#include "esp\_camera.h"

#include <WiFi.h>

#include<ESP\_Mail\_Client.h>

#define ssid "AndroidAPA97A"

#define password "rohini19"

#define SMTP server "smtp.gmail.com"

#define SMTP\_Port 465

#define sender\_email "e.rohininew@gmail.com"

#define sender\_password "samsunggalaxya03s"

#define Recipient\_email "iot2automation@gmail.com"

#define Recipient\_name "rohivaish19"

//

// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality

//            Ensure ESP32 Wrover Module or other board with PSRAM is selected

//            Partial images will be transmitted if image exceeds buffer size

//

//            You must select partition scheme from the board menu that has at least 3MB APP space.

//            Face Recognition is DISABLED for ESP32 and ESP32-S2, because it takes up from 15

//            seconds to process single frame. Face Detection is ENABLED if PSRAM is enabled as well

// ===================

// Select camera model

// ===================

//#define CAMERA\_MODEL\_WROVER\_KIT // Has PSRAM

//#define CAMERA\_MODEL\_ESP\_EYE // Has PSRAM

//#define CAMERA\_MODEL\_ESP32S3\_EYE // Has PSRAM

//#define CAMERA\_MODEL\_M5STACK\_PSRAM // Has PSRAM

//#define CAMERA\_MODEL\_M5STACK\_V2\_PSRAM // M5Camera version B Has PSRAM

//#define CAMERA\_MODEL\_M5STACK\_WIDE // Has PSRAM

//#define CAMERA\_MODEL\_M5STACK\_ESP32CAM // No PSRAM

//#define CAMERA\_MODEL\_M5STACK\_UNITCAM // No PSRAM

#define CAMERA\_MODEL\_AI\_THINKER // Has PSRAM

//#define CAMERA\_MODEL\_TTGO\_T\_JOURNAL // No PSRAM

//#define CAMERA\_MODEL\_XIAO\_ESP32S3 // Has PSRAM

// \*\* Espressif Internal Boards \*\*

//#define CAMERA\_MODEL\_ESP32\_CAM\_BOARD

//#define CAMERA\_MODEL\_ESP32S2\_CAM\_BOARD

//#define CAMERA\_MODEL\_ESP32S3\_CAM\_LCD

//#define CAMERA\_MODEL\_DFRobot\_FireBeetle2\_ESP32S3 // Has PSRAM

//#define CAMERA\_MODEL\_DFRobot\_Romeo\_ESP32S3 // Has PSRAM

#include "camera\_pins.h"

// ===========================

// Enter your WiFi credentials

const char\* ssid = "AndroidAPA97A";

const char\* password = "rohini19";

void startCameraServer();

void setupLedFlash(int pin);

SMTPSession smtp;

void setup() {

  Serial.begin(115200);

  WiFi.begin(ssid, password);

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

  { Serial.print(".");

    delay(200);

  }

  Serial.println("");

  Serial.println("WiFi connected.");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

  Serial.println();

  smtp.debug(1);

  Serial.setDebugOutput(true);

  Serial.println();

  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.frame\_size = FRAMESIZE\_UXGA;

  config.pixel\_format = PIXFORMAT\_JPEG; // for streaming

  //config.pixel\_format = PIXFORMAT\_RGB565; // for face detection/recognition

  config.grab\_mode = CAMERA\_GRAB\_WHEN\_EMPTY;

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

  config.jpeg\_quality = 12;

  config.fb\_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality

  //                      for larger pre-allocated frame buffer.

  if(config.pixel\_format == PIXFORMAT\_JPEG){

    if(psramFound()){

      config.jpeg\_quality = 10;

      config.fb\_count = 2;

      config.grab\_mode = CAMERA\_GRAB\_LATEST;

    } else {

      // Limit the frame size when PSRAM is not available

      config.frame\_size = FRAMESIZE\_SVGA;

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

    }

  } else {

    // Best option for face detection/recognition

    config.frame\_size = FRAMESIZE\_240X240;

#if CONFIG\_IDF\_TARGET\_ESP32S3

    config.fb\_count = 2;

#endif

  }

#if defined(CAMERA\_MODEL\_ESP\_EYE)

  pinMode(13, INPUT\_PULLUP);

  pinMode(14, INPUT\_PULLUP);

#endif

  // camera init

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

  if (err != ESP\_OK) {

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

    return;

  }

  sensor\_t \* s = esp\_camera\_sensor\_get();

  // initial sensors are flipped vertically and colors are a bit saturated

  if (s->id.PID == OV3660\_PID) {

    s->set\_vflip(s, 1); // flip it back

    s->set\_brightness(s, 1); // up the brightness just a bit

    s->set\_saturation(s, -2); // lower the saturation

  }

  // drop down frame size for higher initial frame rate

  if(config.pixel\_format == PIXFORMAT\_JPEG){

    s->set\_framesize(s, FRAMESIZE\_QVGA);

  }

#if defined(CAMERA\_MODEL\_M5STACK\_WIDE) || defined(CAMERA\_MODEL\_M5STACK\_ESP32CAM)

  s->set\_vflip(s, 1);

  s->set\_hmirror(s, 1);

#endif

#if defined(CAMERA\_MODEL\_ESP32S3\_EYE)

  s->set\_vflip(s, 1);

#endif

// Setup LED FLash if LED pin is defined in camera\_pins.h

#if defined(LED\_GPIO\_NUM)

  setupLedFlash(LED\_GPIO\_NUM);

#endif

  WiFi.begin(ssid, password);

  WiFi.setSleep(false);

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

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");

  Serial.print(WiFi.localIP());

  Serial.println(" to connect");

  ESP\_Mail\_Session session;

  session server host\_name = SMTP\_server;

  session.server port = SMTP\_Port:

  session login email = sender\_email;

  session login password= sender\_password;

  session login user\_domain= "";

/\* Declare the message class \*/

  SMTP Message message;

  message.sender.name = "ESP 32";

  message.sender.email sender\_email;

  message.subject = "Theft alert";

  message.addRecipient (Recipient\_name, Recipient\_email);

//Send HTML message

  String htmlMsg = "<div style=\"color: #000000;\"><h1>Hello Semicon!</h1><p>Mail Generated from ESP32</p></div>"; message.html.content= htmlMsg.c\_str();

  message.html.content = htmlMsg.c\_str();

  message.text.charset= "us-ascii";

  message.html.transfer\_encoding = Content\_Transfer\_Encoding::enc\_7bit;

  if (!smtp.connect (&session))

   return;

  if (!MailClient.sendMail(&smtp, &message))

   Serial.println("Error sending Email," + smtp.errorReason());

}

void loop() {

  // Do nothing. Everything is done in another task by the web server

  delay(10000);

}