Take picture with ESP32 cam and store in SD card

The procedure is mainly adapted from [ESP32 cam guide](https://www.diyengineers.com/2023/04/13/esp32-cam-complete-guide/) and [Formatting SD guide](https://www.youtube.com/watch?v=IceA1OxEFa4)\

A close-up of a computer chip

Description automatically generated

ESP32 cam with OV2640 Camera

The ESP32 CAM doesn’t have micro-USB port. It can be connected with ESP32-CAM-MB and use its micro-USB port. The micro-USB port is used for uploading the compiled program. In case when ESP32-CAM-MB isn’t available,FTDI jumper can be used for converting micro-USB to UART serial port.

To compile ESP32 on Arduino, add the following link to Arduino IDE under Preferences->Additional Boards Manager->Additional Boards Manager URLs

<https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json>

Then go to Tools->Board->Board Manager. Install ESP32 by Expressif Systems.

Once Installed, go to Tools->Board->ESP32 Arduino->AI Thinker ESP32-CAM.

Connect ESP32 Cam and upload compiled code.

ESP32 CAM has a SD card socket to store files. However, it only supports SD card with FAT32 format. However, if a SD card is larger than 32GB, FAT32 isn’t supported by default. To format those SD card to FAT32, download [FAT32 Format For Windows](http://ridgecrop.co.uk/index.htm?guiformat.htm) and [SD Memory Card Formatter](https://www.sdcard.org/downloads/formatter/).

Open SD Card Formatter, select quick format, and format. Open FAT32 Formatter, select quick format, and start. Remember to select the correct drive. If there is an error when formatting, try select smaller Allocation unit size and format again

#include "esp\_camera.h"

#include "Arduino.h"

#include "FS.h"                // SD Card ESP32

#include "SD\_MMC.h"            // SD Card ESP32

#include "soc/soc.h"           // Disable brownour problems

#include "soc/rtc\_cntl\_reg.h"  // Disable brownour problems

#include "driver/rtc\_io.h"

#include <EEPROM.h>            // read and write from flash memory

// define the number of bytes you want to access

#define EEPROM\_SIZE 1

RTC\_DATA\_ATTR int bootCount = 0;

// Pin definition for CAMERA\_MODEL\_AI\_THINKER

#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

int pictureNumber = 0;

void setup() {

  WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0); //disable brownout detector

  Serial.begin(115200);

  Serial.setDebugOutput(true);

  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;

  pinMode(4, INPUT);

  digitalWrite(4, LOW);

  rtc\_gpio\_hold\_dis(GPIO\_NUM\_4);

  if(psramFound()){

    config.frame\_size = FRAMESIZE\_UXGA; // FRAMESIZE\_ + QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA

    config.jpeg\_quality = 10;

    config.fb\_count = 2;

  } else {

    config.frame\_size = FRAMESIZE\_SVGA;

    config.jpeg\_quality = 12;

    config.fb\_count = 1;

  }

  // Init Camera

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

  if (err != ESP\_OK) {

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

    return;

  }

  Serial.println("Starting SD Card");

  delay(500);

  if(!SD\_MMC.begin()){

    Serial.println("SD Card Mount Failed");

    //return;

  }

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

  if(cardType == CARD\_NONE){

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

    return;

  }

  camera\_fb\_t \* fb = NULL;

  // Take Picture with Camera

  fb = esp\_camera\_fb\_get();

  delay(1000);//This is key to avoid an issue with the image being very dark and green. If needed adjust total delay time.

  fb = esp\_camera\_fb\_get();

  if(!fb) {

    Serial.println("Camera capture failed");

    return;

  }

  // initialize EEPROM with predefined size

  EEPROM.begin(EEPROM\_SIZE);

  pictureNumber = EEPROM.read(0) + 1;

  // Path where new picture will be saved in SD Card

  String path = "/picture" + String(pictureNumber) +".jpg";

  fs::FS &fs = SD\_MMC;

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

  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); // payload (image), payload length

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

    EEPROM.write(0, pictureNumber);

    EEPROM.commit();

  }

  file.close();

  esp\_camera\_fb\_return(fb);

  delay(1000);

  // Turns off the ESP32-CAM white on-board LED (flash) connected to GPIO 4

  pinMode(4, OUTPUT);

  digitalWrite(4, LOW);

  rtc\_gpio\_hold\_en(GPIO\_NUM\_4);

  esp\_sleep\_enable\_ext0\_wakeup(GPIO\_NUM\_13, 1);

  delay(500);

  esp\_deep\_sleep\_start();

  Serial.println("This will never be printed");

}

void loop() {

}