```
#include <WiFi.h>

#include <WebServer.h>

#include <Wire.h>

#include <Adafruit_MLX90640.h>


// ------------ MLX90640 ------------

Adafruit_MLX90640 mlx;

float frame[32 * 24];  // 768 pixels

float alertTemp = 50.0;  // °C threshold for alarm


// ------------ DFPlayer on Hardware Serial2 ------------

HardwareSerial MP3Serial(2);   // UART2 on ESP32

const int MP3_RX = 16;        // ESP32 RX2  (to DFPlayer TX)

const int MP3_TX = 17;        // ESP32 TX2  (to DFPlayer RX through 1k resistor)


// Send DFPlayer command (no library)

void dfSendCommand(uint8_t cmd, uint16_t param = 0) {

 uint8_t buf[10] = {

  0x7E, 0xFF, 0x06, cmd, 0x00,

  (uint8_t)(param >> 8), (uint8_t)(param & 0xFF),

  0x00, 0x00, 0xEF

 };


 // simple checksum (high & low)

 uint16_t sum = 0;

 for (int i = 1; i < 7; i++) sum += buf[i];

 sum = 0 - sum;

 buf[7] = (uint8_t)(sum >> 8);
```

```
  buf[8] = (uint8_t)(sum & 0xFF);

  for (int i = 0; i < 10; i++) {
    MP3Serial.write(buf[i]);
  }
}

void dfInit() {
  MP3Serial.begin(9600, SERIAL_8N1, MP3_RX, MP3_TX);
  delay(500);
  dfSendCommand(0x3F);      // reset
  delay(500);
  dfSendCommand(0x06, 25);  // set volume (0–30)
}

// Play the first / only track on SD card
void dfPlayAlert() {
  dfSendCommand(0x03, 1);   // play track 1
}

// ------------ WiFi + WebServer ------------
WebServer server(80);
const char *ssid     = "EV_Thermal_Device";
const char *password = "12345678";

unsigned long lastAlertMillis = 0;
const unsigned long alertCooldown = 8000; // 8 seconds between alerts
```

```
// ------------ Web page (HTML + JS) ------------
const char MAIN_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <title>EV Thermal Monitor</title>
 <style>
  body {
   background: #111;
   color: #eee;
   font-family: Arial, sans-serif;
   text-align: center;
  }
  h1 { margin-top: 10px; }
  #info { margin: 10px; font-size: 14px; }
  .grid {
   display: grid;
   grid-template-columns: repeat(32, 10px);
   grid-gap: 2px;
   margin: 0 auto;
   margin-top: 10px;
  }
  .cell {
   width: 10px;
   height: 10px;
   background: #000;
  }
```

```
    #warning {
      margin-top: 12px;
      font-size: 18px;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <h1>EV Thermal Camera (MLX90640)</h1>
  <div id="info">Connecting...</div>
  <div id="grid" class="grid"></div>
  <div id="warning"></div>

  <script>
    const grid = document.getElementById("grid");
    const info = document.getElementById("info");
    const warning = document.getElementById("warning");

    // Create 32x24 cells
    const cells = [];
    for (let i = 0; i < 32*24; i++) {
      const d = document.createElement("div");
      d.className = "cell";
      grid.appendChild(d);
      cells.push(d);
    }

    function tempToColor(t, tmin, tmax) {
```

```
  if (tmax <= tmin) tmax = tmin + 0.01;
  let norm = (t - tmin) / (tmax - tmin);
  if (norm < 0) norm = 0;
  if (norm > 1) norm = 1;
  // 240 = blue, 0 = red (HSL)
  let hue = (1 - norm) * 240;
  return "hsl(" + hue + ", 100%, 50%)";
}

function updateFrame() {
  fetch("/data")
  .then(r => r.json())
  .then(obj => {
    const arr = obj.temps;
    const maxT = obj.max;
    const minT = obj.min;
    info.innerText = "Min: " + minT.toFixed(1) +
                " °C | Max: " + maxT.toFixed(1) + " °C";

    for (let i = 0; i < cells.length && i < arr.length; i++) {
      const t = arr[i];
      cells[i].style.backgroundColor = tempToColor(t, minT, maxT);
    }

    if (maxT >= 50.0) {
      warning.style.color = "red";
      warning.innerText = "⚠ WARNING: High temperature detected!";
    } else {
        warning.innerText = "";
      }
    })
    .catch(e => {
      info.innerText = "Error reading data";
    });
  }

  setInterval(updateFrame, 500);
  updateFrame();
 </script>
</body>
</html>
)=====";

// ------------ HTTP Handlers ------------
void handleRoot() {
 server.send_P(200, "text/html", MAIN_page);
}

void handleData() {
 // Read one frame
 int status = mlx.getFrame(frame);
 if (status != 0) {
  Serial.print("MLX90640 getFrame error: ");
  Serial.println(status);
  server.send(500, "text/plain", "MLX90640 error");
  return;
```

```
}

float maxT = -999;
float minT =  999;
for (int i = 0; i < 768; i++) {
  if (frame[i] > maxT) maxT = frame[i];
  if (frame[i] < minT) minT = frame[i];
}

// Sound alarm if too hot (with cooldown)
unsigned long now = millis();
if (maxT >= alertTemp && (now - lastAlertMillis > alertCooldown)) {
  Serial.println("ALERT: High temperature, playing sound!");
  dfPlayAlert();
  lastAlertMillis = now;
}

// Build JSON: { "temps":[...], "max":X, "min":Y }
String json = "{\"temps\":[";
for (int i = 0; i < 768; i++) {
  json += String(frame[i], 2);
  if (i < 767) json += ",";
}
json += "],\"max\":";
json += String(maxT, 2);
json += ",\"min\":";
json += String(minT, 2);
json += "}";
```

```
  server.send(200, "application/json", json);
}

// ------------ Setup & Loop ------------
void setup() {
  Serial.begin(115200);
  delay(200);

  // I2C for MLX90640
  Wire.begin(21, 22); // SDA, SCL
  Wire.setClock(400000); // 400kHz I2C for speed

  Serial.println("Starting MLX90640...");
  if (!mlx.begin(MLX90640_I2CADDR_DEFAULT, &Wire)) {
    Serial.println("MLX90640 not found, check wiring!");
    while (1) delay(1000);
  }

  mlx.setMode(MLX90640_CHESS);
  mlx.setResolution(MLX90640_ADC_18BIT);
  mlx.setRefreshRate(MLX90640_8_HZ);

  Serial.println("MLX90640 initialized.");

  // DFPlayer init
  dfInit();
  Serial.println("DFPlayer initialized.");
  // WiFi AP mode
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("WiFi AP started. Connect to: ");
  Serial.println(ssid);
  Serial.print("Open in browser: http://");
  Serial.println(IP);

  // WebServer routes
  server.on("/", handleRoot);
  server.on("/data", handleData);
  server.begin();
  Serial.println("Web server started.");
}

void loop() {
  server.handleClient();
}
```