Image Transfer from ESP32 Cam to Console

ESP32 cam needs to be configured and initialized to capture images (see example). After properly initialized, function esp\_camera\_fb\_get() will return a struct of frame buffer:

typedef struct {

    uint8\_t \* buf;              /\*!< Pointer to the pixel data \*/

    size\_t len;                 /\*!< Length of the buffer in bytes \*/

    size\_t width;               /\*!< Width of the buffer in pixels \*/

    size\_t height;              /\*!< Height of the buffer in pixels \*/

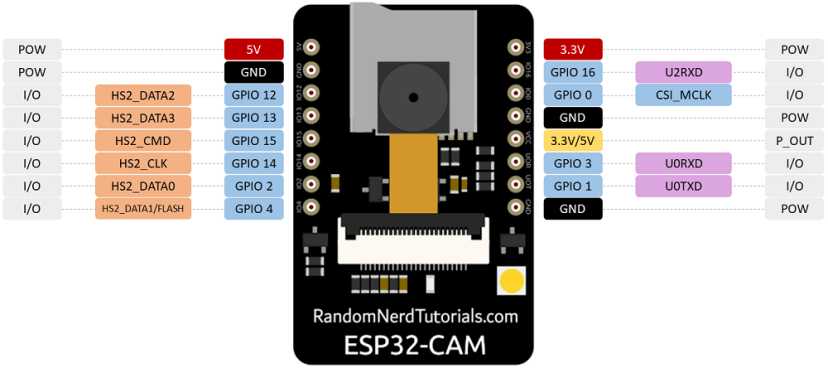
    pixformat\_t format;         /\*!< Format of the pixel data \*/

    struct timeval timestamp;   /\*!< Timestamp since boot of the first DMA buffer of the frame \*/

} camera\_fb\_t;

The frame buffer is at the buf pointer with the size of len. If it is properly configured, the buffer will be in the jpeg format. And these data can be transmitted as byte stream through serial port with Serial.write(fb->buf, fb->len);

ESP32 cam only has 3 pins dedicated for UART, as the figure below:



GPIO 1 and 3 are also used for sketch uploading. And because of the board configuration of ESP32 cam, this serial port isn’t compatible with console emulator like Teraterm. It can be properly interpreted by Arduino serial monitor if the board configuration is loaded. The problem according to [this post](https://forum.arduino.cc/t/esp32cam-ai-thinker-clone-strange-behaviour/1303127/3?u=e1berto) seems to be that ESP32 cam disabled DTR and RTS, and if these are used, the board will keep rebooting? However, the exact solution to configure Teraterm to fit with this setting is still unknown. This will also mess up the transmission if directly connected to HC-06 bluetooth module for serial transmission. Direct connection to other microcontrollers like MSP432P401R through UART seems to be fine though.

For now, in order to receive serial data on PC, [pyserial library](https://pyserial.readthedocs.io/en/latest/pyserial_api.html) is used. To resolve the issue of asserting DTR and RTS, it needs to initiate a port object and set dsrdtr to False before establishing the connection. For example:

    com = serial.Serial()

    com.port = port\_name

    com.baudrate = baud

    com.timeout = 1

    com.dsrdtr = False

    com.open()

With connection properly established, it can receive bytes from serial port and save it as jpg file

Code on ESP32 cam

#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

  pinMode(4, OUTPUT);

  digitalWrite(4, HIGH);

  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();

  digitalWrite(4, LOW);

  if(!fb) {

    Serial.println("Camera capture failed");

    return;

  }

  // 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());

  //   Serial.printf("%u\n", \*fb->buf);

  //   EEPROM.write(0, pictureNumber);

  //   EEPROM.commit();

  // }

  // file.close();

  Serial.printf("transmission started\n");

  Serial.printf("%u\n",fb->len);

  Serial.write(fb->buf, fb->len);

  Serial.printf("\ntransmission ends\n");

  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);

  delay(500);

  esp\_deep\_sleep\_start();

}

void loop() {

}

Code of python script

import serial, time

import sys

if \_\_name\_\_ == '\_\_main\_\_':

    if len(sys.argv) != 3:

        print("python script serial\_port baudrate")

        print("python -m serial.tools.list\_ports")

        exit()

    port\_name = sys.argv[1]

    baud = sys.argv[2]

    com = serial.Serial()

    com.port = port\_name

    com.baudrate = baud

    com.timeout = 1

    com.dsrdtr = False

    com.open()

    #ser = serial.Serial(port\_name, baudrate= baud, timeout = 2, dsrdtr=None)

    print("serial connected")

    while True:

        value = com.readline()

        line = str(value, encoding="UTF-8")

        print(line)

        if(value == b'transmission started\n'):

            value = com.readline()

            line = str(value, encoding="UTF-8")

            file\_len = int(line)

            print(f'file size: {file\_len} bytes')

            start = time.time()

            file\_buf = com.read(file\_len)

            with open("image.jpg", "wb") as f:

                f.write(file\_buf)

            end = time.time()

            print(end - start)

            break