#include "esp\_camera.h"

#include <WiFi.h>

#include <base64.h>

#include <PubSubClient.h>  // Thêm thư viện MQTT

#define CAMERA\_MODEL\_AI\_THINKER

#include "camera\_pins.h"

// WiFi credentials

const char\* ssid = "IoT";

const char\* password = "234567Cn";

// MQTT broker details

const char\* mqtt\_server = "192.168.0.100";

const int mqtt\_port = 1883;

const char\* mqtt\_topic = "esp32/cam/image"; // Chủ đề để gửi ảnh

WiFiClient espClient;

PubSubClient client(espClient);

void connectToWiFi() {

  WiFi.begin(ssid, password);

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

    delay(500);

    Serial.print(".");

  }

  Serial.println("\nWiFi connected!");

}

void reconnectMQTT() {

  while (!client.connected()) {

    if (client.connect("ESP32Client")) {

      Serial.println("Connected to MQTT Broker!");

    } else {

      Serial.print("Failed to connect to MQTT. Retrying...");

      delay(5000);

    }

  }

}

void setupCamera() {

  camera\_config\_t config;

  config.ledc\_channel = LEDC\_CHANNEL\_0;

  config.ledc\_timer = LEDC\_TIMER\_0;

  config.pin\_d0 = Y2\_GPIO\_NUM;

  config.pin\_d1 = Y3\_GPIO\_NUM;

  config.pin\_d2 = Y4\_GPIO\_NUM;

  config.pin\_d3 = Y5\_GPIO\_NUM;

  config.pin\_d4 = Y6\_GPIO\_NUM;

  config.pin\_d5 = Y7\_GPIO\_NUM;

  config.pin\_d6 = Y8\_GPIO\_NUM;

  config.pin\_d7 = Y9\_GPIO\_NUM;

  config.pin\_xclk = XCLK\_GPIO\_NUM;

  config.pin\_pclk = PCLK\_GPIO\_NUM;

  config.pin\_vsync = VSYNC\_GPIO\_NUM;

  config.pin\_href = HREF\_GPIO\_NUM;

  config.pin\_sscb\_sda = SIOD\_GPIO\_NUM;

  config.pin\_sscb\_scl = SIOC\_GPIO\_NUM;

  config.pin\_pwdn = PWDN\_GPIO\_NUM;

  config.pin\_reset = RESET\_GPIO\_NUM;

  config.xclk\_freq\_hz = 20000000;

  config.frame\_size = FRAMESIZE\_QVGA;

  config.pixel\_format = PIXFORMAT\_JPEG;

  config.fb\_location = CAMERA\_FB\_IN\_PSRAM;

  config.jpeg\_quality = 12;

  config.fb\_count = 1;

  if (esp\_camera\_init(&config) != ESP\_OK) {

    Serial.println("Camera init failed");

    return;

  }

  Serial.println("Camera ready!");

}

void sendEncodedImage() {

  camera\_fb\_t\* fb = esp\_camera\_fb\_get();

  if (!fb) {

    Serial.println("Camera capture failed");

    return;

  }

  // Encode image to Base64

  String encodedImage = base64::encode(fb->buf, fb->len);

  // Chia nhỏ Base64 thành các phần nhỏ và gửi qua MQTT

  int partSize = 200; // Kích thước mỗi phần (tùy chỉnh)

  int totalParts = (encodedImage.length() + partSize - 1) / partSize;

  for (int i = 0; i < totalParts; i++) {

    String part = encodedImage.substring(i \* partSize, (i + 1) \* partSize);

    String message = String(i + 1) + "/" + String(totalParts) + ":" + part;

    client.publish(mqtt\_topic, message.c\_str());

    delay(100); // Thời gian giữa các phần gửi

  }

  // Gửi thông báo kết thúc

  client.publish(mqtt\_topic, "end");

  esp\_camera\_fb\_return(fb);

}

void setup() {

  Serial.begin(115200);

  connectToWiFi(); // Kết nối Wi-Fi

  client.setServer(mqtt\_server, mqtt\_port); // Cấu hình MQTT

  reconnectMQTT(); // Kết nối với broker MQTT

  setupCamera(); // Khởi tạo camera

}

void loop() {

  if (!client.connected()) {

    reconnectMQTT(); // Kiểm tra và kết nối lại MQTT nếu mất kết nối

  }

  client.loop(); // Xử lý các tin nhắn MQTT

  // Capture and encode image periodically

  sendEncodedImage();

  delay(5000); // Gửi ảnh mỗi 10 giây

}

Sever

import base64  
from flask import Flask, render\_template, send\_from\_directory, jsonify  
import paho.mqtt.client as mqtt  
import os  
from datetime import datetime  
import threading  
  
# Các thông số MQTT  
MQTT\_BROKER = "192.168.0.100"  
MQTT\_PORT = 1883  
MQTT\_TOPIC = "esp32/cam/image"  
  
# Dữ liệu buffer để lưu ảnh  
image\_data = ""  
total\_parts = 0  
received\_parts = 0  
UPLOAD\_FOLDER = "./uploads" # Thư mục lưu ảnh  
os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)  
  
# Biến lưu tên ảnh hiện tại để hiển thị trong web  
current\_image = ""  
  
def on\_connect(client, userdata, flags, rc):  
 print(f"Connected with result code {rc}")  
 client.subscribe(MQTT\_TOPIC)  
  
def on\_message(client, userdata, msg):  
 global image\_data, total\_parts, received\_parts, current\_image  
 message = msg.payload.decode()  
  
 if message == "end":  
 print("End of image transmission received.")  
 print("Final concatenated image data (Base64):")  
 print(image\_data)  
  
 timestamp = datetime.now().strftime("%Y%m%d\_%H%M%S")  
 file\_name = f"image\_{timestamp}.jpg"  
 file\_path = os.path.join(UPLOAD\_FOLDER, file\_name)  
  
 if not image\_data:  
 print("No image data to decode!")  
 return  
 try:  
 image\_bytes = base64.b64decode(image\_data)  
 with open(file\_path, "wb") as img\_file:  
 img\_file.write(image\_bytes)  
 print(f"Image saved as {file\_name}")  
 current\_image = file\_name  
 except Exception as e:  
 print(f"Error decoding base64 data: {e}")  
 image\_data = ""  
 received\_parts = 0  
 total\_parts = 0  
 else:  
 try:  
 if '/' in message:  
 index, rest = message.split('/', 1)  
 index = int(index)  
 part\_data = rest.split(':', 1)[1]  
 if received\_parts == 0:  
 total\_parts = int(rest.split('/', 1)[0].split(":")[0])  
 image\_data += part\_data  
 received\_parts += 1  
 # print(f"Received part {index}/{total\_parts}")  
 else:  
 print(f"Received invalid message: {message}")  
 except Exception as e:  
 print(f"Error processing message: {e}")  
  
app = Flask(\_\_name\_\_)  
  
@app.route('/')  
def index():  
 return render\_template('index.html')  
  
@app.route('/uploads/<filename>')  
def uploaded\_file(filename):  
 return send\_from\_directory(UPLOAD\_FOLDER, filename)  
  
@app.route('/current-image')  
def current\_image\_route():  
 return jsonify({'image\_name': current\_image})  
  
def mqtt\_loop():  
 client = mqtt.Client()  
 client.on\_connect = on\_connect  
 client.on\_message = on\_message  
 client.connect(MQTT\_BROKER, MQTT\_PORT, 60)  
 client.loop\_start()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 mqtt\_thread = threading.Thread(target=mqtt\_loop)  
 mqtt\_thread.start()  
  
 app.run(host="0.0.0.0", port=5000, debug=True)