Bluetooth Transfer from ESP32 Cam

The ESP32 cam has embedded Bluetooth module that can be initiated using BluetoothSerial.h.

With function SerialBT.begin(device\_name), the device will appear with the provide name as the Bluetooth device. It can be connected to PC with “find Bluetooth Device” option. After connected, two serial ports will be added for connection.

Bluetooth serial has the similar functionality with regular serial like read, write, print, avail, etc. It can also register call back functions using SerialBT.register\_callback(btCallback); This can be used to respond to different events, like connection, data input, etc.

The Bluetooth transmission is simply writing the whole frame buffer through the serial port. It seems like this process can’t be done under the call back function, or else the data will be lost through the Bluetooth transfer. Another potential reason is that the wireless transfer is inherently unreliable. So it might be required some kind of transfer protocol to detect the missing segments.

Currently, the transfer is fine with simply writing the whole frame buffer through serial port. However, the image capture will fail after the first capture.

On the PC, it can start the connection like other serial port using pyserial.

#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

#include "BluetoothSerial.h"   //bluetooth

// 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

//Example SerialtoSerialBT

// Check if Bluetooth is available

#if !defined(CONFIG\_BT\_ENABLED) || !defined(CONFIG\_BLUEDROID\_ENABLED)

#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it

#endif

// Check Serial Port Profile

#if !defined(CONFIG\_BT\_SPP\_ENABLED)

#error Serial Port Profile for Bluetooth is not available or not enabled. It is only available for the ESP32 chip.

#endif

#define CHUNK\_SIZE (size\_t)32

BluetoothSerial SerialBT;

String device\_name = "ESP32\_CAM\_BT";

void setup() {

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

  Serial.begin(115200);

  cam\_init();

  bt\_init();

}

void loop() {

  char read;

  if(SerialBT.available())

  {

    read = SerialBT.read();

    Serial.write(read);

    if(read == 't')

      take\_picture();

    else{

      Serial.print("picture not taken");

      SerialBT.print("picture not taken");

    }

  }

}

void cam\_init()

{

  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;

  if(psramFound()){

    Serial.print("Using UXGA\n");

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

    config.jpeg\_quality = 10;

    config.fb\_count = 2;

  } else {

    Serial.print("Using SVGA\n");

    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;

  }

}

void bt\_init()

{

  SerialBT.begin(device\_name);

  SerialBT.register\_callback(btCallback);

  Serial.println("The device started, now you can pair it with bluetooth");

}

void btCallback(esp\_spp\_cb\_event\_t event, esp\_spp\_cb\_param\_t \*param){

  //Serial Port Profile (SPP) API

  if(event == ESP\_SPP\_SRV\_OPEN\_EVT){

     /\*!< When SPP is initialized, the event comes \*/

    Serial.println("Client Connected!");

  }else if(event == ESP\_SPP\_DATA\_IND\_EVT){

    /\*!< When SPP connection received data, the event comes, only for ESP\_SPP\_MODE\_CB \*/

    // Serial.printf("ESP\_SPP\_DATA\_IND\_EVT len=%d, handle=%d", param->data\_ind.len, param->data\_ind.handle);

    // Serial.printf("value is %.\*s\n", param->data\_ind.len, param->data\_ind.data);

    // char stringRead[50];

    // if(param->data\_ind.len < 50)

    // {

    //   memcpy(stringRead,  param->data\_ind.data, param->data\_ind.len);

    //   stringRead[param->data\_ind.len] = 0;

    //   Serial.printf("value is %.\*s\n", param->data\_ind.len+1, stringRead);

    //   if(stringRead[0] == 't')

    //   {

    //     Serial.printf("take picture\n");

    //     take\_picture();

    //   }

    // }

    // else

    // {

    //   Serial.printf("picture not taken\n");

    // }

  }

}

void take\_picture()

{

  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) {

    esp\_camera\_fb\_return(fb);

    Serial.println("Camera capture failed");

    SerialBT.println("Camera capture failed");

    return;

  }

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

  Serial.printf("%u\n",fb->len);

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

  SerialBT.printf("%u\n",fb->len);

  SerialBT.flush();

  delay(10);

  // size\_t len = 0;

  // while(len < fb->len){

  //   int packet\_size = min(CHUNK\_SIZE, fb->len - len);

  //   SerialBT.write(fb->buf+len, packet\_size);

  //   Serial.printf("%u\n", len);

  //   len += CHUNK\_SIZE;

  //   SerialBT.flush();

  //   delay(1);

  // }

  SerialBT.write(fb->buf, fb->len);

  SerialBT.flush();

  esp\_camera\_fb\_return(fb);

  digitalWrite(4, LOW);

}