



سیستم‌های نهفته

دکتر انصاری

پروژه سامانه تشخیص چهره

الینا هژبری - ۴۰۱۱۷۰۶۶۱

ملیکا علیزاده - ۴۰۱۱۰۶۲۵۵

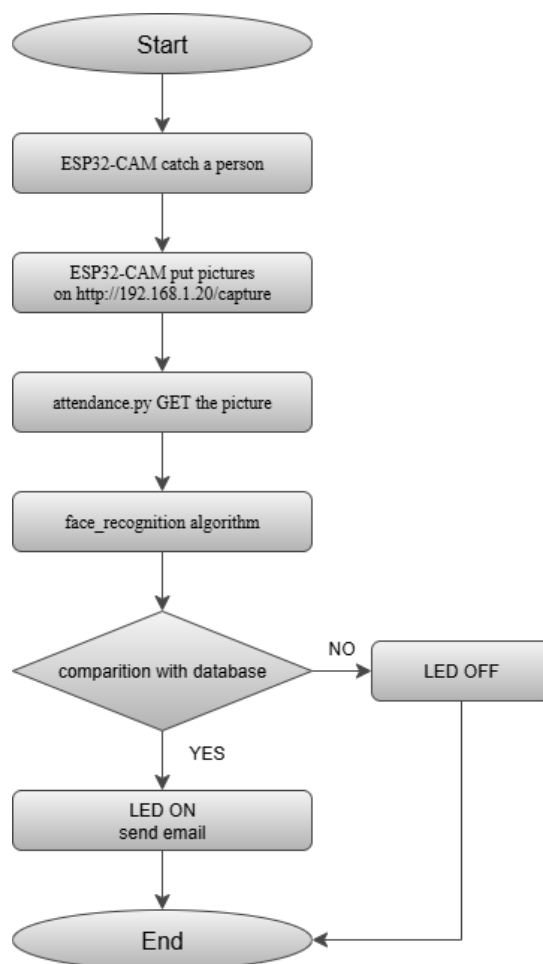
مقدمه

در این پروژه قصد داشتیم یک سامانه تشخیص چهره با استفاده از esp32Cam طراحی کنیم تا یک تجربه عملی از کاربرد اینترنت اشیا در زمینه امنیت و اتوماسیون داشته باشیم. پروژه به این صورت طراحی شده است که با استفاده از ماژول esp32Cam از فرد عکس گرفته می‌شود و عکس به کمک یک اسکریپت پایتون پردازش می‌شود. در صورت شناسایی فرد، نام شخص به همراه زمان ورود و عکس لحظه‌ای برای مدیر سرور ایمیل می‌شود و LED روشن می‌شود.

روند کلی سیستم

در این بخش روند کلی سیستم را توضیح می‌دهیم و مرحله به مرحله می‌گوییم سیستم به چه صورت کار می‌کند:

۱. ابتدا فرد در مقابل ESP32-CAM قرار می‌گیرد.
۲. سپس ماژول از فرد عکس گرفته و روی آدرس `capture` در سایت HTTP - که در پروژه ما `http://192.168.1.20` هست - قرار می‌دهد.
۳. فایل `attendance.py`، عکس را از سایت دریافت می‌کند.
۴. کتابخانه `face_recognition` ویژگی‌های چهره را استخراج کرده و با عکس‌های `encode` شده پایگاه داده مقایسه می‌کند.
۵. در صورتی که فرد با یکی از افراد پایگاه داده یکی باشد، LED روشن می‌شود و اطلاعات فرد + عکس لحظه‌ای او از طریق ایمیل ارسال می‌شود.

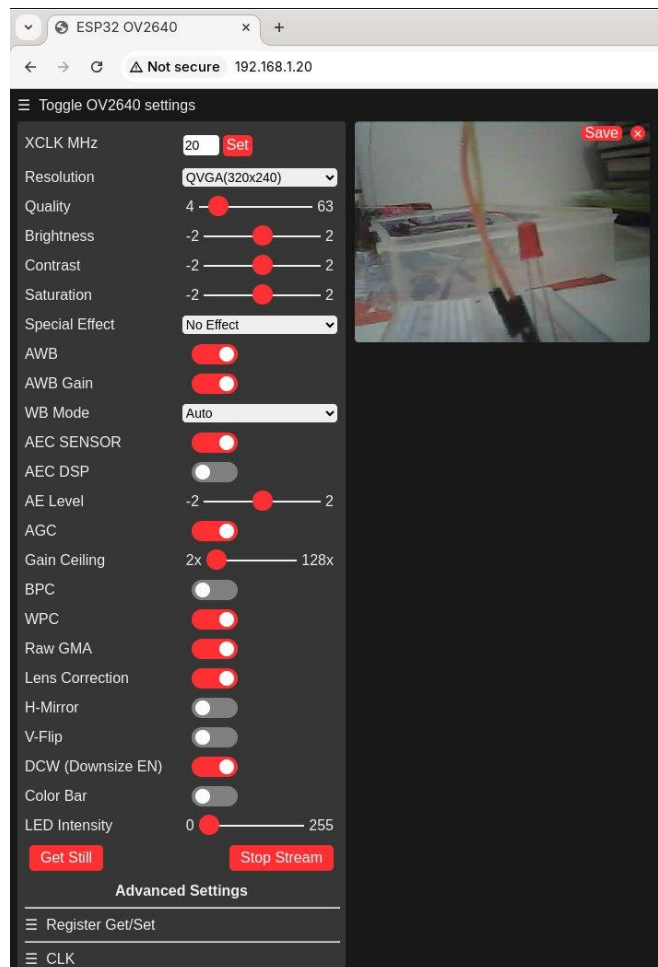


عکس ۱ - روند کلی سیستم

پیاده‌سازی بخش ESP32CAM

این بخش شامل چند فایل می‌شود که هر کدام را توضیح می‌دهیم:

- **Camera_index.h**: این فایل شامل کد HTML فشرده شده‌ای است که برای اینکه داخل حافظه فلش ESP32CAM ذخیره بشه، تبدیل به آرایه بایت شده است. این کد در واقع رابط کاربری وب ESP32CAM هست که می‌توان در آن دکمه‌ها و تنظیمات دوربین را مشاهده کرد.



عکس ۲- وب‌سرور ESP32CAM

- **Camera_pins.h**: این فایل مسئول تعریف پایه‌های سخت‌افزاری (GPIO) مربوط به ماژول‌های دوربین است. این فایل به این دلیل وجود دارد که ESP32 ماژول‌های دوربین متفاوتی دارد و پایه‌های هر کدام متفاوت است.
- **Board_config.h**: در این فایل، ماژول دوربین مورد نظر را انتخاب می‌کنیم که در پروژه ما **CAMERA_MODEL_AI_THINKER** هست.

- App_httpd.cpp: این فایل در واقع HTTP server مازول دوربین را پیاده‌سازی می‌کند. وقتی ESP32CAM از پروتکل WiFi استفاده می‌کند، یک وب‌سرور داخلی روی IP خودش راه‌اندازی می‌کند. فایل app_httpd تعیین می‌کند که هر آدرس چه کاری انجام دهد. در این فایل ما یک آدرس (/led) اضافه کردیم برای روشن و خاموش کردن LED. این کد شامل تعریف آدرس و handler آن می‌شود که در صورتی که action نوشته شده در لینک on باشد، LED را روشن می‌کند و در صورتی که action off باشد، LED را خاموش می‌کند و پاسخ (200) OK می‌فرستد. در صورتی که درخواست HTTP فرستاده شده action نداشته باشد نیز پاسخ (404) Error می‌فرستد.

```

23 #include "Arduino.h"
24
25 extern bool ledState;
26 extern unsigned long ledOnTime;
27 extern const unsigned long LED_DURATION;
28 #define RECOGNITION_LED_PIN 14
29
853 httpd_uri_t led_uri = {
854     .uri = "/led",
855     .method = HTTP_GET,
856     .handler = led_control_handler,
857     .user_ctx = NULL
858 #ifdef CONFIG_HTTPD_WS_SUPPORT
859     ,
860     .is_websocket = true,
861     .handle_ws_control_frames = false,
862     .supported_subprotocol = NULL
863 #endif
864 };
865
866 ra_filter_init(&ra_filter, 20);
867
868 log_i("Starting web server on port: '%d'", config.server_port);
869 if (httpd_start(&camera_httpd, &config) == ESP_OK) {
870     httpd_register_uri_handler(camera_httpd, &index_uri);
871     httpd_register_uri_handler(camera_httpd, &cmd_uri);
872     httpd_register_uri_handler(camera_httpd, &status_uri);
873     httpd_register_uri_handler(camera_httpd, &capture_uri);
874     httpd_register_uri_handler(camera_httpd, &bmp_uri);
875
876     httpd_register_uri_handler(camera_httpd, &xclk_uri);
877     httpd_register_uri_handler(camera_httpd, &reg_uri);
878     httpd_register_uri_handler(camera_httpd, &greg_uri);
879     httpd_register_uri_handler(camera_httpd, &pll_uri);
880     httpd_register_uri_handler(camera_httpd, &win_uri);
881     httpd_register_uri_handler(camera_httpd, &led_uri);
882 }
883
676 static esp_err_t led_control_handler(httpd_req_t *req) {
677     char *buf = NULL;
678     char action[32];
679
680     if (parse_get(req, &buf) != ESP_OK) {
681         return ESP_FAIL;
682     }
683
684     if (httpd_query_key_value(buf, "action", action, sizeof(action)) != ESP_OK) {
685         free(buf);
686         httpd_resp_send_404(req);
687         return ESP_FAIL;
688     }
689     free(buf);
690
691     if (strcmp(action, "on") == 0) {
692         digitalWrite(RECOGNITION_LED_PIN, HIGH);
693         ledState = true;
694         ledOnTime = millis();
695         Serial.println("Recognition LED turned ON via HTTP");
696     } else if (strcmp(action, "off") == 0) {
697         digitalWrite(RECOGNITION_LED_PIN, LOW);
698         ledState = false;
699         Serial.println("Recognition LED turned OFF via HTTP");
700     }
701
702     httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "");
703     return httpd_resp_send(req, "OK", 2);
704 }

```

عکس ۳- تعریف لینک /led و handler آن

- CameraWebServer.ino: این فایل کد اصلی ESP32CAM هست که در آن تنظیم و راه‌اندازی دوربین و وب‌سرور انجام می‌شود. کلیت این فایل از کد CameraWebServer که در Arduino ide هست نوشته شده است. ابتدا کتابخانه‌های مربوط به سخت‌افزار، اتصال به WiFi و ماژول دوربین را اضافه می‌کنیم. سپس اطلاعات WiFi ای که در حال حاضر سیستم به آن وصل است را می‌دهیم که شامل ssid (نام آن) و password می‌شود. پس از آن یکی از GPIO های ESP32CAM برای اتصال به LED و یک سری متغیر برای مدیریت روشن/خاموش کردن LED را تعریف می‌کنیم.

در تابع setup مقاداردهی‌ها و راه‌اندازی‌ها انجام می‌شود. در ابتدا serial فعال می‌شود که برای لاگ‌گیری و دیباگ است. پس از آن LED را به عنوان یک خروجی تعریف می‌کنیم و در ابتدا خاموش می‌کنیم. پس از آن پایه‌ها و عملکردهای دوربین را با یک سری دستور config مشخص می‌کنیم. قسمت بعد چک می‌کند اگر PSRAM وجود داشت عکس‌های باکیفیت‌تری بگیرد. در غیراین‌صورت رزولوشن را کاهش داده و بافرها را در DRAM قرار می‌دهد. همچنین بخش‌های برای مقاداردهی اولیه دوربین و تنظیم سنسورها داریم. سپس می‌رسیم به بخش اتصال WiFi و راه‌اندازی وب‌سرور که در آن یک سری لاگ برای اطمینان از درست کار کردن، قرار دادیم.

در آخر هم تابع loop را داریم که در آن مدیریت LED انجام می‌شود و چک می‌کند که LED فقط برای ۳ ثانیه روشن بماند و پس از آن خاموش شود.

```
15 // LED for face recognition indication
16 #define RECOGNITION_LED_PIN 14
17 bool ledState = false;
18 unsigned long ledOnTime = 0;
19 const unsigned long LED_DURATION = 3000; // LED stays on for 3 seconds
20
21 void startCameraServer();
22 void setupLedFlash();
23
24 void setup() {
25     Serial.begin(115200);
26     Serial.setDebugOutput(true);
27     Serial.println();
28
29     // Initialize recognition LED
30     pinMode(RECOGNITION_LED_PIN, OUTPUT);
31     digitalWrite(RECOGNITION_LED_PIN, LOW);
32
33     void loop() {
34         // Handle LED timing
35         if (ledState && (millis() - ledOnTime >= LED_DURATION)) {
36             digitalWrite(RECOGNITION_LED_PIN, LOW);
37             ledState = false;
38             Serial.println("Recognition LED turned OFF");
39         }
40     }
```

عکس ۴- بخش مدیریت LED در CameraWebServer

پیاده‌سازی بخش Python

بخش پایتون از سه بخش تشکیل شده است: ۱. بخش مربوط به ساخت dataset و ذخیره‌سازی ویژگی‌های چهره‌ها ۲. بخش مربوط به شناسایی عکس‌های گرفته شده از ESP32-CAM ۳. بخش ارسال ایمیل. کد هر بخش را به ترتیب توضیح می‌دهیم.

ساخت dataset و ذخیره‌سازی ویژگی‌های چهره‌ها

در این بخش، برای هر فرد که در database هست در فولدر dataset به نام شخص یک فولدر داریم که چند عکس از فرد در آن قرار دارد تا بتوانیم با استفاده از آن‌ها ویژگی‌های چهره افراد را یاد بگیریم و بعداً با عکس لحظه‌ای که توسط ESP32-CAM گرفته می‌شود، مقایسه کنیم.

برای استخراج ویژگی‌ها، روی فولدرهایی که در dataset قرار دارد for می‌زنیم و برای هر فایل آن ابتدا با استفاده از cv2.imread تصویر را می‌خوانیم. سپس با دستور cv2.cvtColor، رنگ BGR را به RGB تبدیل می‌کنیم و آن را به face_recognition.face_encodings می‌دهیم تا بردار ویژگی را استخراج کند.

در آخر ویژگی‌های استخراج شده‌ی تمامی افراد را در لیستی ذخیره می‌کنیم و در فایل encodings.pkl می‌ریزیم.

```
scripts > encode_faces.py > ...
1 import face_recognition
2 import cv2
3 import os
4 import pickle
5
6 path = "../dataset"
7 encodeList = []
8 classNames = []
9
10 for person in os.listdir(path):
11     person_path = os.path.join(path, person)
12     if not os.path.isdir(person_path):
13         continue
14     for file in os.listdir(person_path):
15         img = cv2.imread(os.path.join(person_path, file))
16         rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
17         enc = face_recognition.face_encodings(rgb)
18         if enc:
19             encodeList.append(enc[0])
20             classNames.append(person)
21
22 with open("../encodings.pkl", "wb") as f:
23     pickle.dump((encodeList, classNames), f)
24
25 print("Encodings saved to encodings.pkl")
```

عکس ۵- فایل encode_faces.py برای استخراج ویژگی‌های چهره‌های دیتاست

شناسایی عکس‌های گرفته شده توسط ESP32CAM

در این بخش، به لینک <http://192.168.1.20/capture> درخواست HTTP_GET می‌فرستیم تا عکس را دریافت کنیم. سپس با دستور `np.array` عکس گرفته شده را به آرایه تبدیل می‌کنیم و آن را به `cv2.imdecode` می‌دهیم تا آن را `decode` کند. سپس با دستور `cv2.cvtColor`، رنگ BGR را به RGB تبدیل می‌کنیم و آن را به `face_recognition.face_encodings` و `face_recognition.face_locations` می‌دهیم تا ابتدا بفهمد چهره فرد در کدام قسمت تصویر است و سپس ویژگی‌های آن قسمت‌ها را استخراج کند.

پس از آن روی ویژگی‌های استخراج شده و چهره `for` می‌زنیم و با دستورهای `face_recognition.compare_faces` و `face_recognition.face_distance` آن را با ویژگی‌های استخراج شده تصویرهای `dataset` مقایسه می‌کنیم و عکسی که کمترین اختلاف را با مدل داشته باشد را انتخاب می‌کنیم. اگر که چنین عکسی وجود داشت، نام، زمان ورود فرد را در یک `database` ذخیره می‌کنیم و آن را همراه با عکس لحظه‌ای فرد به تابع `send_email` می‌دهیم تا ایمیل را بفرستد و همچنین به تابع `turn_on_recognition_led` را صدا می‌کنیم تا LED روشن شود. در غیراین صورت تابع `turn_off_recognition_led` را صدا می‌کند تا LED را خاموش کند.

همچنین برای قسمت بصری کار نیز چهره فرد در عکس را با استفاده از یک مستطیل مشخص می‌کند و بالای آن نیز نام شخص را می‌نویسد.

```
75 while True:
76     try:
77         resp = requests.get(f"{url}/capture", timeout=5)
78         img_arr = np.array(bytearray(resp.content), dtype=np.uint8)
79         frame = cv2.imdecode(img_arr, cv2.IMREAD_COLOR)
80         rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
81
82         faces = face_recognition.face_locations(rgb)
83         encodes = face_recognition.face_encodings(rgb, faces)
84
85         for enc, loc in zip(encodes, faces):
86             matches = face_recognition.compare_faces(encodeListKnown, enc)
87             faceDis = face_recognition.face_distance(encodeListKnown, enc)
88             best = faceDis.argmin()
89
90             if matches[best]:
91                 name = classNames[best]
92                 time_str = datetime.now().strftime('%H:%M:%S')
93                 now = time.time()
94
95                 # Turn on LED for face recognition
96                 turn_on_recognition_led()
97
98                 # check cooldown
99                 if name not in last_sent or (now - last_sent[name]) > cooldown:
100                     last_sent[name] = now
101
102                     # log attendance
103                     df = pd.concat(
104                         [df, pd.DataFrame({'Name': [name], 'Time': [time_str]})])
105
106                     # save frame
107                     os.makedirs("../captures", exist_ok=True)
108                     img_filename = f"../captures/{name}_{time_str.replace(':', '-')}.jpg"
109                     cv2.imwrite(img_filename, frame)
110
111                     # send email
112                     send_email(name, time_str, img_filename)
113
114                     print(f"[+] {name} recognized and logged at {time_str}")
115
116                     # draw bounding box
117                     y1, x2, y2, x1 = loc
118                     cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
119                     cv2.putText(frame, name, (x1, y1 - 10),
120                                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
```

عکس ۶- فایل `attendance.py`، بخش مربوط به `face_recognition` برای عکس‌های گرفته شده توسط ESP32CAM

توابع مدیریت LED

در این بخش، به لینک <http://192.168.1.20/led> درخواست HTTP ارسال می‌کنیم. بسته به اینکه کدام تابع صدا زده شود، این درخواست می‌تواند برای روشن کردن یا خاموش کردن LED باشد. به این گونه که در لینک action مربوطه (on/off) را اضافه می‌کنیم. اگر پاسخ OK(200) دریافت شود یعنی مدیریت LED انجام شده است.

```
44 def turn_on_recognition_led():
45     """Turn on the LED on ESP32 when face is recognized"""
46     try:
47         response = requests.get(f"{url}/led?action=on", timeout=2)
48         if response.status_code == 200:
49             print("[+] Recognition LED turned ON")
50         else:
51             print(f"[-] Failed to turn on LED: {response.status_code}")
52     except requests.exceptions.RequestException as e:
53         print(f"[-] Error controlling LED: {e}")
54
55
56 def turn_off_recognition_led():
57     """Turn off the LED on ESP32"""
58     try:
59         response = requests.get(f"{url}/led?action=off", timeout=2)
60         if response.status_code == 200:
61             print("[+] Recognition LED turned OFF")
62         else:
63             print(f"[-] Failed to turn off LED: {response.status_code}")
64     except requests.exceptions.RequestException as e:
65         print(f"[-] Error controlling LED: {e}")
66
```

عکس ۷- فایل attendance.py، توابع روشن/خاموش کردن LED با ارسال درخواست HTTP

ارسال ایمیل

در این بخش، با استفاده از کتابخانه‌های `smtplib` و `EmailMessage`، یک ایمیل با موضوع ورود شخص به مدیر سرور فرستاده می‌شود. این ایمیل حاوی نام شخص، زمان ورود و عکس لحظه‌ای شخص است. به دلایل امنیتی برای ارسال ایمیل باید در تنظیمات ایمیل و برای ایمیل فرستنده یک `app password` ساخت تا امکان ارسال ایمیل وجود داشته باشد.

```
scripts > attendance.py > ...
1  import cv2
2  import face_recognition
3  import pickle
4  import pandas as pd
5  from datetime import datetime
6  import requests
7  import numpy as np
8  import time
9  import smtplib
10 from email.message import EmailMessage
11 import os
12
13 with open("../encodings.pkl", "rb") as f:
14     encodeListKnown, classNames = pickle.load(f)
15
16 df = pd.DataFrame(columns=['Name', 'Time'])
17 url = "http://192.168.1.20"
18
19 SENDER_EMAIL = "melikaalizadeh0@gmail.com"
20 APP_PASSWORD = "nrgx eixy aicp fjbw"
21 RECEIVER_EMAIL = "alizadehmelika369@gmail.com"
22
23
24 def send_email(name, time_str, image_path):
25     msg = EmailMessage()
26     msg["Subject"] = f"[Attendance] {name} entered at {time_str}"
27     msg["From"] = SENDER_EMAIL
28     msg["To"] = RECEIVER_EMAIL
29     msg.set_content(f"Person recognized: {name}\nTime: {time_str}")
30
31     # attach image
32     with open(image_path, "rb") as f:
33         img_data = f.read()
34     msg.add_attachment(img_data, maintype="image",
35                       subtype="jpeg", filename=os.path.basename(image_path))
36
37     # send email
38     with smtplib.SMTP_SSL("smtp.gmail.com", 465) as server:
39         server.login(SENDER_EMAIL, APP_PASSWORD)
40         server.send_message(msg)
41     print(f"[+] Email sent for {name} at {time_str}")
```

عکس ۸- فایل `attendance.py`، تابع ارسال ایمیل

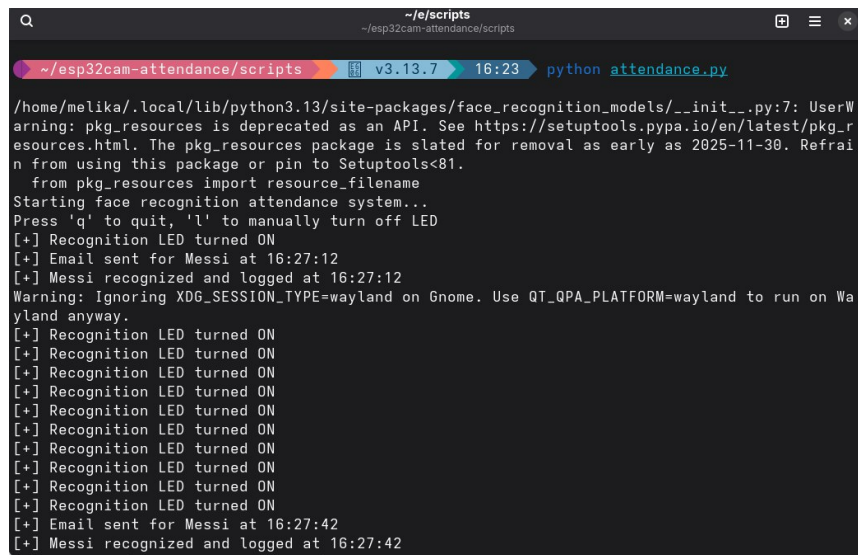
خروجی‌ها

خروجی‌های مربوط به هر بخش را نشان می‌دهیم:

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4980
load:0x40078000,len:16612
load:0x40080400,len:3480
entry 0x400805b4

WiFi connecting...
WiFi connected
Camera Ready! Use 'http://192.168.1.20' to connect
```

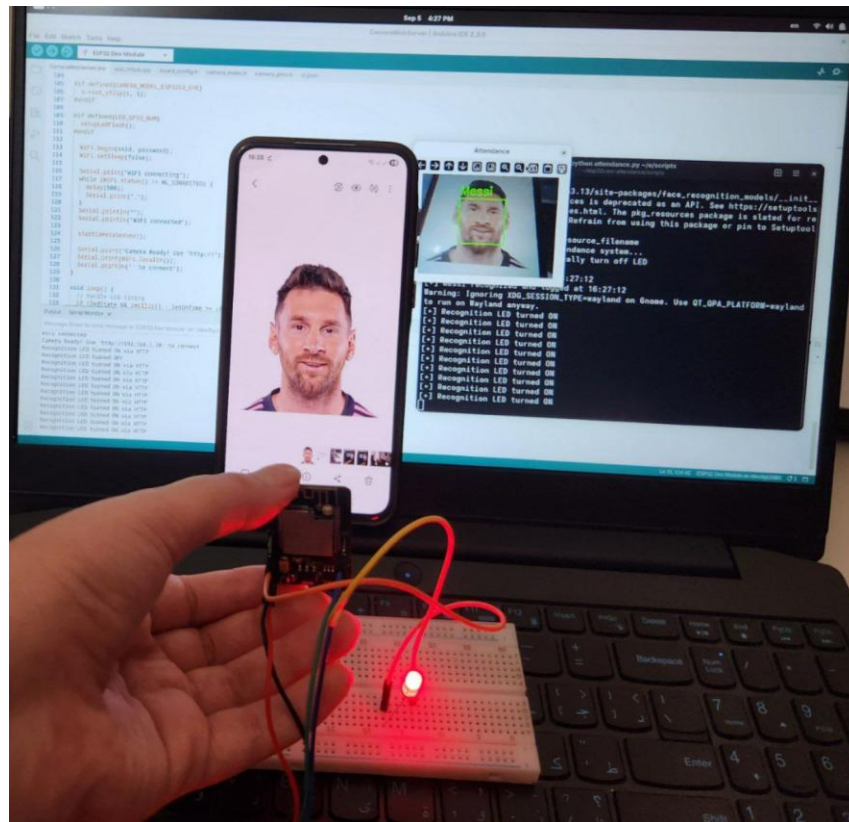
عکس ۹- خروجی CameraWebSerber.ino



```
~/e/scripts
~/esp32cam-attendance/scripts v3.13.7 16:23 python attendance.py

/home/melika/.local/lib/python3.13/site-packages/face_recognition_models/__init__.py:7: UserWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using this package or pin to Setuptools<81.
  from pkg_resources import resource_filename
Starting face recognition attendance system...
Press 'q' to quit, 'l' to manually turn off LED
[+] Recognition LED turned ON
[+] Email sent for Messi at 16:27:12
[+] Messi recognized and logged at 16:27:12
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Recognition LED turned ON
[+] Email sent for Messi at 16:27:42
[+] Messi recognized and logged at 16:27:42
```

عکس ۱۰- خروجی کد attendance.py



عکس ۱ - خروجی تشخیص چهره



عکس ۱۲ - ارسال ایمیل