#include "esp\_camera.h"

#include "FS.h"

#include "SD\_MMC.h"

#include "WiFi.h"

#include "HTTPClient.h"

#include "base64.h"

#include "time.h"

#include <Preferences.h>

#include <esp\_now.h>

#include <esp\_wifi.h>

// Define the camera model

#define CAMERA\_MODEL\_AI\_THINKER

#include "camera\_pins.h"

// Replace with your network credentials

const char\* ssid = "Pixel\_6pro";

const char\* password = "12345687";

// Replace with your Firebase project details

const char\* firebaseHost = "l1project-681ca.firebaseio.com";

const char\* firebaseAuth = "OEy9Z13kNtQjxoZ6YQ0v5rh9Wc9ESDhUqUhBsFKV";

const char\* firebaseStorageBucket = "l1project-681ca.appspot.com";

// Time setup

const char\* ntpServer = "pool.ntp.org";

const long gmtOffset\_sec = 5 \* 3600; // GMT+5 hours

const int daylightOffset\_sec = 30 \* 60;

// Number of images to capture per attempt

const int numImagesPerAttempt = 10;

// Global variable to track attempt number

int folderNumber = 0;

// Preferences object to store the attempt number persistently

Preferences preferences;

// Flag to indicate if image capture should start

bool startCapture = false;

void startCamera() {

  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\_sccb\_sda = SIOD\_GPIO\_NUM;

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

  config.pin\_pwdn = PWDN\_GPIO\_NUM;

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

  config.xclk\_freq\_hz = 20000000;

  config.pixel\_format = PIXFORMAT\_JPEG;

  if (psramFound()) {

    config.frame\_size = FRAMESIZE\_UXGA;

    config.jpeg\_quality = 10;

    config.fb\_count = 2;

  } else {

    config.frame\_size = FRAMESIZE\_SVGA;

    config.jpeg\_quality = 12;

    config.fb\_count = 1;

  }

  esp\_err\_t err = esp\_camera\_init(&config);

  if (err != ESP\_OK) {

    Serial.printf("Camera init failed with error 0x%x\n", err);

    return;

  }

}

void onDataRecv(const esp\_now\_recv\_info \*recvInfo, const uint8\_t \*incomingData, int len) {

  Serial.println("Signal received to start capturing images.");

  startCapture = true;

}

void setup() {

  Serial.begin(115200);

  // Initialize WiFi

  WiFi.mode(WIFI\_STA);

  WiFi.disconnect();

  delay(100);

  // Initialize ESP-NOW

  if (esp\_now\_init() != ESP\_OK) {

    Serial.println("Error initializing ESP-NOW");

    return;

  }

  esp\_now\_register\_recv\_cb(onDataRecv);

  if (!SD\_MMC.begin()) {

    Serial.println("Card Mount Failed");

    return;

  }

  uint8\_t cardType = SD\_MMC.cardType();

  if (cardType == CARD\_NONE) {

    Serial.println("No SD card attached");

    return;

  }

  WiFi.begin(ssid, password);

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

    delay(1000);

    Serial.println("Connecting to WiFi...");

  }

  Serial.println("Connected to WiFi");

  startCamera();

  configTime(gmtOffset\_sec, daylightOffset\_sec, ntpServer);

  struct tm timeinfo;

  if (!getLocalTime(&timeinfo)) {

    Serial.println("Failed to obtain time");

  } else {

    Serial.println("Time obtained successfully");

  }

  // Initialize Preferences and get the attempt number

  preferences.begin("camera", false);

  folderNumber = preferences.getInt("folderNumber", 0);

  preferences.end();

  Serial.println("Setup complete. Waiting for signal to capture images...");

}

void loop() {

  if (startCapture) {

    startCapture = false;

    // Generate folder name based on folder number

    String folderName = getFolderName(folderNumber);

    fs::FS &fs = SD\_MMC;

    fs.mkdir("/" + folderName);

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

      captureAndSaveImage(folderName, i);

      delay(10);

    }

    uploadFolderToFirebase(folderName);

    // Increment folder number for the next session and save it persistently

    folderNumber++;

    preferences.begin("camera", false);

    preferences.putInt("folderNumber", folderNumber);

    preferences.end();

    Serial.println("Image capture and upload complete. Waiting for next signal...");

  }

}

String getFolderName(int folderNum) {

  return String(folderNum);

}

void captureAndSaveImage(const String& folderName, int imageIndex) {

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

  if (!fb) {

    Serial.println("Camera capture failed");

    return;

  }

  String path = "/" + folderName + "/image" + String(imageIndex) + ".jpg";

  fs::FS &fs = SD\_MMC;

  File file = fs.open(path.c\_str(), FILE\_WRITE);

  if (!file) {

    Serial.println("Failed to open file in writing mode");

  } else {

    file.write(fb->buf, fb->len);

    Serial.printf("Saved file to path: %s\n", path.c\_str());

  }

  file.close();

  esp\_camera\_fb\_return(fb);

}

void uploadImageToFirebaseStorage(const uint8\_t \*image, size\_t len, const String& folderName, const String& fileName) {

  String url = "https://firebasestorage.googleapis.com/v0/b/" + String(firebaseStorageBucket) + "/o/" + folderName + "%2F" + fileName + "?uploadType=media&name=" + folderName + "/" + fileName;

  HTTPClient http;

  http.begin(url);

  http.addHeader("Content-Type", "image/jpeg");

  int httpResponseCode = http.POST((uint8\_t\*)image, len);

  if (httpResponseCode > 0) {

    String response = http.getString();

    Serial.println(httpResponseCode);

    Serial.println(response);

    int start = response.indexOf("\"downloadTokens\":\"") + 18;

    int end = response.indexOf("\"", start);

    String downloadURL = "https://firebasestorage.googleapis.com/v0/b/" + String(firebaseStorageBucket) + "/o/" + folderName + "%2F" + fileName + "?alt=media&token=" + response.substring(start, end);

    uploadMetadataToFirebaseDatabase(downloadURL, folderName, fileName);

  } else {

    Serial.print("Error on sending POST: ");

    Serial.println(httpResponseCode);

    Serial.println(http.errorToString(httpResponseCode).c\_str());

  }

  http.end();

}

void uploadMetadataToFirebaseDatabase(const String& downloadURL, const String& folderName, const String& fileName) {

  String url = "https://" + String(firebaseHost) + "/images.json?auth=" + String(firebaseAuth);

  HTTPClient http;

  http.begin(url);

  http.addHeader("Content-Type", "application/json");

  String jsonPayload = "{\"url\":\"" + downloadURL + "\", \"folder\":\"" + folderName + "\", \"file\":\"" + fileName + "\"}";

  int httpResponseCode = http.POST(jsonPayload);

  if (httpResponseCode > 0) {

    String response = http.getString();

    Serial.println(httpResponseCode);

    Serial.println(response);

  } else {

    Serial.print("Error on sending POST: ");

    Serial.println(httpResponseCode);

    Serial.println(http.errorToString(httpResponseCode).c\_str());

  }

  http.end();

}

void uploadFolderToFirebase(const String& folderName) {

  fs::FS &fs = SD\_MMC;

  File root = fs.open("/" + folderName);

  if (!root) {

    Serial.printf("Failed to open directory: %s\n", folderName.c\_str());

    return;

  }

  // Iterate over all files in the folder and upload them

  File file = root.openNextFile();

  while (file) {

    size\_t fileSize = file.size();

    uint8\_t \*buffer = (uint8\_t\*) malloc(fileSize);

    file.read(buffer, fileSize);

    String fileName = file.name();

    fileName = fileName.substring(fileName.lastIndexOf('/') + 1); // Extract just the filename

    uploadImageToFirebaseStorage(buffer, fileSize, folderName, fileName);

    free(buffer);

    file.close();

    file = root.openNextFile();

  }

  Serial.printf("Folder %s has been uploaded to Firebase.\n", folderName.c\_str());

  // Remove all files in the folder

  root.rewindDirectory(); // Ensure we're at the beginning

  file = root.openNextFile();

  while (file) {

    fs.remove(file.name());

    file.close();

    file = root.openNextFile();

  }

  // Remove the folder itself

  fs.rmdir("/" + folderName);

  Serial.printf("Folder %s has been deleted from SD card.\n", folderName.c\_str());

}