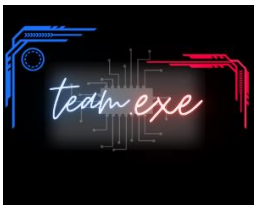
Project report on

Face recognition & Alerting system with ESP32 Cam

Submitted by:

Group Name: Team.exe



Group Members:

1. Jjateen Gundesha BT22ECI002
2. Parth Singh BT22ECI026
3. Lakshit Khandelwal BT22ECI027

A report submitted for the partial fulfilment of the requirements of the course

ECL-108 IoT Workshop - 1

Submission Date: 8/06/2023

Task Number # 4

Under the guidance of:

Dr. Mayank B. Thacker

Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

Indian Institute of Information Technology, Nagpur

Table of Contents:

Chapter 1: Introduction

In this project our main goal was to create a face recognition system using ESP-32 CAM and using the same system for Door unlocking and locking system

- ESP-WROOM-32 CAM Board.
- IC-7805 (5V Voltage Regulator)
- Solenoid Lock 12V
- 5V 10A Sugar cube Relay.
- 1N4007 Diode
- BC547 npn Transistor
- 330 Ω Resistor
- 100µF Capacitor
- Buzzer

We are controlling the solenoid lock with the use of Esp32 Cam via relay. On getting supply of 12V it get unlocked for 3 seconds. We also interfaced a GUI with the help of python to make it more user friendly.

Additionally, we've also made an intruder alert system which will respond on Unsuccessful Face detection. This alert system which send the image of intruder to the user via Telegram in delay of 5 seconds and at the same time it will turn on the buzzer of 1.5 seconds.
We've also upgraded this project by adding heat sink which will reduce the heat produced in ESP32 CAM drastically.

Chapter 2: About ESP32-CAM Board , Solenoid Lock and IC-7805

ESP - 32 CAM is a very popular development board especially used in IoT applications. It Combines Esp32 microcontroller with a camera module, which allows user to capture, record and perform various image processing tasks. We can easily program the board by the Arduino IDE and other development frameworks. Also it has in-built Wi-Fi Module which allows it to connect with internet wirelessly.

Fig (2.1)

One of the drawbacks of this board is that even to upload a simple code you need a separate TTL programmer to upload. And it also comes with a single Camera which means it can't be used in a project where multiple cameras are required.

It also has number of GPIO pins but they are limited if we campare them to Arduino UNO or Esp32-WROOM.

Solenoid lock is a type of electronic lock which can be controlled depending on its power supply. These type of solenoid locks are widely used in electronic and biometric based locks.

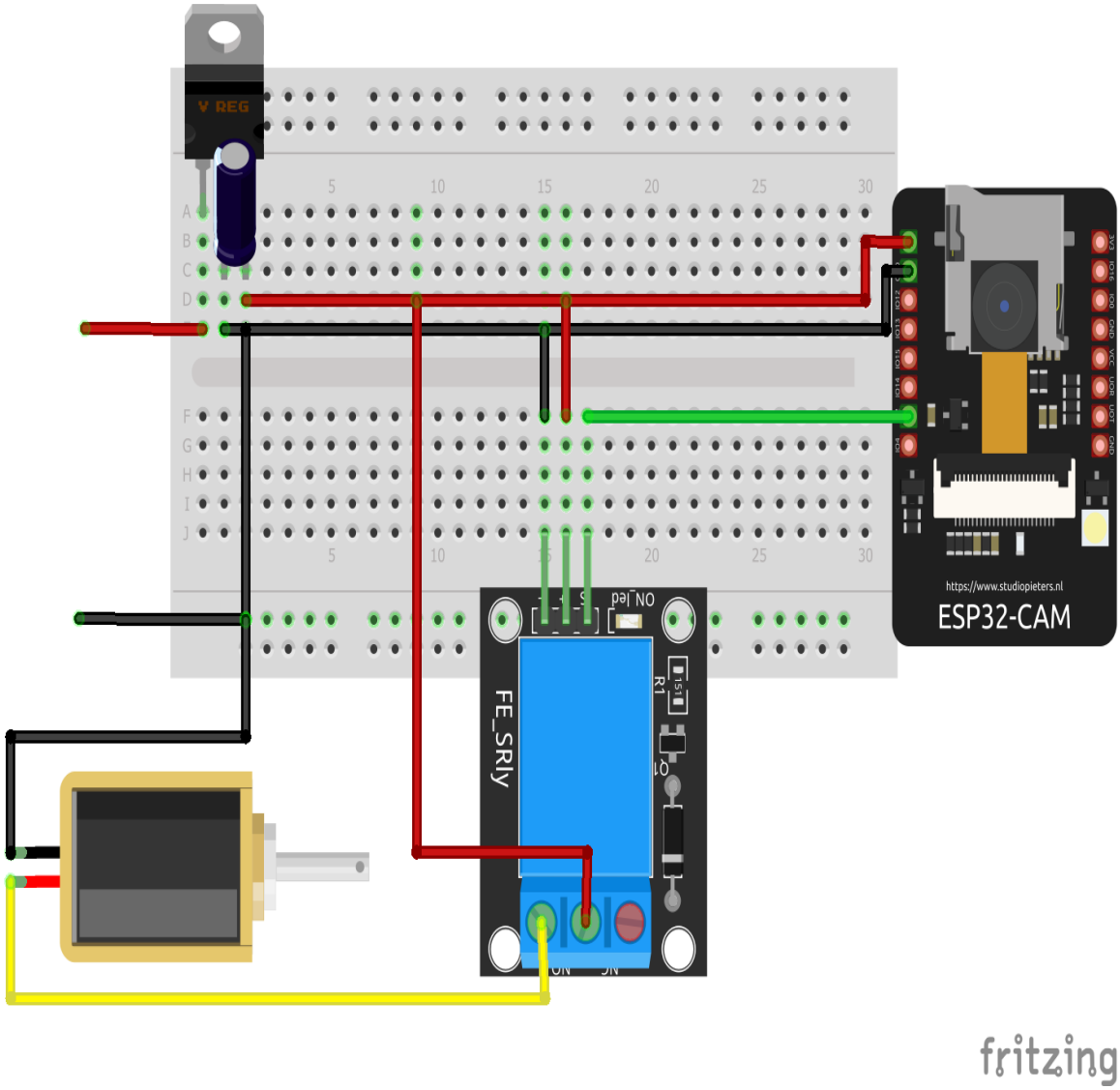
Fig (2.2)

IC-7805, it is a voltage regulator IC which is used to supply the constant 5V output signal. It has 3 pins named input, ground and output pin.

Fig (2.3)

A **Buzzer** is an audio signalling device which produces alarming sound when turned on.

Fig (2.4)



Chapter 3: Code

MAINFRAME ESP CODE-

```
#include <esp32cam.h>
#include <WebServer.h>
#include <WiFi.h>

const char* WIFI_SSID = "jjj";
const char* WIFI_PASS = "12345678";

WebServer server(80);

const int lockPin = 2;
const int buzzPin = 13;

static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
                static_cast<int>(frame->size()));
```

```
  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleLock1() {
  digitalWrite(lockPin, HIGH);
  server.send(200, "text/plain", "Lock set to 1");
  delay(3000);
  digitalWrite(lockPin, LOW);

}


void handleBuzz1() {
  digitalWrite(buzzPin, HIGH);
  server.send(200, "text/plain", "Buzzer set to 1");
  delay(1500);
  digitalWrite(buzzPin, LOW);
}


void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
```

```
  serveJpg();
}

void handleJpgHi()
{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}

void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}



void setup(){
  pinMode(lockPin, OUTPUT);
  pinMode(buzzPin, OUTPUT);
  digitalWrite(lockPin, LOW);

  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
```

```
  Config cfg;
  cfg.setPins(pins::AiThinker);
  cfg.setResolution(hiRes);
  cfg.setBufferCount(2);
  cfg.setJpeg(80);

  bool ok = Camera.begin(cfg);
  Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
 }
 WiFi.persistent(false);
 WiFi.mode(WIFI_STA);
 WiFi.begin(WIFI_SSID, WIFI_PASS);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
 }
 Serial.print("http://");
 Serial.println(WiFi.localIP());
 Serial.println("  /cam-lo.jpg");
 Serial.println("  /cam-hi.jpg");
 Serial.println("  /cam-mid.jpg");

 server.on("/cam-lo.jpg", handleJpgLo);
 server.on("/cam-hi.jpg", handleJpgHi);
 server.on("/cam-mid.jpg", handleJpgMid);
 server.on("/lock/1", HTTP_GET, handleLock1);
 server.on("/buzz/1", HTTP_GET, handleBuzz1);

 server.begin();
}
```

```
void loop()
{
  server.handleClient();
}
```

OLD-SCHOOL GUI VERSION(Have high latency)-

```
import cv2
import urllib.request
import numpy as np
import os
from datetime import datetime
import face_recognition
import time
import requests
import tkinter as tk
from PIL import ImageTk, Image

path = r'C:\Users\jjate\OneDrive\Desktop\esp32camTask4\image_folder'

def sendPhoto(image):
    api_token = '5955424544:AAFLjRnBna8E4bM3anpOnN_yjjCzwPLq0fE'
    chat_id = '5763052274'
    url = f'https://api.telegram.org/bot{api_token}/sendPhoto'
    temp_filename = 'temp_image.jpg'
    cv2.imwrite(temp_filename, image)
    with open(temp_filename, 'rb') as photo:
```

```
        files = {'photo': photo}
        params = {'chat_id': chat_id}
        response = requests.post(url, files=files, params=params)
    if response.status_code == 200:
        print('Photo sent successfully!')
    else:
        print('Failed to send photo.')

images = []
classNames = []
myList = os.listdir(path)
print(myList)
for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classNames.append(os.path.splitext(cl)[0])
print(classNames)

def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList

encodeListKnown = findEncodings(images)
print('Encoding Complete')
```

```
def process_frame():
    img_resp = urllib.request.urlopen(url)
    imgnp = np.array(bytearray(img_resp.read()), dtype=np.uint8)
    img = cv2.imdecode(imgnp, -1)
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

    for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
            name = classNames[matchIndex].upper()
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
            cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255,
255, 255), 2)
            requests.get(urlOn)
        else:
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 0, 255), cv2.FILLED)
```

```
            cv2.putText(img, "Intruder!!!", (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX,
1, (255, 255, 255), 2)
            sendPhoto(img)
            requests.get(buzzOn)
            print("Intruder!!!")

    cv2image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)
    panel.imgtk = imgtk
    panel.config(image=imgtk)
    panel.after(1, process_frame)

def start_processing():
    global url, urlOn, buzzOn
    ip_address = ip_entry.get()
    url = f'http://{ip_address}/cam-hi.jpg'
    urlOn = f'http://{ip_address}/lock/1'
    buzzOn = f'http://{ip_address}/buzz/1'
    root.after(1, process_frame)

root = tk.Tk()
root.title("ESP32Cam Security System")
root.configure(bg='black')

input_frame = tk.Frame(root, bg='grey')
input_frame.pack(pady=10)

ip_label = tk.Label(input_frame, text="IP Address:", bg='black', fg='cyan', font=('Arial', 14))
```

```
ip_label.grid(row=0, column=0)


ip_entry = tk.Entry(input_frame, bg='black', fg='orange', font=('Arial', 14))

ip_entry.grid(row=0, column=1)


start_button = tk.Button(root, text="Start", command=start_processing, bg='black', fg='cyan',
font=('Arial', 14),

                         relief='groove')

start_button.pack(pady=10)


panel = tk.Label(root, bg='black')

panel.pack(padx=10, pady=10)

root.mainloop()
```

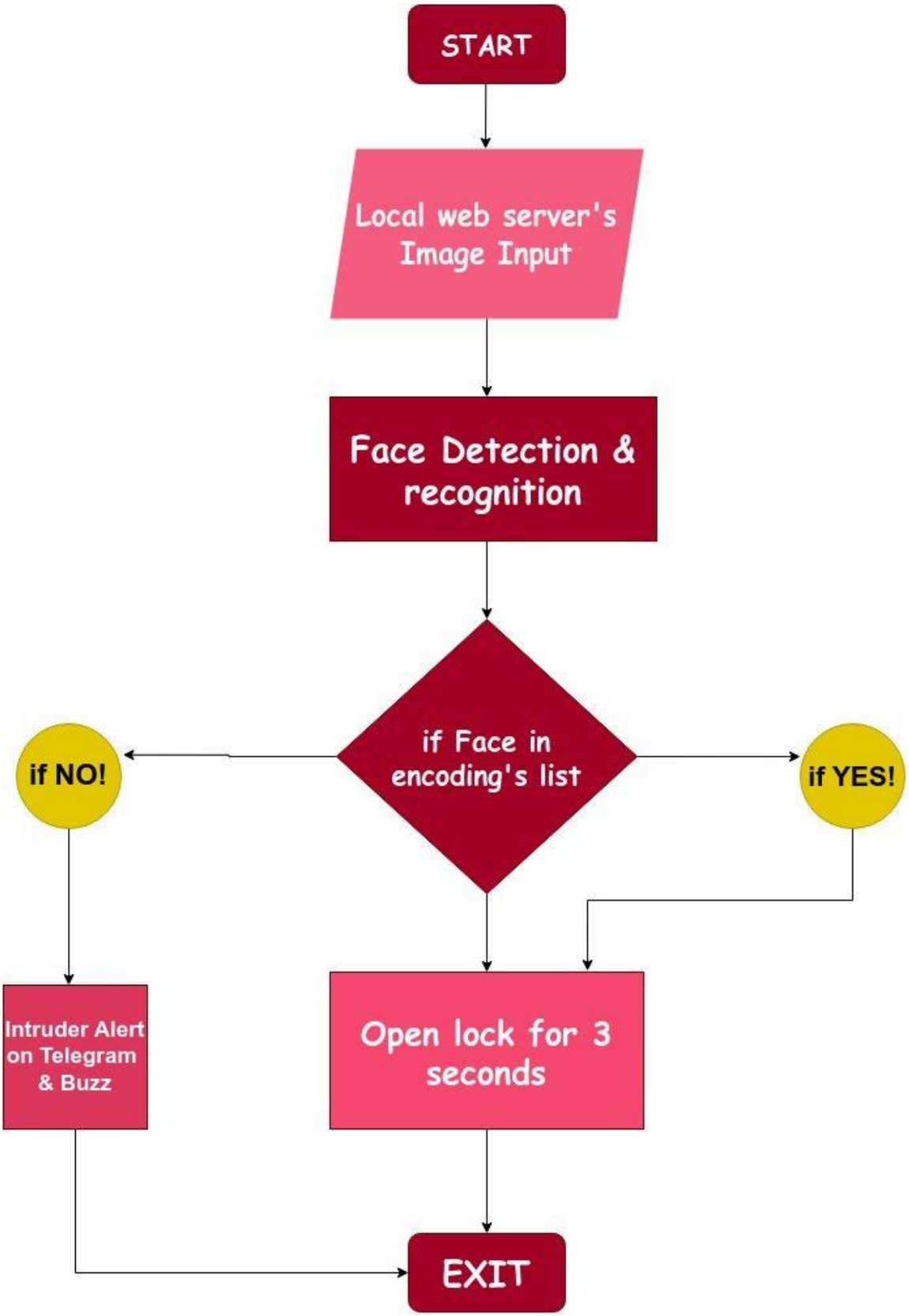
GUI MORE MORDERN VERSION (Have low latency but for higher resolution(above 1920 x 1080) its not responsive)-

```
import cv2
import urllib.request
import numpy as np
import os
from datetime import datetime
import face_recognition
import time
import requests
from PIL import Image, ImageTk
import customtkinter as tk
from customtkinter import CTkLabel as Label
from customtkinter import CTkButton as Button
from customtkinter import CTkEntry as Entry

path = r'C:\Users\jjate\OneDrive\Desktop\esp32camTask4\image_folder'

def sendPhoto(image):
```

```
    api_token =
'5955424544:AAFLjRnBna8E4bM3anpOnN_yjjCzwPLq0fE'
    chat_id = '5763052274'
    url = f'https://api.telegram.org/bot{api_token}/sendPhoto'
    temp_filename = 'temp_image.jpg'
    cv2.imwrite(temp_filename, image)
    with open(temp_filename, 'rb') as photo:
        files = {'photo': photo}
        params = {'chat_id': chat_id}
        response = requests.post(url, files=files, params=params)
    if response.status_code == 200:
        print('Photo sent successfully!')
    else:
        print('Failed to send photo.')

images = []
classNames = []
myList = os.listdir(path)
print(myList)
for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classNames.append(os.path.splitext(cl)[0])
print(classNames)

def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList

encodeListKnown = findEncodings(images)
print('Encoding Complete')

def process_frame():
    img_resp = urllib.request.urlopen(url)
    imgnp = np.array(bytearray(img_resp.read()), dtype=np.uint8)
    img = cv2.imdecode(imgnp, -1)
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
```

```
    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS,
facesCurFrame)

    for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown,
encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown,
encodeFace)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
            name = classNames[matchIndex].upper()
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0),
cv2.FILLED)
            cv2.putText(img, name, (x1 + 6, y2 - 6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
            requests.get(urlOn)
        else:
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 0, 255),
cv2.FILLED)
            cv2.putText(img, "Intruder!!!", (x1 + 6, y2 - 6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
            print("Intruder")
            sendPhoto(img)
            requests.get(buzzOn)

    cv2image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)
    panel.imgtk = imgtk
    panel.configure(image=imgtk)
    panel.after(1, process_frame)

def start_processing():
```

```
    global url, urlOn, urlOff, buzzOn, buzzOff
    ip_address = ip_entry.get()
    url = f'http://{ip_address}/cam-hi.jpg'
    urlOn = f'http://{ip_address}/lock/1'
    buzzOn = f'http://{ip_address}/buzz/1'
    root.after(1, process_frame)

root = tk.CTk()
root.title("ESP32Cam Security System")

input_frame = tk.CTkFrame(root, )
input_frame.pack(pady=10)

ip_label = Label(input_frame, text="IP Address:")
ip_label.grid(row=0, column=0)

ip_entry = Entry(input_frame)
ip_entry.grid(row=0, column=1)

start_button = Button(root, text="Start", command=start_processing)
start_button.pack(pady=10)

panel = Label(root, text="")
panel.pack(padx=10, pady=10)

root.mainloop()
```


START
Local web server's Image Input
Face Detection & recognition
if Face in encoding's list
if NO!
if YES!
Intruder Alert on Telegram & Buzz
Open lock for 3 seconds
EXIT

References:

1. Fig (2.1) taken from given website " Tim's PC Applications: Tim's ESP32 Cam Viewer (tims-pc-applications.blogspot.com)"

2. Fig (2.2) taken from website "DC12V Magnetic Solenoid Lock For Smart Storage Cabinet,Intelligent Access Control System -S4A Access Control (s4a-access.com)".

3. Fig (2.3) taken from "ic 7805 - Bing images".

4. Fig (2.4) taken from website https://core-electronics.com.au/standard-buzzer.html.

4. Code –
"https://github.com/Jjateen/ArduinoProjects/tree/main/face_ecog_esp32Cam_Test1"