## ****整体工作流程****

**初始化**：

· Flask 启动后监听视频流，M5StickC Plus 连接到 Wi-Fi。

**人脸检测激活 M5**：

· Flask 检测到人脸后，通过 /activate 通知 M5 激活。

**M5 触发拍照**：

· 用户按下 M5 的 A 按钮，M5 发送 /take\_photo 请求到 Flask。

**Flask 处理拍照逻辑**：

· Flask 抓取当前视频帧并保存为图像文件，同时上传到服务器。

### ****运作模式****

**人脸检测**：

* 1. Flask 服务器持续监听 ESP32-CAM 的视频流。
  2. 使用 OpenCV 的人脸检测模型检测人脸，并在检测到人脸后激活 M5StickC Plus。

**拍照请求**：

* 1. 服务器提供一个 /take\_photo 路由，用于接收 M5 的拍照请求。
  2. 当收到请求时，从视频流抓取当前帧并保存为图像文件，同时将其上传到服务器存储。

### ****主要代码逻辑****

#### ****1. 视频流初始化****

cap = cv2.VideoCapture(url) # 打开 ESP32-CAM 的视频流

if not cap.isOpened():

print("无法打开视频流，请检查地址或摄像头状态！")

exit()

**2.人脸检测和激活 M5**

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

if len(faces) > 0 and current\_time - last\_trigger\_time > trigger\_interval:

response = requests.get("http://192.168.x.x/activate") # 通知 M5 激活

· 使用 OpenCV 的 Haar 模型对视频流帧进行人脸检测。

· 检测到人脸后，通过 HTTP 请求通知 M5 激活。

**3. 接收拍照请求并保存照片**

@app.route('/take\_photo', methods=['GET'])

def take\_photo():

ret, frame = cap.read() # 从视频流读取一帧

filename = os.path.join(UPLOAD\_FOLDER, f"photo\_{timestamp}.jpg")

cv2.imwrite(filename, frame) # 保存帧为图像文件

· 监听 /take\_photo 路由，接收 M5 的拍照请求。

· 抓取当前视频帧并保存为图像文件。

**4.上传照片到服务器**

with open(local\_filename, 'rb') as f:

response = requests.post(upload\_url, files={'photo': f}) # 上传照片

通过 HTTP POST 请求，将拍摄的照片上传到远程服务器进行存储。

**Flask 代码的作用**

* 监听和处理 ESP32-CAM 的视频流。
* 实现人脸检测并激活 M5StickC Plus。
* 接收 M5 的拍照请求并保存照片，同时上传到服务器。

## ****M5StickC Plus 代码说明****

### ****运作模式****

### ****Wi-Fi 连接****：

* 1. M5 连接到 Wi-Fi 网络，并显示 IP 地址。
  2. 在未激活状态时，屏幕背景为红色，显示 "Not Activated" 和设备 IP。

**激活状态**：

* 1. 当 Flask 服务器通过 /activate 路由通知 M5 激活时，背景变为绿色，显示 "Press A to Take Photo"。

**拍照请求**：

* 1. 用户按下 A 按钮，M5 向 Flask 服务器发送 /take\_photo 请求，触发拍照和照片上传逻辑。

**主要代码逻辑**

1. Wi-Fi 连接

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(xxx);

Serial.print(".");

}

M5.Lcd.println(WiFi.localIP().toString()); // 显示 IP 地址

· M5 连接到指定的 Wi-Fi 网络。

· 屏幕显示设备的 IP 地址。

1. 激活状态处理

server.on("/activate", HTTP\_GET, activateHandler);

void activateHandler(AsyncWebServerRequest \*request) {

isActivated = true;

M5.Lcd.fillScreen(GREEN);

M5.Lcd.println("Press A to Take Photo");

request->send(200, "M5 Activated");

}

· M5 通过 /activate 路由接收 Flask 的激活通知。

· 更新屏幕为绿色背景，显示 "Press A to Take Photo"。

1. 按钮触发拍照请求

if (M5.BtnA.wasPressed()) {

HTTPClient http;

http.begin("http://192.168.0.xxx:xxx/take\_photo"); // 发送拍照请求

int httpCode = http.GET();

Serial.println(httpCode > 0 ? "拍照请求成功" : "拍照请求失败");

}

· 按下 A 按钮时，M5 通过 HTTP GET 请求触发 Flask 的 /take\_photo 路由。

· 日志通过串口打印拍照请求的结果。

### ****M5StickC Plus 代码的作用****

* 连接到 Wi-Fi 网络，并监听激活请求。
* 激活后允许用户通过按钮发送拍照请求。
* 将拍照请求发送到 Flask 服务器，由服务器执行拍照逻辑。

### ****适用场景****

* **智能监控系统**：当检测到人脸时激活拍照功能。
* **IoT 自动化应用**：M5 作为简单控制设备，触发服务器执行复杂逻辑。
* **分布式网络架构**：通过 Flask 服务器统一管理摄像头和其他 IoT 设备。