
بسم الله الرحمن الرحيم

گزارش پروژه کارشناسی

موضوع: تشخیص چهره با استفاده از الگوریتم Haar Cascade بر روی Raspberry Pi و ارسال لاگ به اپلیکیشن اندرویدی با استفاده از پروتکل MQTT .

استاد راهنما: دکتر وحید سلوک

داور: دکتر پرویز رشیدی

دانشجو: حمیدرضا بابایی – 991152104

دانشگاه: صنعتی ارومیه

ترم تابستان سال تحصیلی 1403-1402

فهرست مطالب

مقدمه

رزبری پای یک مینی‌کامپیوتر با ابعاد کوچک است که به دلیل قدرت پردازش قابل قبول و قابلیت اجرای سیستم‌عامل‌های مختلف مانند لینوکس، به‌ویژه در پروژه‌های الکترونیکی، برنامه‌نویسی و اینترنت اشیا (IoT) به شدت مورد استفاده قرار می‌گیرد. این دستگاه با وجود ابعاد کوچک، قابلیت‌های زیادی مانند اجرای برنامه‌های مختلف، اتصال به شبکه‌های بی‌سیم، و همچنین پورت‌های ورودی و خروجی متعدد برای اتصال به سنسورها و دیگر دستگاه‌ها دارد.

رزبری پای در پروژه‌های تحقیقاتی، آموزشی و صنعتی به کار می‌رود و به عنوان یک ابزار ایده‌آل برای آموزش مبانی علوم کامپیوتر و الکترونیک مورد استفاده قرار می‌گیرد. این مینی‌کامپیوتر به راحتی قابل برنامه‌ریزی است و می‌تواند برای انجام وظایف مختلف مانند کنترل دستگاه‌های الکترونیکی، مانیتورینگ داده‌های سنسوری، و اجرای الگوریتم‌های پردازش تصویر و ویدئو استفاده شود.

مزایای رزبری پای شامل:

- **هزینه پایین:** رزبری پای نسبت به سایر کامپیوترهای شخصی قیمت بسیار پایینی دارد، که این امر آن را به یک گزینه مناسب برای پروژه‌های کم‌هزینه تبدیل می‌کند.
- **اندازه کوچک و قابل حمل:** ابعاد کوچک این دستگاه به کاربران اجازه می‌دهد تا آن را به راحتی در پروژه‌های خود یکپارچه کنند.
- **مصرف برق کم:** این دستگاه برای اجرای برنامه‌ها و عملیات‌های روزمره از مصرف برق کمتری برخوردار است.
- **پشتیبانی گسترده از جامعه برنامه‌نویسان:** وجود یک جامعه فعال از توسعه‌دهندگان و کاربران که به صورت مداوم در حال توسعه ابزارها و پروژه‌های جدید هستند.

معایب رزبری پای نیز شامل:

- **قدرت پردازش محدود:** در مقایسه با کامپیوترهای شخصی و سرورها، رزبری پای قدرت پردازشی کمتری دارد و ممکن است نتواند برنامه‌های سنگین را به راحتی اجرا کند.
- **نیاز به دانش فنی اولیه:** استفاده موثر از این دستگاه نیاز به دانش پایه در زمینه سیستم‌عامل‌های لینوکس و برنامه‌نویسی دارد.

کتابخانه‌های مورد استفاده:

- **OpenCV:** این کتابخانه یکی از پرکاربردترین ابزارها برای پردازش تصویر و ویدئو است. با استفاده ازOpenCV ، می‌توان پروژه‌های مختلفی مانند تشخیص چهره، شناسایی اشیاء و دیگر عملیات پردازش تصویر را انجام داد. این کتابخانه از زبان‌های برنامه‌نویسی مختلفی از جمله Python پشتیبانی می‌کند و دارای توابع و الگوریتم‌های متنوعی است که می‌توانند برای تحلیل تصاویر و ویدئوها استفاده شوند.
- **paho-mqtt:** این کتابخانه یک کلاینت برای پروتکل MQTT است که معمولاً در پروژه‌های اینترنت اشیا (IoT) برای انتقال داده‌ها بین دستگاه‌ها و سرورها استفاده می‌شود. MQTT یک پروتکل سبک و سریع است که برای انتقال داده‌ها به صورت پایدار و مطمئن در محیط‌های با پهنای باند محدود طراحی شده است.

در پروژه حاضر، از رزبری پای برای جمع‌آوری و پردازش داده‌ها از طریق یک دوربین متصل به دستگاه استفاده شده است. سپس این داده‌ها از طریق پروتکل MQTT به اپلیکیشن اندرویدی ارسال می‌شوند تا اطلاعات مربوطه به کاربر نمایش داده شود. در این راستا، رزبری پای به عنوان یک واحد پردازشگر عمل می‌کند که داده‌های ویدئویی را دریافت، تحلیل و سپس نتایج را به یک سرور مرکزی یا اپلیکیشن اندرویدی منتقل می‌کند.

شرح پروژه

سخت‌افزار :

در این پروژه، **Raspberry Pi** به عنوان واحد پردازش مرکزی به کار گرفته شده است. به دلیل اندازه کوچک، مصرف برق پایین، و قابلیت‌های متنوع در پروژه‌های مختلف به ویژه در زمینه‌های اینترنت اشیا (IoT) و پردازش تصویر به کار می‌رود.

رزبری پای به یک **دوربین** متصل شده است که وظیفه ثبت و ضبط تصاویر و ویدیوها را بر عهده دارد. این دوربین، که به پورت CSI (Camera Serial Interface) رزبری پای متصل می‌شود، می‌تواند تصاویر با وضوح بالا را در زمان واقعی (Real-Time) به رزبری پای ارسال کند تا پردازش‌های لازم بر روی آن‌ها انجام شود. این تصاویر برای شناسایی و تشخیص چهره‌ها به کار می‌روند.

(در این پروژه به دلیل نبود دوربین مخصوص رزبری مجبور به استفاده از گوشی یا تبلت هستیم.)
گوشی و رزبری هر دو به یک Access Point متصل می‌شوند، در واقع هر دو در یک شبکه بوده و ویدیو از طریق این شبکه به رزبری ارسال می‌شود.

در این پروژه، رزبری پای با استفاده از دوربین متصل، به طور مداوم تصاویر و ویدئوهای دریافتی را پردازش می‌کند و در صورت شناسایی چهره‌ها، اطلاعات مربوطه را ذخیره و یا به سرور ارسال می‌کند. از این رو، رزبری پای نه تنها وظیفه پردازش تصاویر را بر عهده دارد، بلکه نقش کلیدی در برقراری ارتباط و انتقال داده‌ها نیز ایفا می‌کند.

کتابخانه‌های استفاده شده:

کتابخانه:OpenCV

یکی از مهم‌ترین کتابخانه‌هایی که در این پروژه استفاده شده است، کتابخانه **OpenCV (Open Source Computer Vision Library)** است. OpenCV یک کتابخانه منبع باز است که در زمینه پردازش تصویر و ویدئو به طور گسترده‌ای استفاده می‌شود. این کتابخانه دارای مجموعه‌ای از الگوریتم‌های قدرتمند برای انجام وظایفی مانند شناسایی چهره، تشخیص اشیاء، ویدئو استریمینگ، پردازش تصاویر سه‌بعدی، و بسیاری از کاربردهای دیگر است.

در این پروژه، OpenCVبه منظور **شناسایی چهره‌ها** از ویدیوهای دریافتی از دوربین رزبری پای به کار گرفته شده است. این کتابخانه از الگوریتم‌های مختلفی برای تشخیص چهره استفاده می‌کند که می‌توانند چهره‌ها را در تصاویر شناسایی کنند، آن‌ها را با دیتابیس‌های موجود تطبیق دهند و نتایج حاصل را ذخیره یا انتقال دهند.

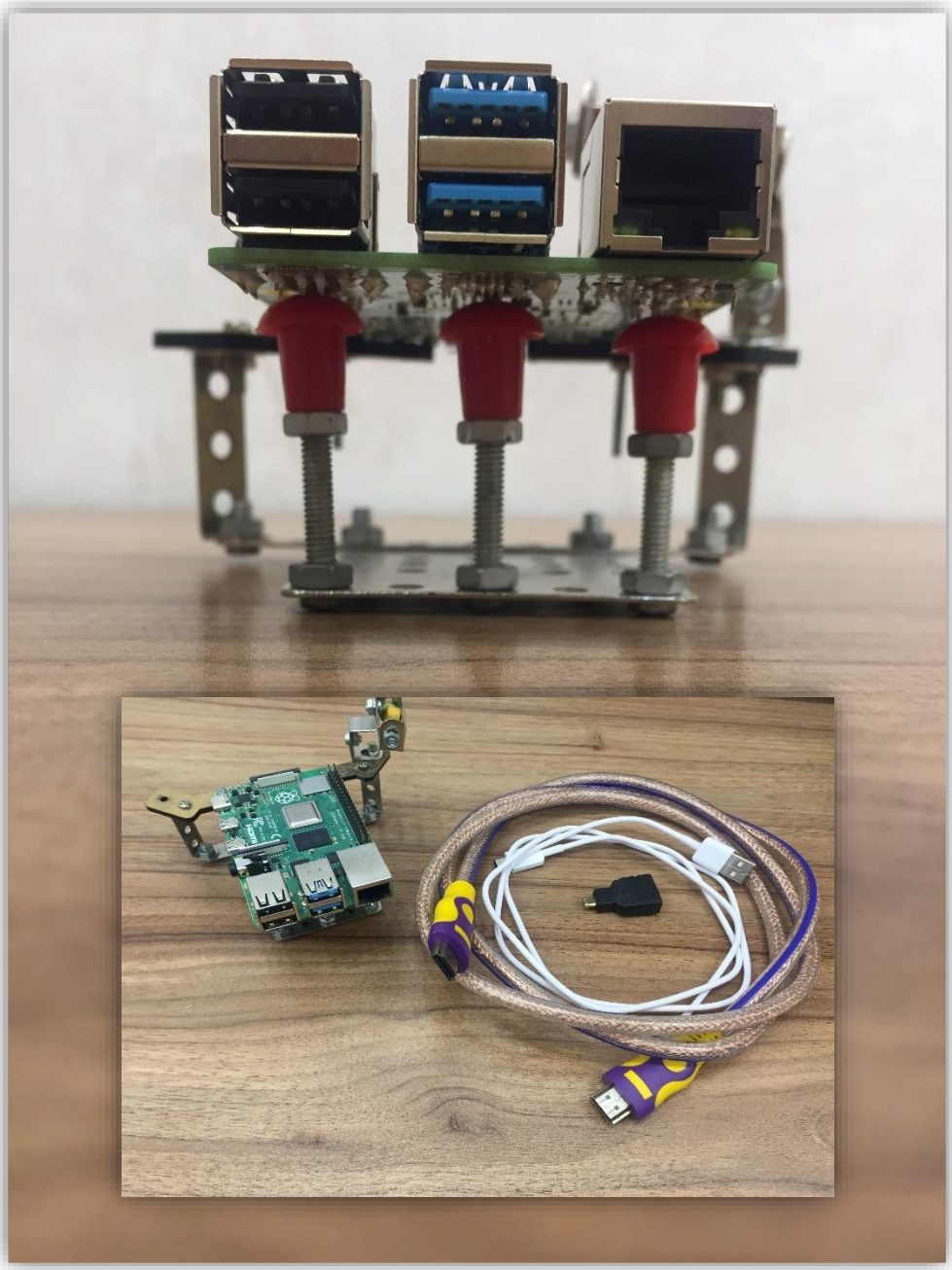
کتابخانه paho-mqtt :

paho-mqtt یک کتابخانه قدرتمند و منبع باز برای ارتباطات **MQTT** است. پروتکل MQTT یک پروتکل سبک و سریع برای ارتباطات اینترنت اشیا است که به دلیل سرعت بالا و کارایی بالا در انتقال داده‌ها بین دستگاه‌های مختلف، به ویژه در محیط‌های با پهنای باند محدود، به طور گسترده‌ای استفاده می‌شود.

در این پروژه، کتابخانه paho-mqtt برای **ارسال اطلاعات به یک اپلیکیشن اندرویدی** از طریق پروتکل MQTT به کار گرفته شده است. زمانی که رزبری پای یک چهره را شناسایی می‌کند، اطلاعات مربوط به این شناسایی مانند تصویر چهره و سایر داده‌ها از طریق این پروتکل به سرور یا مستقیم به اپلیکیشن اندرویدی ارسال می‌شود. این اپلیکیشن اندرویدی سپس این داده‌ها را دریافت و به کاربر نمایش می‌دهد.

پروتکل MQTT به دلیل طراحی کارآمد و سبک خود، برای کاربردهای متعددی در اینترنت اشیا (IoT) مناسب است، از جمله ارتباطات ماشین به ماشین(M2M) ، انتقال داده‌های سنسوری، و کنترل از راه

دور دستگاه‌ها. در این پروژه، استفاده از MQTT به رزبری پای این امکان را می‌دهد که داده‌های خود را به صورت پایدار و سریع به سیستم‌های دیگر منتقل کند، که برای کاربردهایی که نیاز به سرعت بالا و اطمینان از انتقال داده‌ها دارند، ایده‌آل است.



تحلیل کدهای پایتون

کد اول (تغییر سایز تصاویر دیتاست) :

این کد تصاویر را از یک مسیر مشخص می‌خواند و اندازه آن‌ها را به 256 x 256 تغییر داده و سپس در یک مسیر خروجی ذخیره می‌کند.

```
import cv2
import os

output_path = 'path/'
resized_path = 'path/'

target_size = (256, 256)

def resize_image(image_path, output_path, size):
    img = cv2.imread(image_path)
    resized_img = cv2.resize(img, size)
    cv2.imwrite(output_path, resized_img)
    print(f"Saved resized image to {output_path}")

counter = 1
for file_name in os.listdir(output_path):
    str = ""
    if counter < 10:
        str = f"person4_00{counter}.jpg"
    elif 10<=counter<=99:
        str = f"person4_0{counter}.jpg"
    else:
        str = f"person4_{counter}.jpg"
    image_path = os.path.join(output_path, file_name)
    output_file = os.path.join(resized_path, str)
    resize_image(image_path, output_file, target_size)
    counter = counter+1
```

در این بخش از پروژه، کدی نوشته شده که به منظور **تغییر سایز تصاویر** موجود در دیتاست استفاده می‌شود. تغییر سایز تصاویر یکی از مراحل مهم پیش‌پردازش در پروژه‌های پردازش تصویر و یادگیری ماشین است، زیرا این کار کمک می‌کند که تمام تصاویر به یک اندازه استاندارد درآیند و الگوریتم‌های یادگیری به‌طور یکنواخت‌تری آموزش ببینند. این مرحله همچنین باعث کاهش حجم داده‌ها و تسهیل در پردازش آن‌ها می‌شود.

هدف از تغییر سایز تصاویر :

- **هماهنگ‌سازی اندازه تصاویر:** در بسیاری از مدل‌های یادگیری ماشین، تصاویر ورودی باید دارای اندازه یکسان باشند. این هماهنگ‌سازی به مدل کمک می‌کند تا بدون توجه به ابعاد مختلف تصاویر، آن‌ها را به خوبی تحلیل کند.
- **کاهش حجم محاسبات :** کاهش اندازه تصاویر، تعداد پیکسل‌های هر تصویر را کاهش می‌دهد و در نتیجه، پردازش آن‌ها سریع‌تر انجام می‌شود. این موضوع در پروژه‌هایی که حجم داده‌ها زیاد است و نیاز به پردازش سریع دارند، بسیار حائز اهمیت است.
- **بهینه‌سازی استفاده از حافظه:** تصاویر بزرگ حجم زیادی از حافظه را اشغال می‌کنند. با تغییر سایز آن‌ها، می‌توان از حافظه به‌صورت بهینه‌تری استفاده کرد و ظرفیت بیشتری برای سایر فرآیندهای پروژه فراهم آورد.

نحوه عملکرد کد تغییر سایز تصاویر:

کد نوشته شده به صورت زیر عمل می‌کند:

- **خواندن تصاویر از مسیر مشخص:** ابتدا کد از یک مسیر داده شده (مثلاً یک پوشه روی سیستم) تصاویر موجود را می‌خواند. این پوشه می‌تواند شامل صدها یا هزاران تصویر باشد که باید پردازش شوند.

- **تغییر سایز تصاویر:** پس از خواندن هر تصویر، کد اندازه آن تصویر را به 256 x 256 پیکسل تغییر می‌دهد. این اندازه در بسیاری از پروژه‌های پردازش تصویر یک استاندارد معمول است که به‌خوبی بین دقت مدل و سرعت پردازش تعادل برقرار می‌کند.
- **ذخیره‌سازی تصاویر در مسیر خروجی:** تصاویر تغییر سایز داده شده در یک مسیر خروجی ذخیره می‌شوند. این مسیر می‌تواند یک پوشه جدید باشد که تمامی تصاویر پردازش شده در آن ذخیره می‌شوند.

توضیحات کد:

- **کتابخانه OpenCV :** برای پردازش و تغییر سایز تصاویر از کتابخانه OpenCV استفاده شده است. این کتابخانه یکی از پرکاربردترین ابزارها در زمینه پردازش تصویر است.
- کتابخانه**os :** os برای کار با سیستم فایل استفاده شده و به ما کمک می‌کند تا مسیرها را مدیریت کنیم و پوشه‌های جدید ایجاد کنیم.
- **cv2.imread :** این تابع برای خواندن تصاویر از مسیر ورودی به کار می‌رود.
- **cv2.resize :** این تابع تصویر خوانده شده را به ابعاد 256 x 256 تغییر سایز می‌دهد.
- **cv2.imwrite :** پس از تغییر سایز، تصویر جدید در مسیر خروجی ذخیره می‌شود.

کد دوم (آموزش دیتاست):

این کد تصاویر چهره را با استفاده از LBPHFaceRecognizer آموزش می‌دهد و مدل را در یک فایل yml. ذخیره می‌کند.

```
import cv2
import os
import numpy as np
import csv

dataset_path = 'dataset/'

face_cascade        =        cv2.CascadeClassifier(cv2.data.haarcascades        +
'haarcascade_frontalface_default.xml')

images = []
labels = []
labels_name =['Hamidreza']
with open('labels.csv') as csvfile:
    reader = csv.reader(csvfile)
    next(reader)
    for row in reader:
        image_path = os.path.join(dataset_path, row[0])
        label = row[1]

        image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        image = cv2.resize(image, (960, 1280))
        images.append(image)
        labels.append(int(label))

def read_training_data():
    return np.array(images), np.array(labels)

def train_recognizer():
    images, labels = read_training_data()
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.train(images, labels)
    recognizer.save('trained_model.yml')
    return recognizer

recognizer = train_recognizer()
print("the model trained")
```

این بخش از پروژه به آموزش مدل تشخیص چهره اختصاص دارد. برای این کار از کتابخانه OpenCV و الگوریتم LBPHFaceRecognizer استفاده می‌شود. الگوریتم LBPH (Local Binary Patterns Histogram) یکی از تکنیک‌های محبوب و موثر در حوزه‌ی تشخیص چهره است.

هدف از آموزش مدل تشخیص چهره:

- **تشخیص الگوهای محلی در تصاویر چهره:** الگوریتم LBPH قادر است الگوهای محلی مانند لبه‌ها و گوشه‌ها را در تصاویر تشخیص دهد. این ویژگی‌ها به عنوان نماینده‌های مهم چهره در نظر گرفته می‌شوند و مدل بر اساس این اطلاعات آموزش داده می‌شود.
- **ساخت مدل قابل استفاده برای تشخیص:** پس از آموزش مدل با استفاده از دیتاست موجود، مدل به شکلی تبدیل می‌شود که بتواند چهره‌ها را تشخیص داده و آن‌ها را به درستی شناسایی کند.
- **ذخیره‌سازی مدل برای استفاده‌های بعدی:** مدل آموزش‌دیده در یک فایل yml.ذخیره می‌شود. این فایل حاوی پارامترها و اطلاعاتی است که مدل برای تشخیص چهره نیاز دارد. با استفاده از این فایل، می‌توان مدل را در هر زمان و مکانی بارگذاری و استفاده کرد.

نحوه عملکرد کد آموزش دیتاست:

کد به صورت زیر عمل می‌کند:

- **خواندن تصاویر و برچسب‌ها از دیتاست:** در این مرحله، تصاویر چهره به همراه برچسب‌های مربوط به هر فرد از دیتاست خوانده می‌شود. این برچسب‌ها به مدل کمک می‌کنند تا بداند هر تصویر به کدام فرد تعلق دارد.

- **آموزش مدل با استفاده از تصاویر و برچسب‌ها:** تصاویر و برچسب‌ها به عنوان ورودی به مدل LBPHFaceRecognizer داده می‌شوند. مدل این داده‌ها را پردازش کرده و بر اساس ویژگی‌های استخراج شده، یک مدل تشخیص چهره ایجاد می‌کند.
- **ذخیره‌سازی مدل آموزش‌دیده:** پس از پایان آموزش، مدل در یک فایل yml. ذخیره می‌شود. این فایل به عنوان یک مدل آموزش‌دیده عمل می‌کند و در مراحل بعدی می‌توان از آن برای تشخیص چهره استفاده کرد.

کد سوم:(Real Stream)

این کد از دوربین به صورت زنده ویدئو دریافت کرده و چهره‌ها را شناسایی و تشخیص می‌دهد.

```
import cv2
from mqtt import start
from datetime import datetime

face_cascade            =            cv2.CascadeClassifier(cv2.data.haarcascades            +
'haarcascade_frontalface_default.xml')

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trained_model.yml')

labels_name =['Hamidreza']

address = "http://192.168.1.113:8080/video"
cap = cv2.VideoCapture(address)
counter = 0

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:
```

```
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 255), 2)
        face_roi = gray[y:y+h, x:x+w]

        label, confidence = recognizer.predict(face_roi)

        # ##########
        if confidence <= 50 and (label == 1 or label==3 or label==5):
            text = f'Person:{labels_name[int(label)-1]}-{confidence}'
            cv2.putText(frame, text, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1,cv2.LINE_AA)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            counter += 1
            if counter >= 100:
                print('mqtt message sent...')
                today =  datetime.now()
                cal_time = today.strftime("%Y-%m-%d %H:%M:%S")
                start(f"{labels_name[int(label)-1]}, {cal_time}, 200")
                counter = 0
        elif confidence <= 50 and (label == 2 or label==4):
            text = f'Person:{labels_name[int(label)-1]}-{confidence}'
            cv2.putText(frame, text, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8,(0, 0, 255), 1, cv2.LINE_AA)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            counter += 1
            if counter >= 100:
                print('mqtt message sent...')
                today =  datetime.now()
                cal_time = today.strftime("%Y-%m-%d %H:%M:%S")
                start(f"{labels_name[int(label)-1]}, {cal_time}, 400")
                counter = 0
        else:
            text = f'Unknown Person'
            cv2.putText(frame, text, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2.imshow('Face Recognition', frame)

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

cap.release()
cv2.destroyAllWindows()
```

این بخش از پروژه به کدی اختصاص دارد که وظیفه دریافت ویدئو به صورت زنده از دوربین و سپس شناسایی و تشخیص چهره‌ها را بر عهده دارد. کد از طریق OpenCV و استفاده از مدل آموزشی که قبلاً ساخته شده است، اقدام به شناسایی چهره‌ها در ویدئو می‌کند.

هدف از کد تست تشخیص چهره از استریم زنده:

- **دریافت استریم زنده از دوربین** :این کد به دوربین متصل شده و استریم ویدئویی زنده را به صورت فریم به فریم دریافت می‌کند.
- **شناسایی چهره‌ها در هر فریم ویدئو:** در هر فریم از ویدئو، چهره‌ها توسط مدل تشخیص چهره شناسایی و استخراج می‌شوند.
- **تشخیص هویت افراد شناسایی‌شده:** مدل آموزش‌دیده LBPH پس از شناسایی چهره‌ها، سعی می‌کند هویت آن‌ها را با داده‌های آموزشی مطابقت دهد و نام یا ID فرد را تشخیص دهد.
- **نمایش خروجی در ویدئو به صورت زنده:** فریم‌های پردازش‌شده دوباره به عنوان خروجی نمایش داده می‌شوند، به گونه‌ای که چهره‌های شناسایی‌شده با نام یا ID مشخص شده‌اند.

نحوه عملکرد کد:

این کد به صورت گام‌به‌گام به شرح زیر عمل می‌کند:

- **بارگذاری مدل تشخیص چهره از فایل ذخیره‌شده:** اولین گام، بارگذاری مدل آموزشی است که در مرحله قبل ذخیره شده بود. این مدل حاوی پارامترهای لازم برای تشخیص چهره است.
- **دریافت استریم ویدئویی زنده از دوربین:** کد با استفاده از cv2.VideoCapture(0)به دوربین متصل شده و شروع به دریافت استریم زنده ویدئو می‌کند.
- **پردازش هر فریم ویدئو برای شناسایی چهره‌ها:** هر فریم ویدئویی که از دوربین دریافت می‌شود، به یک تصویر پردازش‌شده تبدیل می‌شود که در آن چهره‌ها شناسایی و استخراج می‌شوند.

- **تشخیص هویت چهره‌های شناسایی‌شده:** پس از شناسایی چهره، مدل تلاش می‌کند تا هویت فرد را از طریق مقایسه با داده‌های آموزشی مشخص کند. اگر چهره شناسایی شده با یک چهره در دیتاست مطابقت داشته باشد، مدل ID یا نام فرد را نمایش می‌دهد.
- **نمایش خروجی در فریم‌های ویدئو:** خروجی پردازش‌شده شامل چهره‌های شناسایی‌شده و هویت‌های تشخیص‌داده‌شده دوباره به صورت یک فریم ویدئویی نمایش داده می‌شود.
- **خروج از برنامه:** کد به گونه‌ای نوشته شده که با فشردن کلید q توسط کاربر، استریم زنده قطع شده و برنامه پایان می‌یابد.

کد چهارم (ارسال پیام با MQTT):

این کد برای ارسال پیام به یک سرور MQTT استفاده می‌شود.

```
from paho.mqtt import client as mqtt
import time
MQTT_SERVER = "test.mosquitto.org"
MQTT_PORT = 8080
CLIENT_NAME = "Bannana"
MQTT_TOPIC = "testmqttshaf"

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe(MQTT_TOPIC)

def on_message(client, userdata, msg):
    print(f"Received message: {msg.topic} {msg.payload.decode()}")

client = mqtt.Client(CLIENT_NAME, transport="websockets", protocol=mqtt.MQTTv311)

client.on_connect = on_connect
client.on_message = on_message

client.connect(MQTT_SERVER, MQTT_PORT, 60)

def send_message(message):
```

```
    client.publish(MQTT_TOPIC, message, qos=2)

def start(message_to_mobile="hamidam"):

    client.loop_start()

    send_message(message_to_mobile)

    try:
        while True:
            time.sleep(2)
            break
    except KeyboardInterrupt:
        client.loop_stop()
        client.disconnect()
```

این بخش از پروژه به کدی اختصاص دارد که وظیفه ارسال پیام به یک سرور MQTT را بر عهده دارد.

هدف از کد ارسال پیام باMQTT :

1. **اتصال به یک سرور MQTT (که به عنوان Broker شناخته می‌شود):** ابتدا، کلاینت MQTT به یک سرور مرکزی متصل می‌شود که پیام‌ها را بین ناشران و مشترکان توزیع می‌کند.
2. **ارسال پیام به یک موضوع (Topic) خاص:** پیام‌ها به یک موضوع خاص ارسال می‌شوند. موضوعات برای دسته‌بندی و مسیریابی پیام‌ها در سیستم‌های MQTT استفاده می‌شوند.
3. **اطمینان از تحویل پیام به سرور MQTT :** این کد تضمین می‌کند که پیام‌ها با موفقیت به سرور ارسال می‌شوند.

ساختار کلی کد:

این کد به صورت گام‌به‌گام به شرح زیر عمل می‌کند:

- **ایجاد یک کلاینتMQTT :** ابتدا، یک کلاینت MQTT با استفاده از کتابخانه‌های مربوطه ایجاد می‌شود. این کلاینت توانایی برقراری ارتباط با یک سرور MQTT را دارد.
- **تنظیمات اولیه کلاینت:** برخی از پارامترهای اولیه مانند آدرس سرور(Broker) ، پورت، و مشخصات اتصال (مانند نام کاربری و رمز عبور، در صورت نیاز) تنظیم می‌شوند.
- **اتصال به سرور MQTT :**کلاینت به سرور MQTT متصل می‌شود. پس از اتصال موفقیت‌آمیز، کلاینت آماده ارسال پیام است.
- **ارسال پیام به یک موضوع خاص:** با استفاده از متد ()publish، پیام به موضوعی خاص در سرور MQTT ارسال می‌شود. این پیام می‌تواند داده‌هایی مانند وضعیت سنسور، اطلاعات هشدار یا هر نوع داده‌ای باشد که در سیستم IoT مورد نیاز است.
- **بستن اتصال و خاتمه عملیات:** پس از ارسال پیام، کلاینت اتصال خود را با سرور MQTT قطع می‌کند و برنامه خاتمه می‌یابد.

اپلیکیشن اندرویدی

در اپلیکیشن تنها قسمت مهم Log Activity است، که به دنبال اتصال به سرور MQTT و دریافت پیام‌ها است، سپس این پیام‌ها را پردازش کرده و بر اساس اطلاعات نمایش می‌دهد.

تحلیل کدهای جاوا

LOG Activity :

این کد مربوط به اتصال به سرور MQTT و دریافت پیام‌های ارسالی است. این پیام‌ها در اپلیکیشن اندرویدی دریافت شده و سپس پردازش می‌شوند.

```
import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;
import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;

public class MainActivity extends AppCompatActivity {

    private MqttAndroidClient mqttAndroidClient;
    private String brokerUrl = "tcp://broker.emqx.io:1883";
    private String clientId = MqttClient.generateClientId();
    private String subscriptionTopic = "raspberry/topic";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

            mqttAndroidClient = new MqttAndroidClient(getApplicationContext(),
brokerUrl, clientId);

        mqttAndroidClient.setCallback(new MqttCallback() {
```

```
            @Override
            public void connectionLost(Throwable cause) {
                Log.d("MQTT", "Connection lost: " + cause.getMessage());
            }

            @Override
             public void messageArrived(String topic, MqttMessage message) throws
Exception {
                String receivedMessage = new String(message.getPayload());
                Log.d("MQTT", "Message arrived: " + receivedMessage);
                // Update UI with the received message
            }

            @Override
            public void deliveryComplete(IMqttDeliveryToken token) {
                Log.d("MQTT", "Delivery complete");
            }
        });

        connectToMqttBroker();
    }

    private void connectToMqttBroker() {
        try {
            MqttConnectOptions options = new MqttConnectOptions();
            options.setAutomaticReconnect(true);
            options.setCleanSession(true);

            mqttAndroidClient.connect(options, null, new IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken asyncActionToken) {
                    Log.d("MQTT", "Connected to broker");
                    subscribeToTopic();
                }

                @Override
                   public void onFailure(IMqttToken asyncActionToken, Throwable
exception) {
                            Log.d("MQTT", "Failed to connect to broker: " +
exception.getMessage());
                }
            });

        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }

    private void subscribeToTopic() {
        try {
            mqttAndroidClient.subscribe(subscriptionTopic, 0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

تعریف و ایجاد یک کلاینت MQTT : این بخش از کد، آدرس سرور MQTT (broker) و شناسه کلاینت (clientId) را تعریف می‌کند. سپس با استفاده از این اطلاعات، یک کلاینت MQTT با استفاده از کلاس MqttClient ایجاد می‌شود. MemoryPersistence به این معناست که پیام‌های ارسال شده یا دریافت شده در حافظه موقت ذخیره می‌شوند و با خاموش شدن برنامه از بین می‌روند.

```
String broker = "tcp://broker.hivemq.com:1883";
String clientId = "JavaSample";
MemoryPersistence persistence = new MemoryPersistence();
MqttClient sampleClient = new MqttClient(broker, clientId, persistence);
```

این خط کد کلاینت را به سرور MQTT متصل می‌کند. پس از این مرحله، کلاینت آماده ارسال و دریافت پیام‌ها می‌شود.

```
sampleClient.connect();
```

این بخش از کد، کلاینت را در موضوعی به نام "test/topic" مشترک می‌کند. هر پیامی که در این موضوع منتشر شود، توسط کلاینت دریافت می‌شود و محتوای آن در کنسول چاپ می‌شود. در اینجا از یک تابع برگشتی (callback) استفاده شده است که به محض دریافت پیام فراخوانی می‌شود و محتوای پیام را پردازش می‌کند.

```
sampleClient.subscribe("test/topic", (topic, message) -> {
    System.out.println("Message received:\t" + new String(message.getPayload()));
});
```

اتصال به سرورMQTT :

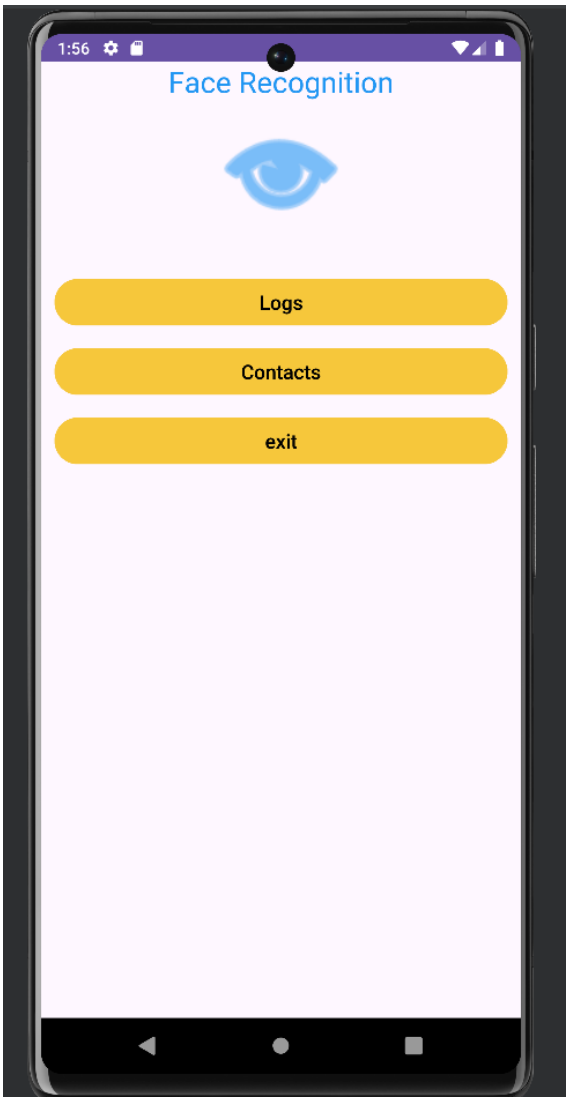
- برای ارتباط و تبادل اطلاعات بین دستگاه‌ها، ابتدا باید به سرور MQTT متصل شد. این سرور مانند یک واسطه عمل می‌کند که پیام‌ها را از ناشر (publisher) دریافت کرده و به مشترکان (subscribers)منتقل می‌کند.

اشتراک در موضوعات:

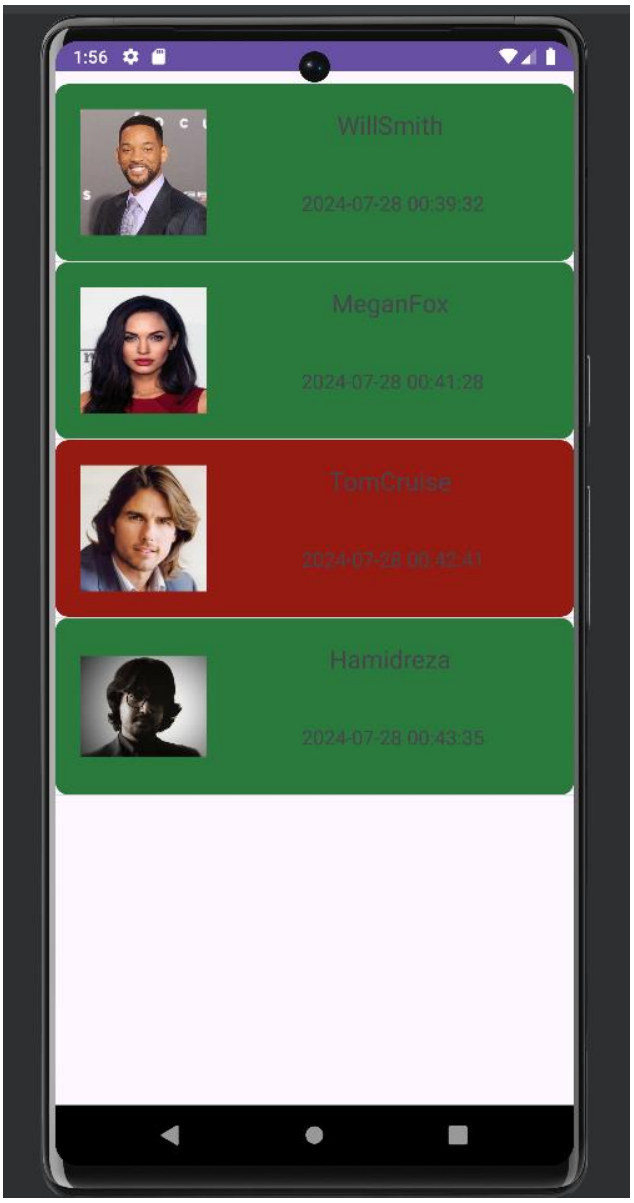
- موضوعات در MQTT به عنوان یک ساختار سلسله مراتبی برای دسته‌بندی پیام‌ها عمل می‌کنند. هر کلاینت می‌تواند در یک یا چند موضوع مشترک شود و پیام‌های مربوط به آن موضوعات را دریافت کند. در این کد، موضوع "test/topic" برای دریافت پیام‌ها انتخاب شده است.

پردازش پیام‌های دریافتی:

- پس از دریافت پیام‌ها، باید آن‌ها را پردازش کرد. در این کد، پیام‌های دریافتی در کنسول چاپ می‌شوند، اما در یک اپلیکیشن واقعی، این پیام‌ها ممکن است برای بروزرسانیUI ، ذخیره در دیتابیس، یا سایر پردازش‌های مهم استفاده شوند.



قسمت اصلی اپلیکیشن Activity log است که با بروکر اتصال برقرار کرده و اطلاعات را دریافت می‌کند.



در قسمت لاگ علاوه بر عکس شخص و اسم، تاریخ و ساعت دقیق تشخیص چهره و ارسال اطلاعات نیز درج می‌شود. قسمت contacts فقط مخاطبین تعریف شده را نشان می‌دهد و دکمه exit تمام فرآیندهای مربوط به اپلیکیشن را kill می‌کند و از برنامه خارج می‌شود.

نتیجه‌گیری کلی از پروژه

این پروژه با هدف بررسی و پیاده‌سازی، تشخیص چهره و لیبل گذاری روی تصویر، ارتباطات از طریق پروتکل MQTT در محیط برنامه‌نویسی جاوا و پایتون بر روی Raspberry Pi انجام شده است.

نقاط قوت پروژه:

1. **کارایی و مقیاس‌پذیری بالا:**
   - استفاده از پروتکل MQTT به دلیل سبک بودن و مصرف کم پهنای باند، عملکرد بهینه و مقیاس‌پذیری بالایی را برای سیستم فراهم کرده است. این پروتکل به خوبی از ارتباطات همزمان بین تعداد زیادی دستگاه پشتیبانی می‌کند که برای پروژه‌های بزرگ و پیچیده یک مزیت مهم به شمار می‌آید.
2. **پیاده‌سازی ساده و موثر:**
   - کدهای جاوا و پایتون در این پروژه به شکلی ساده و قابل فهم پیاده‌سازی شده‌اند که به توسعه‌دهندگان امکان می‌دهد به راحتی این سیستم را گسترش داده و با تغییرات مختلف سازگار کنند.
3. **پشتیبانی از کاربردهای مختلف:**
   - این پروژه به دلیل انعطاف‌پذیری بالای پروتکلMQTT ، قابلیت تطبیق با طیف گسترده‌ای از کاربردها، از جمله ارتباطات اینترنت اشیاء، سیستم‌های توزیع‌شده، و انتقال داده‌های بلادرنگ را دارد.
4. **سبک بودن مدل آموزش دیده:**
   - مدل آموزش دیده که روی رزبری وظیفه تشخیص چهره و لیبل گذاری را دارد سنگین نبوده و باعث کاهش بازدهی سیستم نمی‌شود.

نقاط ضعف پروژه:

1. امنیت:

- یکی از نقاط ضعف مهم پروژه عدم توجه کافی به مسائل امنیتی است. پروتکل MQTT به طور پیش‌فرض سطح امنیتی بالایی ندارد و نیاز است تا مکانیزم‌های امنیتی مناسبی مانند رمزنگاری ارتباطات و احراز هویت قوی به آن اضافه شود.

2. محدودیت‌های مدیریت خطا:

- سیستم مدیریت خطا در این پروژه به اندازه کافی توسعه نیافته است. در صورت بروز مشکلاتی مانند قطع ارتباط با سرور یا عدم دریافت پیام‌ها، بهتر است مکانیزم‌های پیشرفته‌تری برای مدیریت و بازیابی از خطاها پیاده‌سازی شود.

3. عدم وجود مستندات کامل:

- مستندات پروژه به اندازه کافی جامع و کامل نیستند که می‌تواند مشکلاتی را برای توسعه‌دهندگان آینده در فهم و نگهداری سیستم ایجاد کند.

4. عدم وجود منابع کافی بر روی Raspberry Pi :

- به دلیل محدودیت‌های سخت افزاری امکان دارد سیستم به راحتی پس از مدتی کرش کند یا توانایی کار کردن به مدت طولانی را نداشته باشد.

5. دقت پایین الگوریتم:

- به دلیل عیب ذکر شده در بالا، نمیتوان از الگوریتمی مانند ResNet که دقتی در حدود 95% دارد، در این پروژه با این نوع سخت افزار به کار گرفت.

پیشنهادات برای بهبود پروژه:

1. **افزایش امنیت:**

- بهبود مکانیزم‌های امنیتی با اضافه کردن قابلیت‌هایی مانند SSL/TLS برای رمزنگاری داده‌ها و استفاده از احراز هویت قوی برای اطمینان از امنیت ارتباطات ضروری است.

2. **افزایش مقیاس‌پذیری:**

- بهبود ساختار کدها و استفاده از معماری‌های مقیاس‌پذیرتر می‌تواند به پروژه کمک کند تا در مقیاس‌های بزرگ‌تر نیز به خوبی عمل کند.

3. **گسترش مستندات و آموزش‌ها:**

- ایجاد مستندات جامع‌تر و آموزشی می‌تواند کمک بزرگی به توسعه‌دهندگان آینده برای فهم بهتر و گسترش سیستم باشد.

4. **اضافه کردن قابلیت‌های مانیتورینگ و گزارش‌دهی:**

- اضافه کردن ابزارها و قابلیت‌های مانیتورینگ و گزارش‌دهی برای مشاهده و بررسی عملکرد سیستم در لحظه می‌تواند به بهبود مدیریت و نگهداری سیستم کمک کند.