#include "esp\_camera.h"     // Library to interface with the camera

#include <WiFi.h>           // WiFi library for connecting to the network

#include <WebServer.h>      // Library to set up a web server on ESP32

#include "soc/soc.h"        // Used to disable brownout detector

#include "soc/rtc\_cntl\_reg.h" // Used for brownout control

// WiFi/hotspot credentials change according to ur deatils.

const char\* ssid     =  "HMI";

const char\* password = "12345678";

// Web server on port 80

WebServer server(80);

// Camera GPIO configuration for AI Thinker Camera board

#define PWDN\_GPIO\_NUM     32

#define RESET\_GPIO\_NUM    -1

#define XCLK\_GPIO\_NUM      0

#define SIOD\_GPIO\_NUM     26

#define SIOC\_GPIO\_NUM     27

#define Y9\_GPIO\_NUM       35

#define Y8\_GPIO\_NUM       34

#define Y7\_GPIO\_NUM       39

#define Y6\_GPIO\_NUM       36

#define Y5\_GPIO\_NUM       21

#define Y4\_GPIO\_NUM       19

#define Y3\_GPIO\_NUM       18

#define Y2\_GPIO\_NUM        5

#define VSYNC\_GPIO\_NUM    25

#define HREF\_GPIO\_NUM     23

#define PCLK\_GPIO\_NUM     22

void setup() {

  // Section 1: Disable brownout detector (prevents resets during voltage drops)

  WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0);

  // Section 2: Initialize Serial for debugging

  Serial.begin(115200);

  Serial.setDebugOutput(true);

  // Section 3: Initialize camera

  if (init\_camera() != ESP\_OK) {

    Serial.println("Camera init failed");

    return;

  }

  // Section 4: Connect to WiFi

  WiFi.begin(ssid, password);

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

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

  Serial.print(WiFi.localIP());

  // Section 5: Set up the web server to handle MJPEG stream

  server.on("/stream", HTTP\_GET, handle\_jpg\_stream);

  server.begin();

}

void loop() {

  // Section 6: Handle client requests for streaming video

  server.handleClient();

}

// Function to handle the MJPEG stream

void handle\_jpg\_stream() {

  camera\_fb\_t \* fb = NULL;

  char buffer[64];

  int64\_t last\_frame = esp\_timer\_get\_time();

  server.setContentLength(CONTENT\_LENGTH\_UNKNOWN);

  server.send(200, "multipart/x-mixed-replace; boundary=frame");

  while (true) {

    // Section 1: Capture image frame

    fb = esp\_camera\_fb\_get();

    if (!fb) {

      Serial.println("Camera capture failed");

      break;

    }

    // Section 2: Send current frame over the stream

    sprintf(buffer, "--frame\r\nContent-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n", fb->len);

    server.sendContent(buffer);  // Header

    server.sendContent((const char\*)fb->buf, fb->len);  // JPEG frame

    server.sendContent("\r\n");  // End of frame

    // Section 3: Return frame buffer to free memory

    esp\_camera\_fb\_return(fb);

    // Section 4: Calculate and print the time per frame

    int64\_t now = esp\_timer\_get\_time();

    int64\_t frame\_time = now - last\_frame;

    last\_frame = now;

    frame\_time /= 1000;

    Serial.printf("MJPG: %u ms (%.1f fps)\n", (uint32\_t)frame\_time, 1000.0 / (uint32\_t)frame\_time);

    // Section 5: Break loop if client disconnects

    if (!server.client().connected()) {

      break;

    }

  }

}

// Camera initialization function

esp\_err\_t init\_camera() {

  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.pixel\_format = PIXFORMAT\_JPEG;

  config.frame\_size = FRAMESIZE\_VGA;

  config.jpeg\_quality = 12;

  config.fb\_count = 2;

  // Initialize the camera

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

  if (err != ESP\_OK) {

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

    return err;

  }

  return ESP\_OK;

}