ESP32 CAM SD

esp32camSD.ino

```
/*********************************************************************

L'ESP32-CAM présente une page web qui permet de prendre des photos
et de les enregistrer sur une carte SD, de visionner les photos
déjà présentes sur la carte, et de supprimer les photos non-désirées.

Pour plus d'informations:

http://electroniqueamateur.blogspot.com/2020/07/esp32-cam-gestion-distance-de-la-carte.html

*********************************************************************/

#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include "FS.h"       // manipulation de fichiers
#include "SD_MMC.h"   // carte SD
#include "esp_camera.h" // caméra!

// écrivez le nom et le mot de passe de votre réseau WIFI
const char* ssid = "**********";
const char* password = "**********";

WebServer server(80);

static bool cartePresente = false;

int numero_fichier = 0; // numéro de la photo (nom du fichier)


// prise de la photo et création du fichier jpeg

void enregistrer_photo (void)
{
  char adresse[20] = ""; // chemin d'accès du fichier .jpeg
  camera_fb_t * fb = NULL; // frame buffer

  // prise de la photo
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Echec de la prise de photo.");
    return;
  }
```

```
  numero_fichier = numero_fichier + 1;

  // enregitrement du fichier sur la carte SD

  sprintf (adresse, "/%d.jpg", numero_fichier);

  fs::FS &fs = SD_MMC;
  File file = fs.open(adresse, FILE_WRITE);

  if (!file) {
    Serial.println("Echec lors de la creation du fichier.");
  }
  else {
    file.write(fb->buf, fb->len); // payload (image), payload length
    Serial.printf("Fichier enregistre: %s\n", adresse);
  }
  file.close();
  esp_camera_fb_return(fb);

   // on affiche un message de confirmation
  server.setContentLength(CONTENT_LENGTH_UNKNOWN);

  server.send(200,  "text/html", "");

  WiFiClient client = server.client();

  server.sendContent("<h1>Une nouvelle photo a &eacute;t&eacute;
prise.</h1>");
  server.sendContent("<p><a href = / > Retour &agrave; la liste des
fichiers </a></p>");

}

void returnOK() {
  server.send(200, "text/plain", "");
}

void returnFail(String msg) {
  server.send(500, "text/plain", msg + "\r\n");
}

// Affichage d'un fichier présent sur la carte
bool loadFromSdCard(String path) {
  String dataType = "text/plain";

  if (path == "/") {
    printDirectory();
  }
  else {
```

```
    if (path.endsWith(".src")) {
      path = path.substring(0, path.lastIndexOf("."));
    } else if (path.endsWith(".htm")) {
      dataType = "text/html";
    } else if (path.endsWith(".css")) {
      dataType = "text/css";
    } else if (path.endsWith(".js")) {
      dataType = "application/javascript";
    } else if (path.endsWith(".png")) {
      dataType = "image/png";
    } else if (path.endsWith(".gif")) {
      dataType = "image/gif";
    } else if (path.endsWith(".jpg")) {
      dataType = "image/jpeg";
    } else if (path.endsWith(".ico")) {
      dataType = "image/x-icon";
    } else if (path.endsWith(".xml")) {
      dataType = "text/xml";
    } else if (path.endsWith(".pdf")) {
      dataType = "application/pdf";
    } else if (path.endsWith(".zip")) {
      dataType = "application/zip";
    }

    fs::FS &fs = SD_MMC;

    File dataFile = fs.open(path.c_str());

    if (!dataFile) {
      return false;
    }

    if (server.hasArg("download")) {
      dataType = "application/octet-stream";
    }

    if (server.streamFile(dataFile, dataType) != dataFile.size()) {
      Serial.println("Sent less data than expected!");
    }

    dataFile.close();

  }
  return true;
}


// utilisé lors de la suppression d'un fichier
void deleteRecursive(String path) {
```

```
  fs::FS &fs = SD_MMC;
  File file = fs.open((char *)path.c_str());
  if (!file.isDirectory()) {
    file.close();
    fs.remove((char *)path.c_str());
    return;
  }

  file.rewindDirectory();
  while (true) {
    File entry = file.openNextFile();
    if (!entry) {
      break;
    }
    String entryPath = path + "/" + entry.name();
    if (entry.isDirectory()) {
      entry.close();
      deleteRecursive(entryPath);
    } else {
      entry.close();
      fs.remove((char *)entryPath.c_str());
    }
    yield();
  }

  fs.rmdir((char *)path.c_str());
  file.close();
}

// suppression d'un fichier

void handleDelete() {

  fs::FS &fs = SD_MMC;

  if (server.args() == 0) {
    return returnFail("Mauvais arguments?");
  }
  String path = server.arg(0);
  if (path == "/" || !fs.exists((char *)path.c_str())) {
    returnFail("BAD PATH");
    return;
  }
  deleteRecursive(path);

  // on affiche un message de confirmation
  server.setContentLength(CONTENT_LENGTH_UNKNOWN);

  server.send(200,  "text/html", "");
```

```
  WiFiClient client = server.client();

  server.sendContent("<h1>Le fichier a &eacute;t&eacute;
supprim&eacute;</h1>");
  server.sendContent("<p><a href = / > Retour &agrave; la liste des
fichiers </a></p>");

}

// Affichage du contenu de la carte
void printDirectory() {

  fs::FS &fs = SD_MMC;
  String path = "/";

  File dir = fs.open((char *)path.c_str());
  path = String();
  if (!dir.isDirectory()) {
    dir.close();
    return returnFail("PAS UN REPERTOIRE");
  }
  dir.rewindDirectory();
  server.setContentLength(CONTENT_LENGTH_UNKNOWN);

  server.send(200,  "text/html", "");

  WiFiClient client = server.client();

  server.sendContent("<h1>Prise de photo</h1>");

  server.sendContent("<br> <form action='/clic' method='GET'> <INPUT
type='submit' value='Prendre une photo'></form><br> ");

  server.sendContent("<h1>Contenu de la carte SD</h1>");

  for (int cnt = 0; true; ++cnt) {
    File entry = dir.openNextFile();
    if (!entry) {
      break;
    }

    String output;

    output += "<a href = ";
    output += entry.name();
    output += "> ";
    output += entry.name();

    // on ajoute un bouton delete:
    output += "</a>          <a href =
```

```
/delete?url=";
    output += entry.name();
    output += "> [Supprimer] </a> <br>";

    server.sendContent(output);
    entry.close();
  }
  dir.close();
}

// on tente d'afficher le fichier demandé. Sinon, message d'erreur
void handleNotFound() {
  if (cartePresente && loadFromSdCard(server.uri())) {
    return;
  }
  String message = "Carte SD non detectee ou action imprevue\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " NAME:" + server.argName(i) + "\n VALUE:" +
server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
  Serial.print(message);
}

void setup(void) {

  // définition des broches de la caméra pour le modèle AI Thinker -
ESP32-CAM
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = 5;
  config.pin_d1 = 18;
  config.pin_d2 = 19;
  config.pin_d3 = 21;
  config.pin_d4 = 36;
  config.pin_d5 = 39;
  config.pin_d6 = 34;
  config.pin_d7 = 35;
  config.pin_xclk = 0;
  config.pin_pclk = 22;
  config.pin_vsync = 25;
```

```
  config.pin_href = 23;
  config.pin_sscb_sda = 26;
  config.pin_sscb_scl = 27;
  config.pin_pwdn = 32;
  config.pin_reset = -1;

  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;  //YUV422|GRAYSCALE|RGB565|JPEG
  config.frame_size = FRAMESIZE_VGA;  //
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
  config.jpeg_quality = 10;  // 0-63 ; plus bas = meilleure qualité
  config.fb_count = 2; // nombre de frame buffers

  // initialisation de la caméra
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Echec de l'initialisation de la camera, erreur
0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();

  // connexion au WIFI

  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connexion au reseau Wifi ");
  Serial.println(ssid);

  uint8_t i = 0;
  while (WiFi.status() != WL_CONNECTED && i++ < 20) {//wait 10 seconds
    delay(500);
  }
  if (i == 21) {
    Serial.print("Impossible de se connecter au reseau ");
    Serial.println(ssid);
    while (1) {
      delay(500);
    }
  }
  Serial.print("Connecte a l'adresse IP: ");
  Serial.println(WiFi.localIP());

  // initialisation du web server
  server.on("/delete", HTTP_GET, handleDelete);
  server.on("/clic", HTTP_GET, enregistrer_photo);
  server.onNotFound(handleNotFound);
  server.begin();
  Serial.println("Serveur HTTP en fonction.");
```

```
  // initialisation de la carte micro SD
  if (SD_MMC.begin()) {
    uint8_t cardType = SD_MMC.cardType();
    if (cardType != CARD_NONE) {
      Serial.println("Carte SD Initialisee.");
      cartePresente = true;
    }
  }
}

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