial recognition makes it possible to operate a door lock with nothing but your face.

Proposed System

The system will work in two different parts. The first part is for capturing images and upload them to the server. And the second one is to compare the image with the stored images cloud database

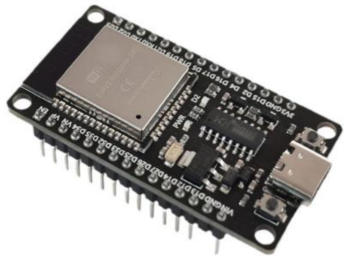


.

The System's primary input will be image captured by a Camera. This image data will be processed by a software application. For training the system, we need pre-captured face images and facial feature data in the training database. This will be integrated in the software application. The Node MCU board control the automated solenoid lock. The software application will give out command to Node MCU board. Relay driver board will drive solenoid lock, eventually allowing or denying access in the restricted area to the person subject to authentication.

Components

- NODEMCU (ESP32)



ESP32 is the name of the chip that was developed by Espressif Systems. This provides Wi-Fi (and in some models) dual-mode Bluetooth connectivity to embedded devices. While ESP32 is technically just the chip, modules and development boards that contain this chip are often also referred to as "ESP32" by the manufacturer.

- ESP32 Camera Module

ESP32-CAM

The ESP32-CAM is a very small camera module with the ESP32-S chip. The module has an OV2640 camera. It also has several GPIOs to connect peripherals. The ESP32-CAM module is developed by so many manufacturers, but the board developed by AI-Thinker is the most popular one.

- Solenoid Door Lock

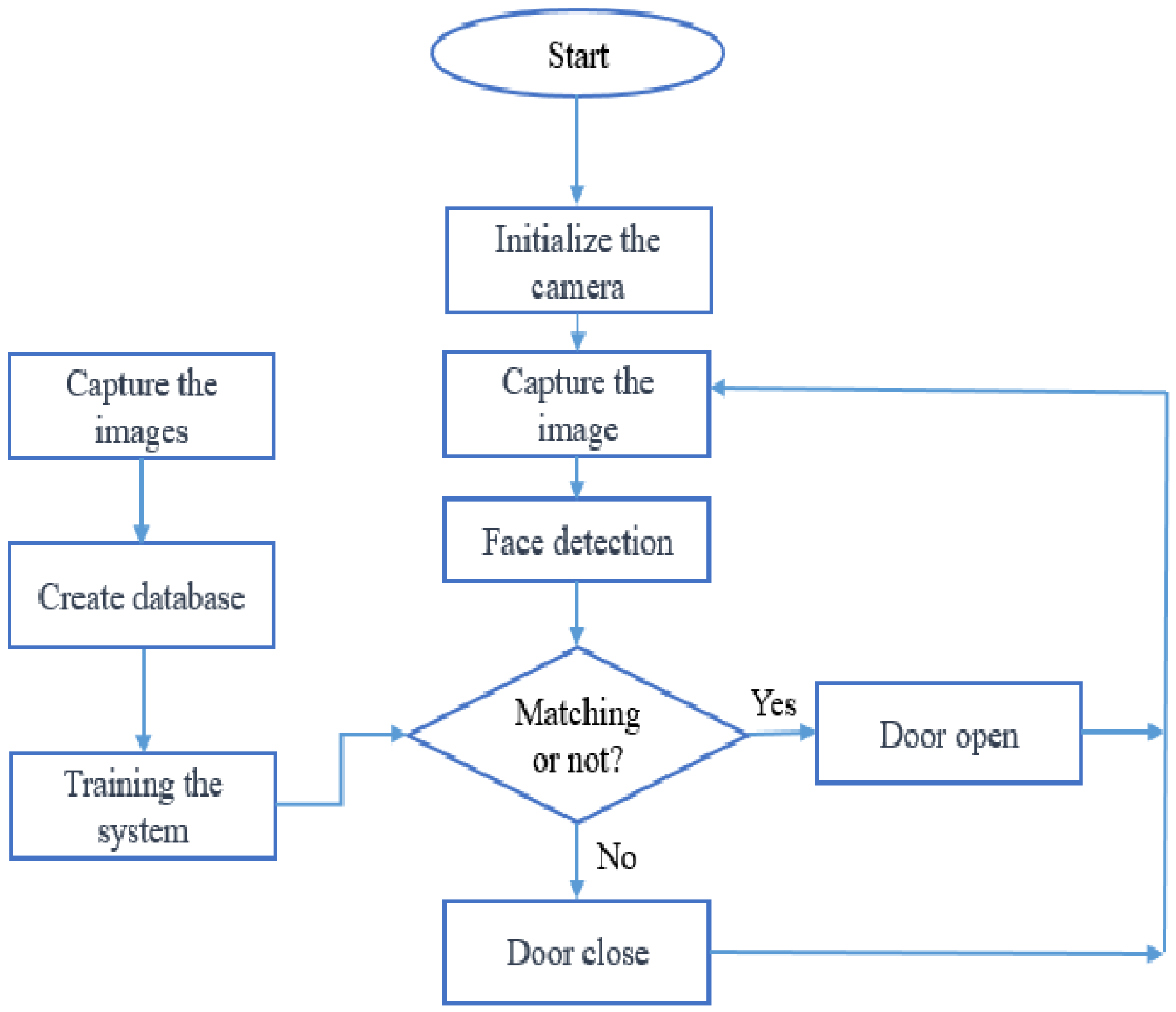
The solenoid lock denotes a latch for electrical locking and unlocking. It is available in unlocking in the power-on mode type and locking and keeping in the power-on mode type, which can be used selectively for situations. The power-on unlocking type enables unlocking only while the solenoid is powered on.

- Relay module

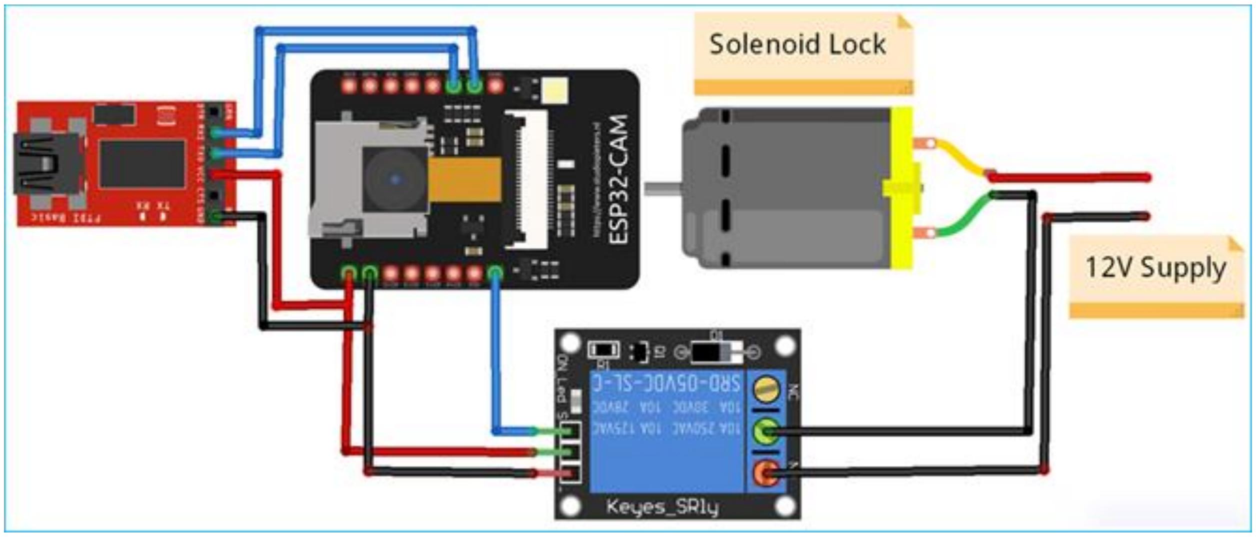
The relay module is an electrically operated switch that can be turned on or off deciding to let current flow through or not. They are designed to be controlled with low voltages like 3.3V like the ESP32, ESP8266, etc, or 5V like your Arduino

Design Overview

- ❖ FLOW CHART



❖ SIMULATE CIRCUIT DIAGRAM

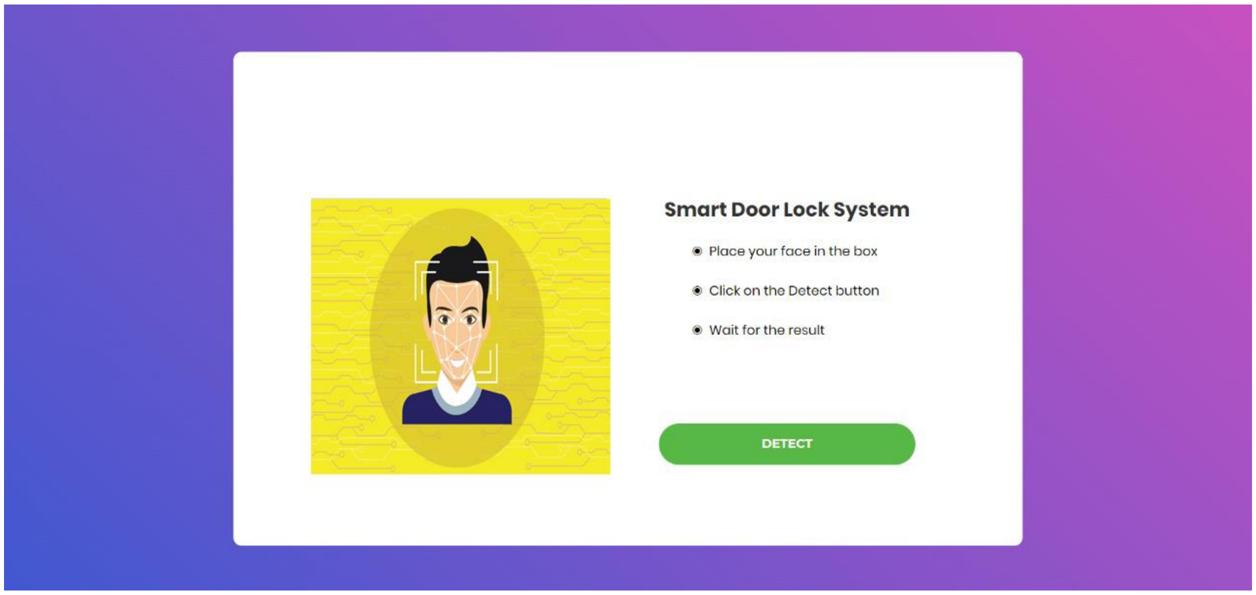


Project Cost

ITEM	COST
NODEMCU	2000
ESP32 Camera Module	2350
Solenoid Door Lock	1750
Relay Module	210
Jumper wires	240
Breadboard	240
TOTAL	6790

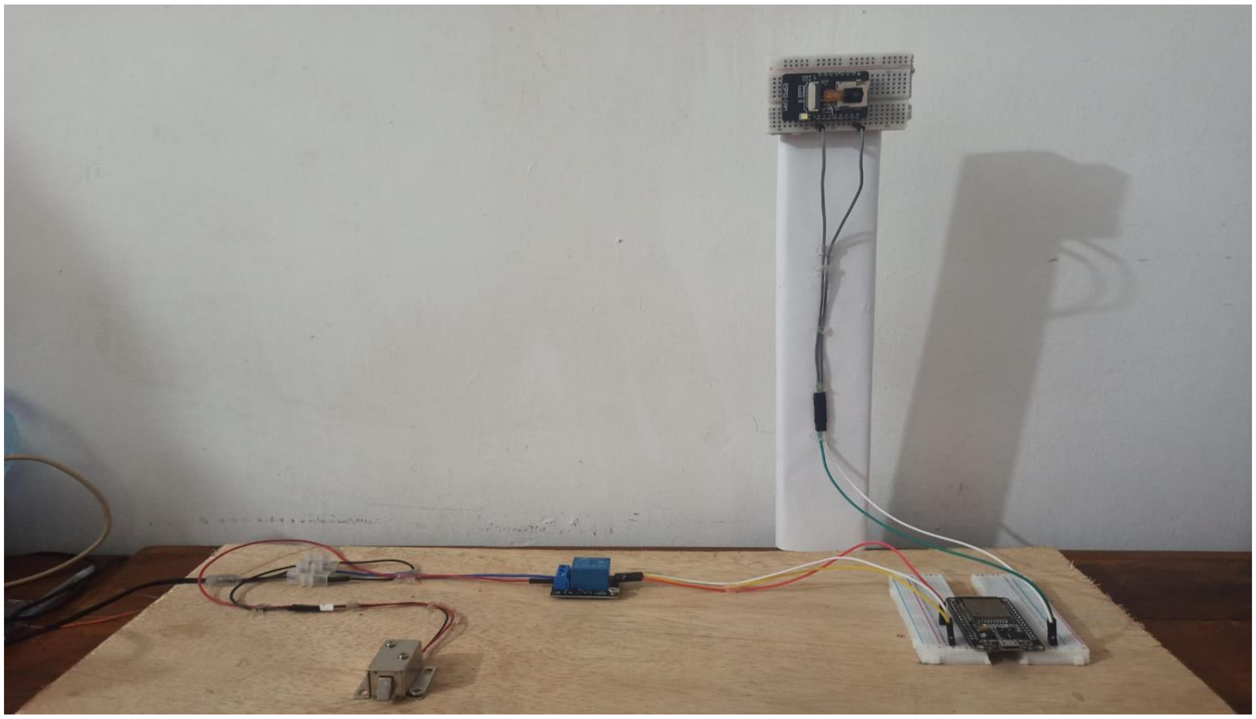
Structure of the prototype

Home Screen


Smart Door Lock System

- Place your face in the box
- Click on the Detect button
- Wait for the result

Prototype



Code

Main file

```
#include <WiFi.h>

// Replace with your network credentials
const char *ssid = "******";
const char *password = "******";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;
int s_lock_pin = 33;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms
= 2s)
const long timeoutTime = 2000;

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

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  pinMode(s_lock_pin, OUTPUT);
}

void loop()
{
  WiFiClient client = server.available(); // Listen for
incoming clients

  if (client)
  { // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client."); // print a message
out in the serial port
    String currentLine = "";        // make a String to
hold incoming data from the client
    while (client.connected() && currentTime -
previousTime <= timeoutTime)
    { // loop while the client's connected
```

```
      currentTime = millis();
      if (client.available())
      {                          // if there's bytes to
read from the client,
        char c = client.read(); // read a byte, then
        Serial.write(c);        // print it out the
serial monitor
        header += c;
        if (c == '\n')
        { // if the byte is a newline character
          // if the current line is blank, you got two
newline characters in a row.
          // that's the end of the client HTTP request,
so send a response:
          if (currentLine.length() == 0)
          {
            // HTTP headers always start with a response
code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows
what's coming, then a blank line:

            // GET /?value=180& HTTP/1.1
            if (header.indexOf("GET /unlock") >= 0)
            {
              digitalWrite(s_lock_pin, HIGH);
              client.println("HTTP/1.1 200 OK");
              client.println("Content-type:text/html");
              client.println();
              client.println("Connection closed");
              delay(5000);
              digitalWrite(s_lock_pin, LOW);
            }
      // The HTTP response ends with another blank
line
      client.println();
      // Break out of the while loop
      break;
     }
     else
     { // if you got a newline, then clear
currentLine
      currentLine = "";
     }
    }
    else if (c != '\r')
    {         // if you got anything else
but a carriage return character,
     currentLine += c; // add it to the end of the
currentLine
    }
   }
  }
  // Clear the header variable
  header = "";
  // Close the connection
  client.stop();
  Serial.println("Client disconnected.");
  Serial.println("");
 }
}
```

Face Recognition Part

```
from flask import Flask, render_template
import numpy as np
import requests
import cv2
from edge_impulse_linux.image import ImageImpulseRunner
app = Flask(__name__,
            static_url_path='',
            static_folder='./assets',
            template_folder='./templates')

runner = None
modelfile = './models/ashan_modelfile.eim'

async def detectFace(f_img):
    with ImageImpulseRunner(modelfile) as runner:
        try:
            model_info = runner.init()
            labels = 
model_info['model_parameters']['labels']
            image = cv2.cvtColor(f_img, 
cv2.COLOR_BGR2RGB)
            features, cropped = 
runner.get_features_from_image(image)
            res = runner.classify(features)
            if "classification" in res["result"].keys():
                for label in labels:
                    score = 
res['result']['classification'][label]
                print('', flush=True)
            elif "bounding_boxes" in
res["result"].keys():
                detected_labels = {}
                for bb in
res["result"]["bounding_boxes"]:
                    if bb['label'] in detected_labels:
                        detected_labels[bb['label']]['co
unt'] += 1
                        detected_labels[bb['label']]['sc
ore'] += bb['value']
                    else:
                        detected_labels[bb['label']] =
{'count': 1, 'score': bb['value']}
                    cropped = cv2.rectangle(
                        cropped, (bb['x'], bb['y']),
(bb['x'] + bb['width'], bb['y'] + bb['height']), (255,
0, 0), 1)

            mean_valued_labels = {}
            for label in detected_labels:
                mean_valued_labels[label] =
detected_labels[label]['score'] /
detected_labels[label]['count']

            return mean_valued_labels

        finally:
            if (runner):
                runner.stop()

@app.route('/')
@app.route('/index', methods=['GET'])
```

```
def index():
    return render_template('face_detection.html')


@app.route('/detect', methods=['GET'])
async def capture():
    img_url = "http://192.168.1.100/capture"
    detected_labels = {}
    try:
        img_resp = requests.get(img_url)
    except requests.exceptions.Timeout:
        return 'Request Timeout'
    except requests.exceptions.ConnectionError:
        return 'Camera Connection Error'
    img_arr = np.array(bytearray(img_resp.content),
dtype=np.uint8)
    img = cv2.imdecode(img_arr, -1)
    det_labels = await detectFace(img)
    for label in det_labels:
        if label in detected_labels:
            detected_labels[label]['count'] += 1
            detected_labels[label]['score'] +=
det_labels[label]
        else:
            detected_labels[label] = {'count': 1,
'score': det_labels[label]}
    mean_valued_labels = {}
    for label in detected_labels:
        mean_valued_labels[label] =
detected_labels[label]['score'] /
detected_labels[label]['count']
```

```
    mean_valued_labels = {k: v for k, v in
mean_valued_labels.items() if v > 0.8}
    print (mean_valued_labels)
    if len(mean_valued_labels) > 0:
         unlock_door =
requests.get('http://192.168.1.181/unlock')
         if unlock_door.status_code == 200:
            return {'success': 'true' }
    else :
      return {'success': 'false' }

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

References

https://www.youtube.com/

https://www.instructables.com/Programming-ESP32-CAM-With-ESP8266/

https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

https://esp32io.com/tutorials/esp32-solenoid-lock

https://www.hackster.io/o1991o/nodemcu-rfid-door-lock-1dfa34