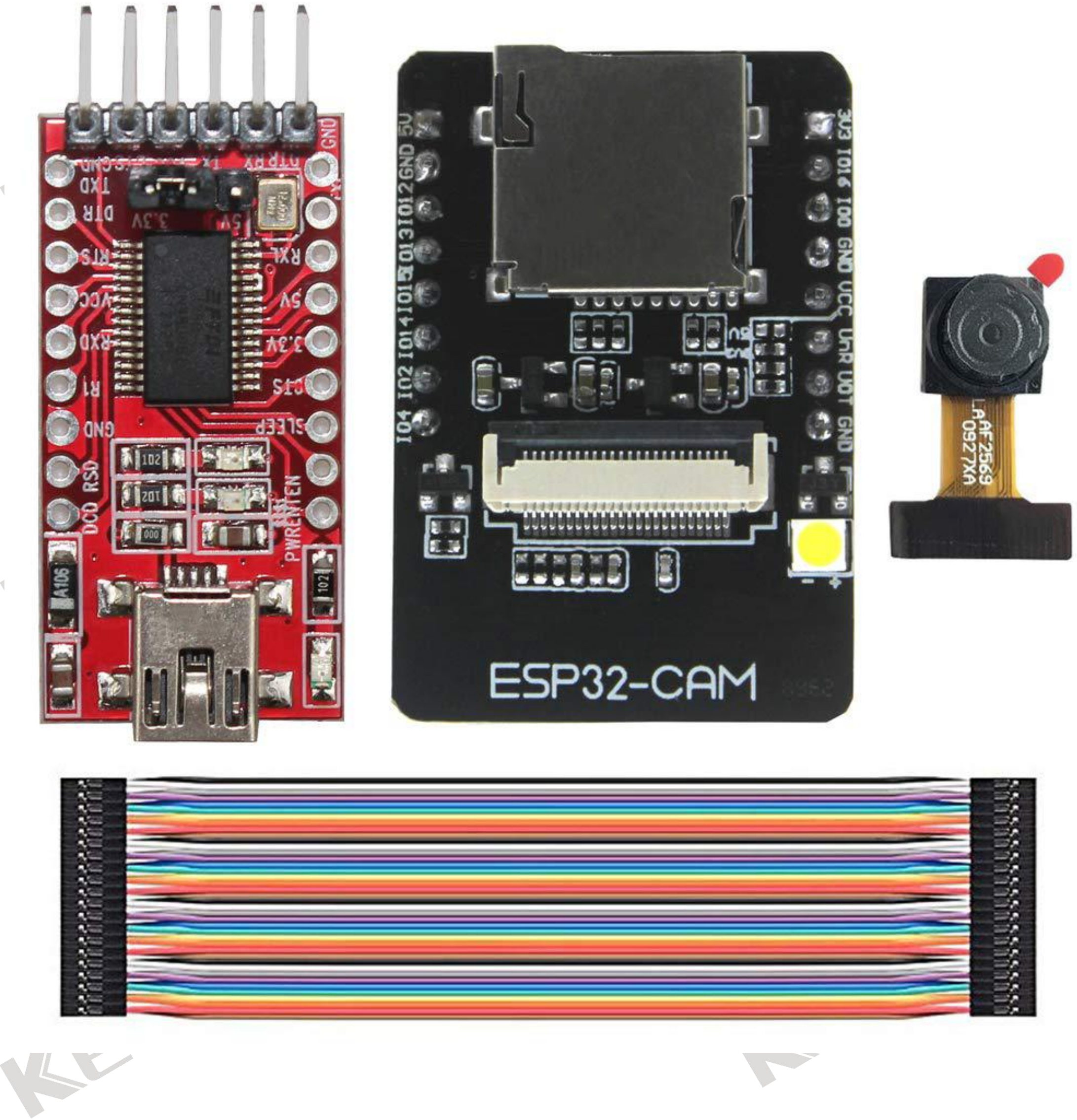
Preface

About Our Company

Located in Shenzhen, the Silicon Valley of China, KeeYees Technology Inc. is a big & professional Electronic Products Manufacturer and Seller, dedicated to open-source hardware research & development, production and marketing. All of our products comply with International Quality Standards and are very popular in a variety of different markets throughout of the world. KeeYees is your best choice in various electronic modules & components designed for customers of any level to learn Arduino and Raspberry Pi knowledge. In addition, we also sell products like 3D printer accessories, connectors and terminals kits, DIY parts and tools to support your work and design challenges from Home, School to Industrial applications!

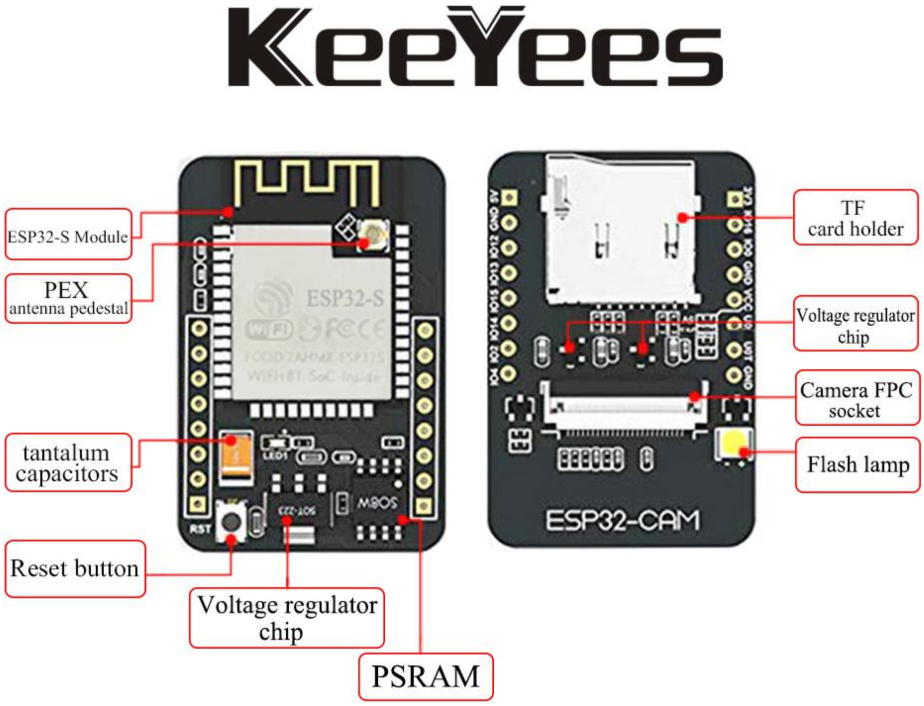
US Amazon Store Homepage:
https://www.amazon.com/shops/A2K4DGCC72N9AG
UK Amazon Store Homepage:
https://www.amazon.co.uk/shops/A1F4U6XVWUBG1U
DE Amazon Store Homepage:
https://www.amazon.de/shops/A1F4U6XVWUBG1U
FR Amazon Store Homepage:
https://www.amazon.fr/shops/A1F4U6XVWUBG1U
IT Amazon Store Homepage:
https://www.amazon.it/shops/A1F4U6XVWUBG1U
ES Amazon Store Homepage:
https://www.amazon.es/shops/A1F4U6XVWUBG1U
JP Amazon Store Homepage:
https://www.amazon.co.jp/shops/A7NY3JX21TGU2

KeeYees ESP32-Cam WiFi and Bluetooth Camera Module



Introducing the ESP32-CAM

The ESP32-CAM is a very small camera module with the ESP32-S chip. Besides the OV2640 camera, and several GPIOs to connect peripherals, it also features a microSD card slot that can be useful to store images taken with the camera or to store files.



Features

- The smallest 802.11b/g/n Wi-Fi BT SoC module
- Low power 32-bit CPU, can also serve the application processor
- Up to 160MHz clock speed, summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, built-in flash lamp
- Support image WiFI upload
- Support TF card
- Supports multiple sleep modes
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

ESP32-CAM Video Streaming and Face Recognition with Arduino IDE

The following guide will show you how to setup a video streaming web server with face recognition and detection with Arduino IDE.

Note: In this tutorial we use the example from the arduino-esp32 library. This tutorial doesn't cover how to modify the example.

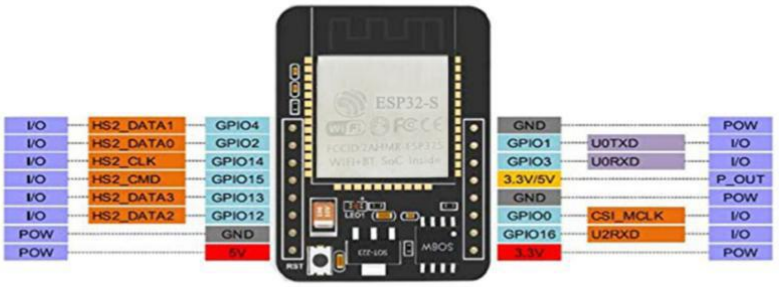
Parts Required

- ESP32-CAM with OV2640
- FTDI programmer
- Female-to-female jumper wires

The ESP32-CAM doesn't come with a USB connector, so you need an FTDI programmer to upload code through the U0R and U0T pins (serial pins).

ESP32-CAM Pinout

The following figure shows the ESP32-CAM pinout (AI-Thinker module).



There are three GND pins and two pins for power: either 3.3V or 5V.

GPIO 1 and GPIO 3 are the serial pins. You need these pins to upload code to your board. Additionally, GPIO 0 also plays an important role, since it determines whether the ESP32 is in flashing mode or not. When GPIO 0 is connected to GND, the ESP32 is in flashing mode.

The following pins are internally connected to the microSD card reader:

GPIO 14: CLK

GPIO 15: CMD

GPIO 2: Data 0

GPIO 4: Data 1 (also connected to the on-board LED)

GPIO 12: Data 2

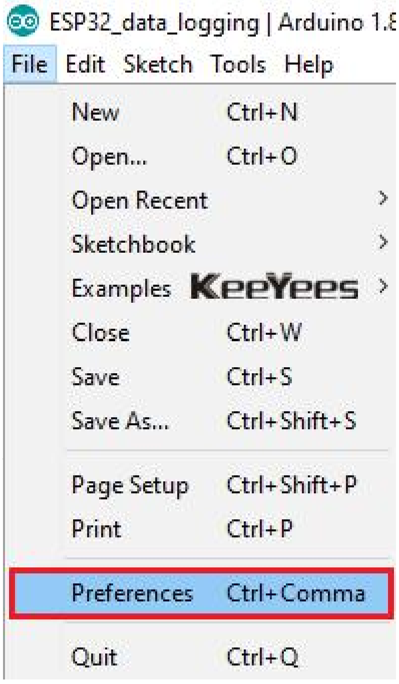
GPIO 13: Data 3

Video Streaming Server

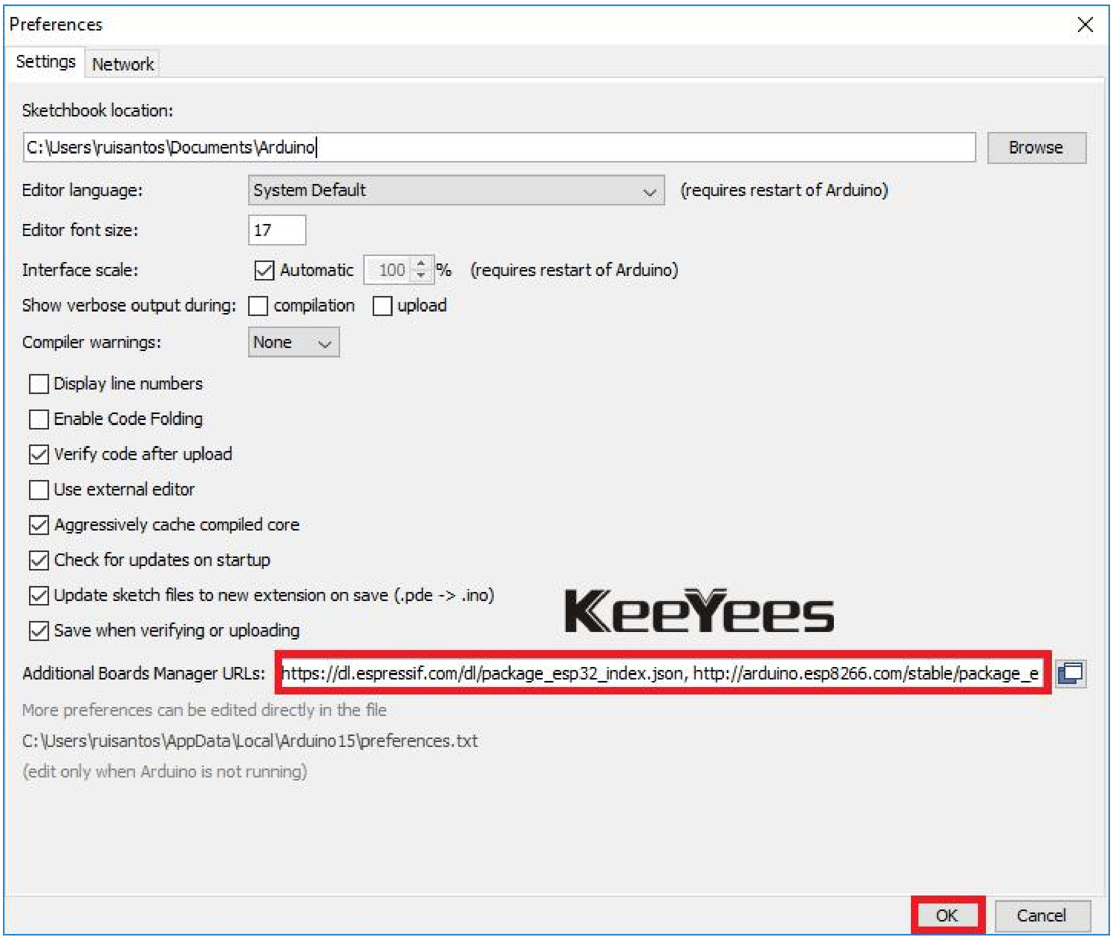
Install the ESP32 add-on

In this example, we use Arduino IDE to program the ESP32-CAM board. So, you need to have Arduino IDE installed as well as the ESP32 add-on.

Open Arduino IDE and click file->Preferences as shown below.



Then enter https://dl.espressif.com/dl/package_esp32_index.json in the additilnal boards manaper URLS field, then click on "ok" as shown below.



Preferences

Settings | Network

Sketchbook location:
C:\Users\ruisantos\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 17

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

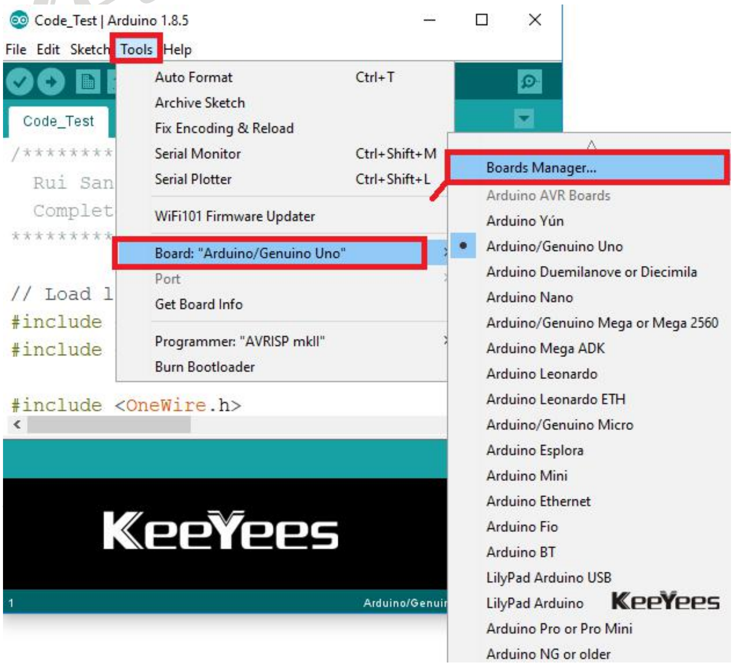
Compiler warnings: None

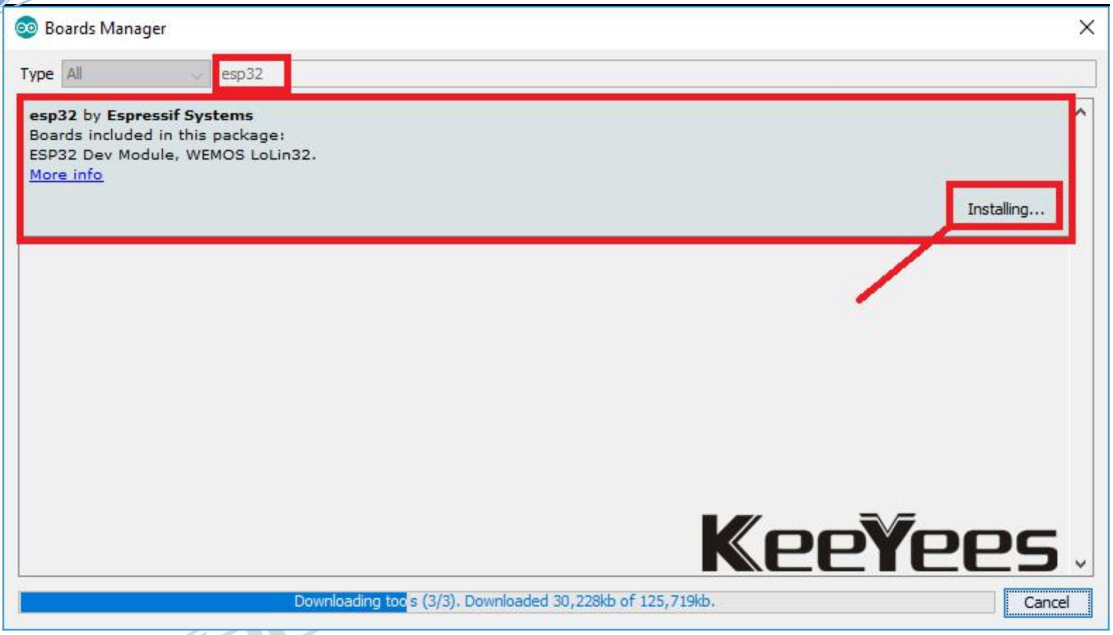
□ Display line numbers
□ Enable Code Folding
☑ Verify code after upload
□ Use external editor
☑ Aggressively cache compiled core
☑ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Save when verifying or uploading

Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_e

More preferences can be edited directly in the file
C:\Users\ruisantos\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Click tools->board:->Blards Manager, then enter ESP32 in the pop-up interface, click Install. As shown below





esp32 by Espressif Systems
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

Downloading tools (3/3). Downloaded 30,228kb of 125,719kb.

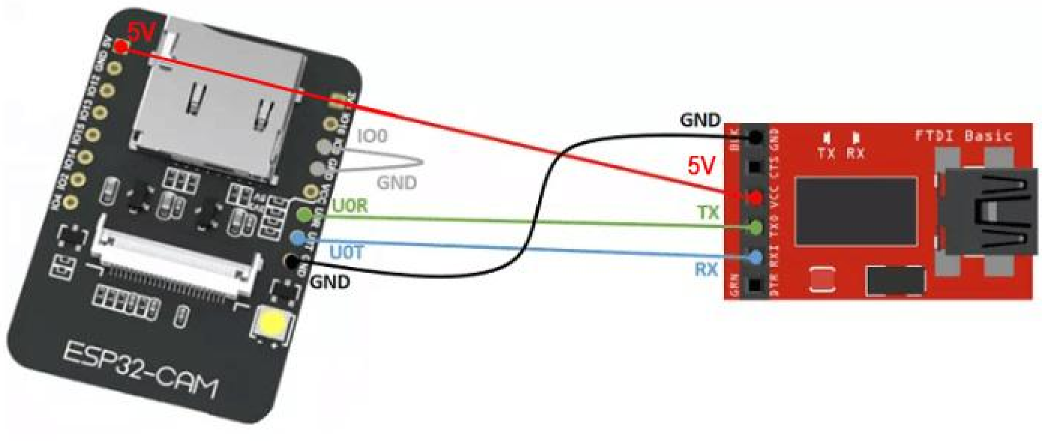
After finishing the above steps, you can use the module now.

Steps

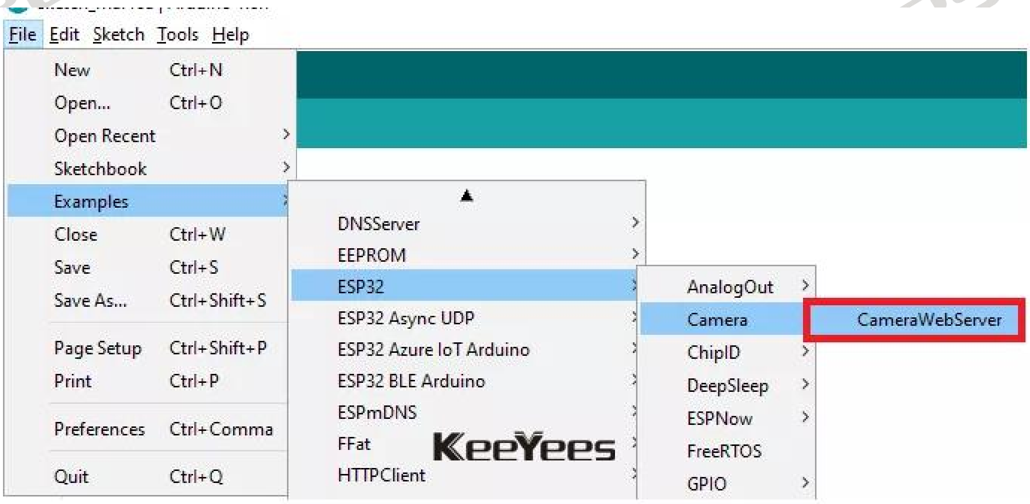
1. Prepare a memory card.

2. Insert the memory card and the OV2640 camera module into the card holder.

Connect the ESP32-CAM board to your computer using an FTDI programmer. Follow the next schematic diagram:

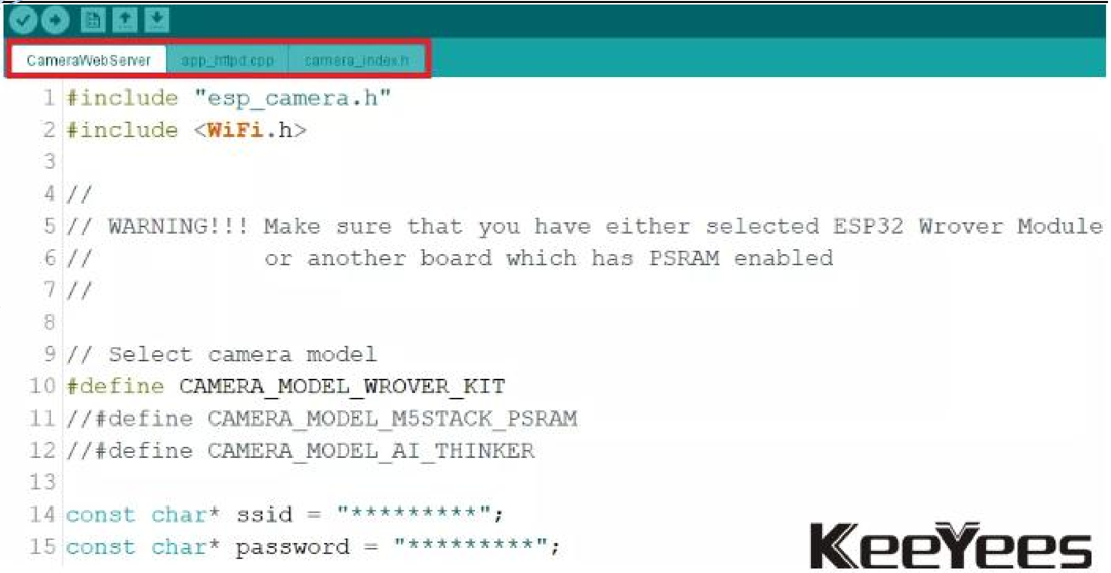


2. In your Arduino IDE, go to File > Examples > ESP32 > Camera and open the CameraWebServer example.



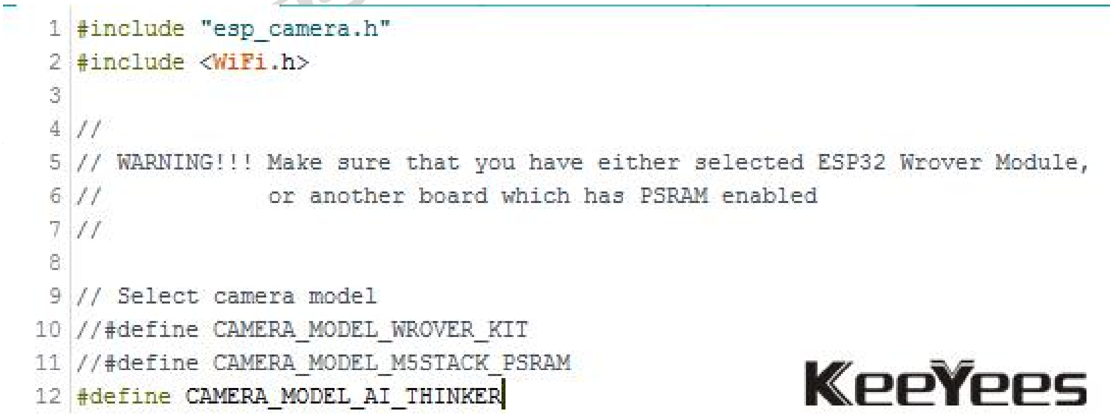
3.

4. The following code should load.



```
 1 #include "esp_camera.h"
 2 #include <WiFi.h>
 3
 4 //
 5 // WARNING!!! Make sure that you have either selected ESP32 Wrover Module
 6 //            or another board which has PSRAM enabled
 7 //
 8
 9 // Select camera model
10 #define CAMERA_MODEL_WROVER_KIT
11 //#define CAMERA_MODEL_M5STACK_PSRAM
12 //#define CAMERA_MODEL_AI_THINKER
13
14 const char* ssid = "*********";
15 const char* password = "*********";
```

5. Make sure you select the right camera module. In this case, we’re using the AI-THINKER Model. Comment the 10th line of code #define CAMERA_MODEL_WROVER_KIT and uncomment the 12th line of code #define CAMERA_MODEL_AI_THINKER



```
 1 #include "esp_camera.h"
 2 #include <WiFi.h>
 3
 4 //
 5 // WARNING!!! Make sure that you have either selected ESP32 Wrover Module,
 6 //            or another board which has PSRAM enabled
 7 //
 8
 9 // Select camera model
10 //#define CAMERA_MODEL_WROVER_KIT
11 //#define CAMERA_MODEL_M5STACK_PSRAM
12 #define CAMERA_MODEL_AI_THINKER
```

6. You need to insert your network credentials in the following variables:

const char* ssid = "REPLACE_WITH_YOUR_SSID";

const char* password = "REPLACE_WITH_YOUR_PASSWORD";

Do make sure that you enter the correct SSID and Password.

7. Now, the code is ready to be uploaded to your ESP32

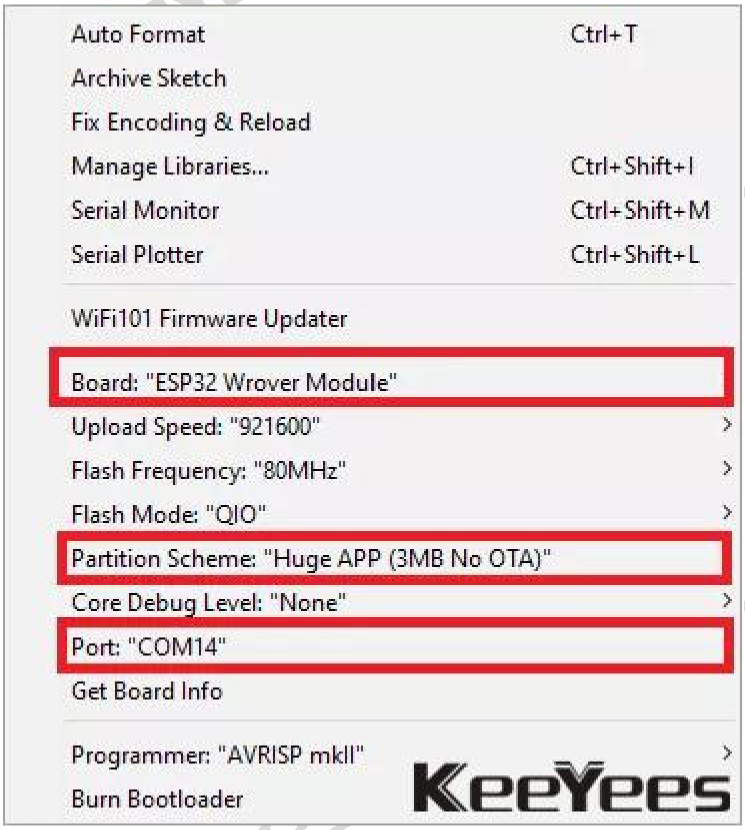
To upload the code, follow the next steps:

Go to Tools > Board and select ESP32 Wrover Module

Go to Tools > Port and select the COM port the ESP32 is connected to

In Tools > Partition Scheme, select “Huge APP (3MB No OTA)“

Then, click the upload button to upload the code.

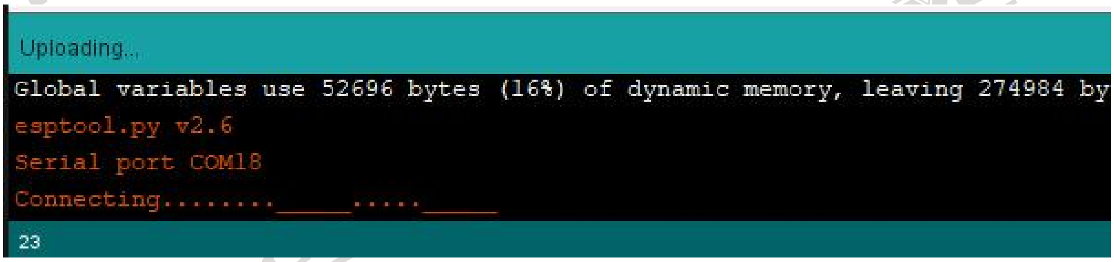


```
Auto Format                      Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries...              Ctrl+Shift+I
Serial Monitor                   Ctrl+Shift+M
Serial Plotter                   Ctrl+Shift+L

WiFi101 Firmware Updater

Board: "ESP32 Wrover Module"
Upload Speed: "921600"                        >
Flash Frequency: "80MHz"                      >
Flash Mode: "QIO"                             >
Partition Scheme: "Huge APP (3MB No OTA)"
Core Debug Level: "None"                      >
Port: "COM14"
Get Board Info

Programmer: "AVRISP mkII"                     >
Burn Bootloader
```

Important: if you can’t upload the code, double-check that GPIO 0 is connected to GNDs and that you selected the right settings in the Tools menu. You should also press the on-board Reset button to

restart your ESP32 in flashing mode.

8. If the problem shown in the figure below appears during the download process, please click the reset button on the development board.



```
Uploading...
Global variables use 52696 bytes (16%) of dynamic memory, leaving 274984 by
esptool.py v2.6
Serial port COM18
Connecting........_____....._____
23
```

Important: If the program still can't be downloaded to the development board, then you need to switch the jumper cap on the FTDI232 module to 3.3v. The 5V pin on the development board also switches to 3.3V, plug and unplug the power supply and download the program.

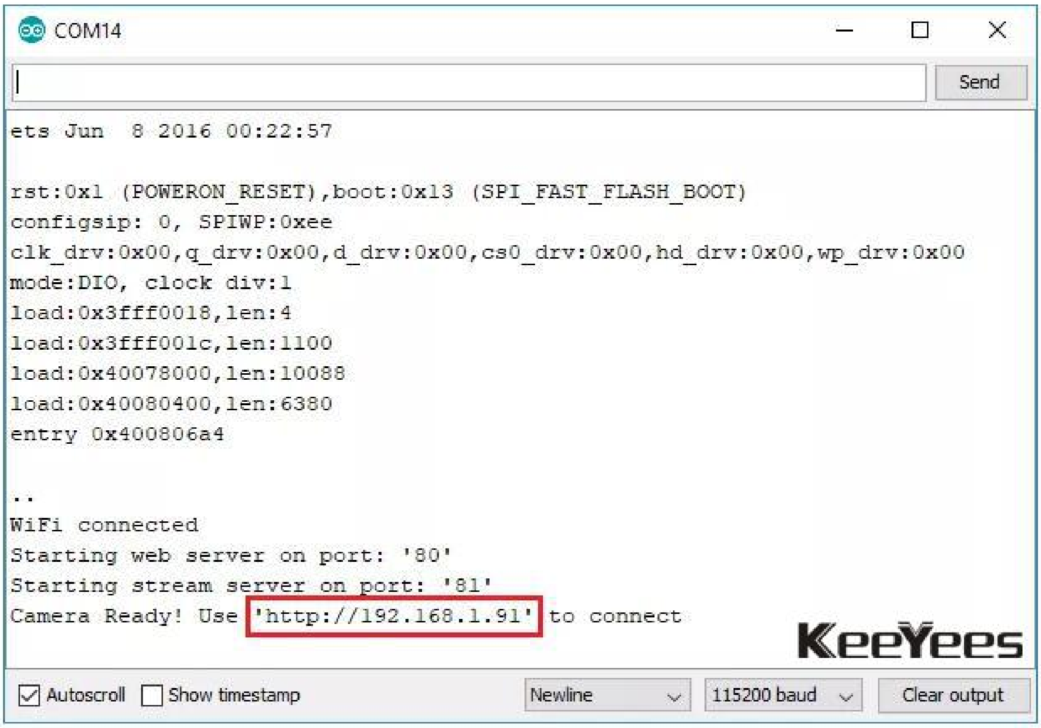
After the download is complete, switch the jumper cap on the FIDI232 module to 5v and switch the power cord of the ESP32-S module to 5v.

9. After uploading the code, disconnect GPIO 0 from GND.

10. Getting the IP address

Open the Serial Monitor at a baud rate of 115200. Press the ESP32-CAM on-board Reset button.

The ESP32 IP address should be printed in the Serial Monitor.

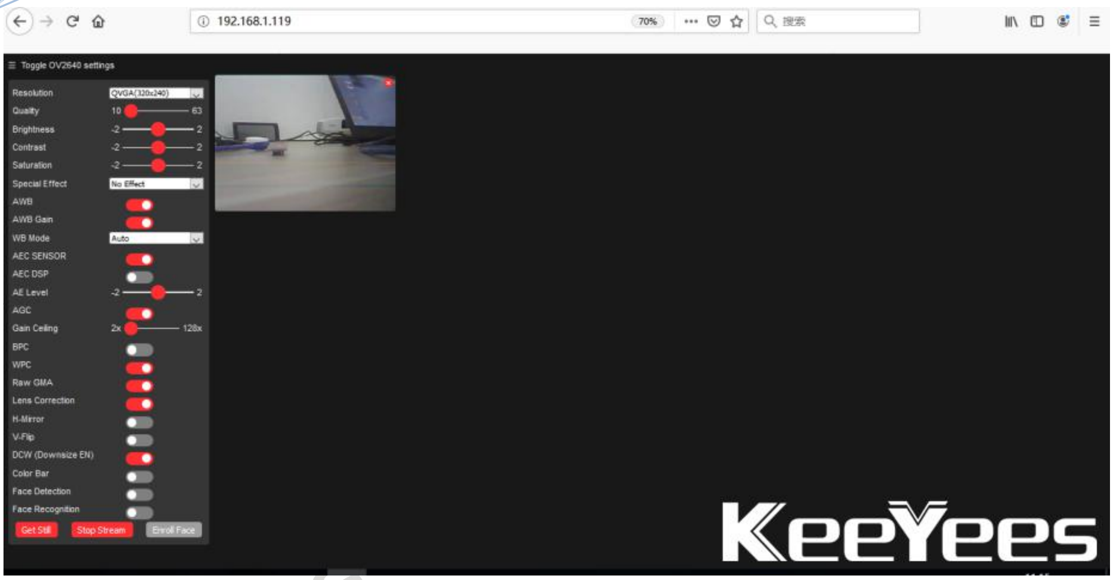


```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1100
load:0x40078000,len:10088
load:0x40080400,len:6380
entry 0x400806a4

..
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.1.91' to connect
```

☑ Autoscroll □ Show timestamp · 115200 baud

11. Accessing the Video Streaming Server

Now, you can access your camera streaming server on your local network. Open a browser and type the ESP32-CAM IP address. Press the Start Streaming button to start video streaming.



192.168.1.119

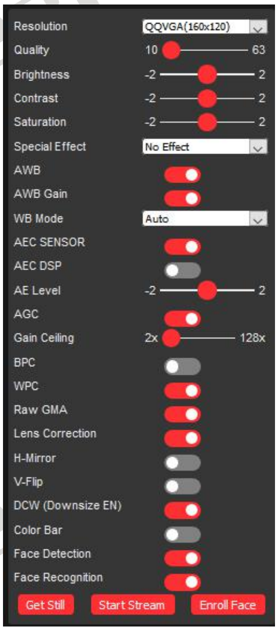
Toggle OV2640 settings

- Resolution: QVGA(320x240)
- Quality: 10 — 63
- Brightness: -2 — 2
- Contrast: -2 — 2
- Saturation: -2 — 2
- Special Effect: No Effect
- AWB ☑
- AWB Gain ☑
- WB Mode: Auto
- AEC SENSOR ☑
- AEC DSP ☐
- AE Level: -2 — 2
- AGC ☑
- Gain Ceiling: 2x — 128x
- BPC ☐
- WPC ☑
- Raw GMA ☑
- Lens Correction ☑
- H-Mirror ☐
- V-Flip ☐
- DCW (Downsize EN) ☑
- Color Bar ☐
- Face Detection ☐
- Face Recognition ☐

You also have the option to take photos by clicking the Get Still button. There are also several camera settings that you can play with to adjust the image settings.

Finally, you can do face recognition and detection.

1. Click the Face Detection and Face Recognition buttons (if the resolution is larger, you need to make it smaller)



- Resolution: QQVGA(160x120)
- Quality: 10 — 63
- Brightness: -2 — 2
- Contrast: -2 — 2
- Saturation: -2 — 2
- Special Effect: No Effect
- AWB ☑
- AWB Gain ☑
- WB Mode: Auto
- AEC SENSOR ☑
- AEC DSP ☐
- AE Level: -2 — 2
- AGC ☑
- Gain Ceiling: 2x — 128x
- BPC ☐
- WPC ☑
- Raw GMA ☑
- Lens Correction ☑
- H-Mirror ☐
- V-Flip ☐
- DCW (Downsize EN) ☑
- Color Bar ☐
- Face Detection ☑
- Face Recognition ☑

2. You need to enroll a new face. It will make several attempts to save

the face. After enrolling a new user, it should detect the face later on (subject 0).

And that’s it. Now you have your video streaming web server up and running with face detection and recognition.

Problems you may have

Issue 1

```
camera_probe(): Detected camera not supported
SCCB_Write(): SCCB_Write Failed addr:0x30, reg:0xff, data:0x01, ret:-1
SCCB_Write(): SCCB_Write Failed addr:0x30, reg:0xff, data:0x01, ret:-1
SCCB_Write(): SCCB_Write Failed addr:0x30, reg:0x12 data:0x01, ret:-1
```

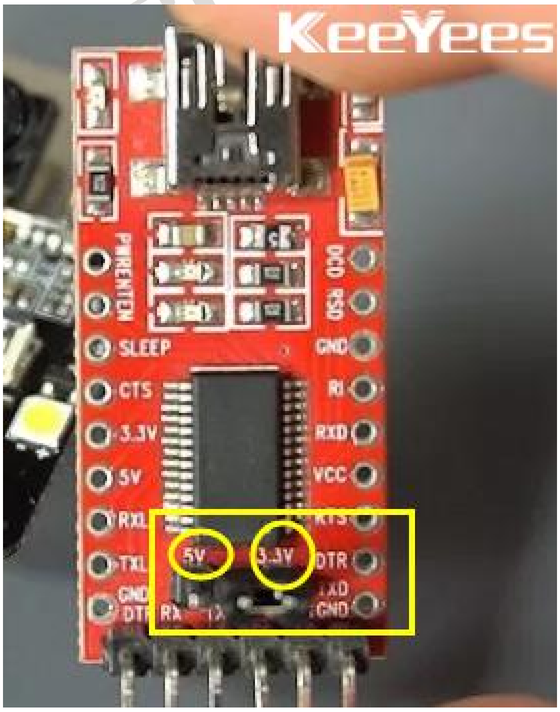
esp_camera_init(): Camera probe failed with error 0x20004

Reason: OV2640 camera module is not plugged well!

Issue 2

brownout detector was triggered

Reason: This problem may occur when you start connecting to WiFi! The FT232RL FTDI mini USB to TTL serial converter Module has a trigger function. If the voltage is too low, the board will automatically restart to protect the board.



How to solve it?

1. Set the usb-TTL to 3.3V

2. Connect it to the ESP32-CAM as shown in all the diagrams (but put the 3.3V from the usb-Tl to 3.3V on the ESP32-CAM.)

3. Connect the Io0 and gnd （Make sure the pins are correct. It's very important.）

4. Power up and upload the code

Now test the ESP32-CAM

1. Remove the IO0 and gnd jumper

2. Change the usb-TTL to 5v (changing the pin)

3. Change the voltage on the ESP32-CAM to 5V pin

4. Power up

5. Open up the serial monitor

6. Press the reset button on the ESP32-CAM

7. Get the IP address

Issue 3

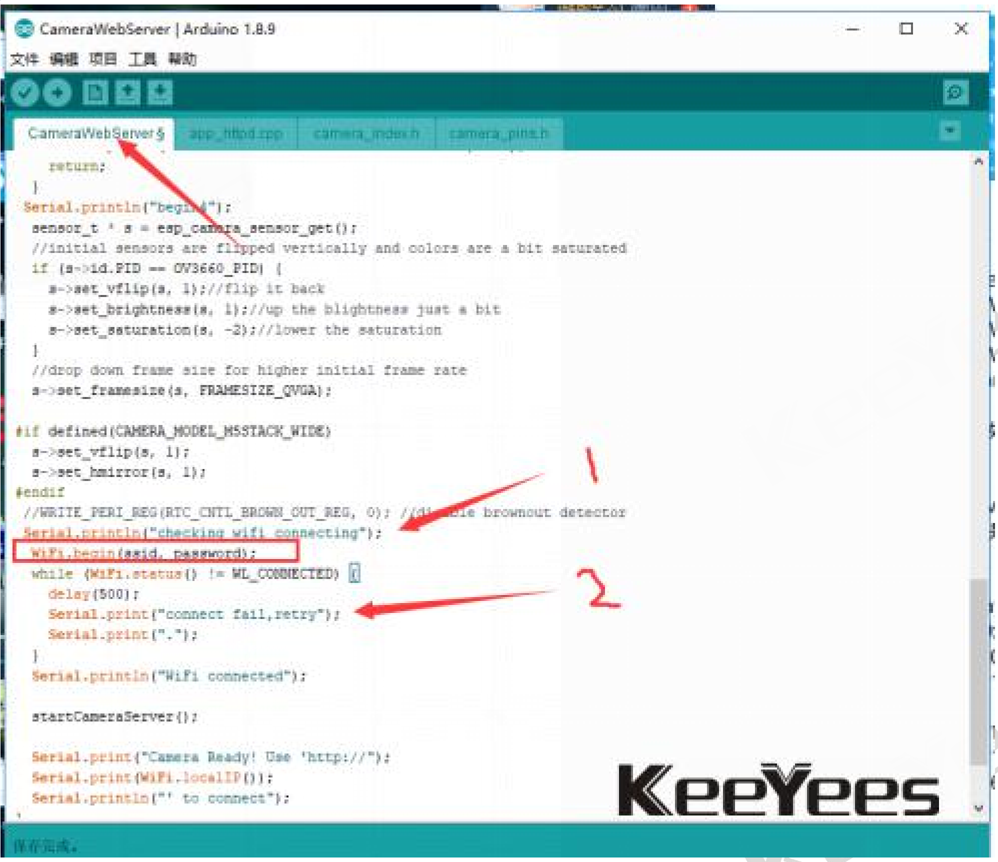
```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:
```

```
0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1100
load:0x40078000,len:10088
load:0x40080400,len:6380
entry 0x400806a4
```

Reason: No IP address. Wifi is not connected successfully.

```
Serial.println("checking wifi connecting");
Serial.print("connect fail,retry");
```

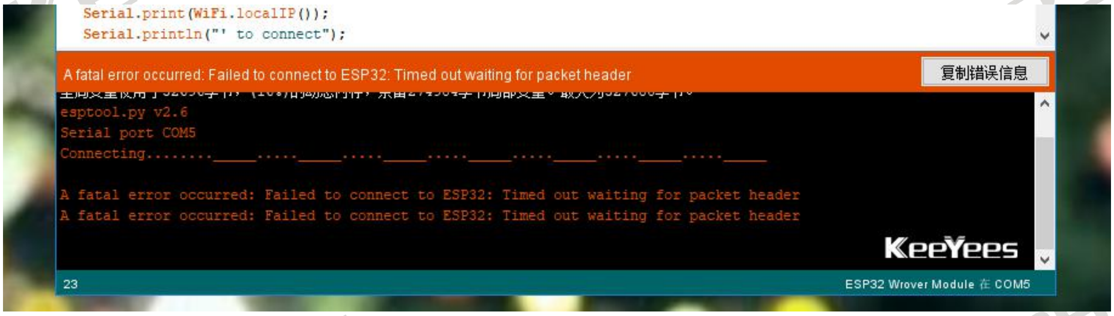
Double confirm that the WiFi SSID and password are 100% correct.



```
    return;
  }
 Serial.println("begin4");
  sensor_t * s = esp_camera_sensor_get();
  //initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);//flip it back
    s->set_brightness(s, 1);//up the blightness just a bit
    s->set_saturation(s, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif
 //WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
 Serial.println("checking wifi connecting");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print("connect fail,retry");
    Serial.print(".");
  }
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
```

Issue 4

Upload failed



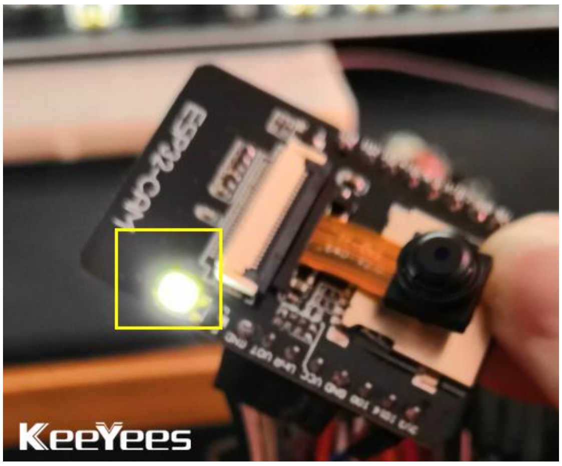
```
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
```

A fatal error occurred: Failed to connect to ESP32: Timed out waiting for packet header

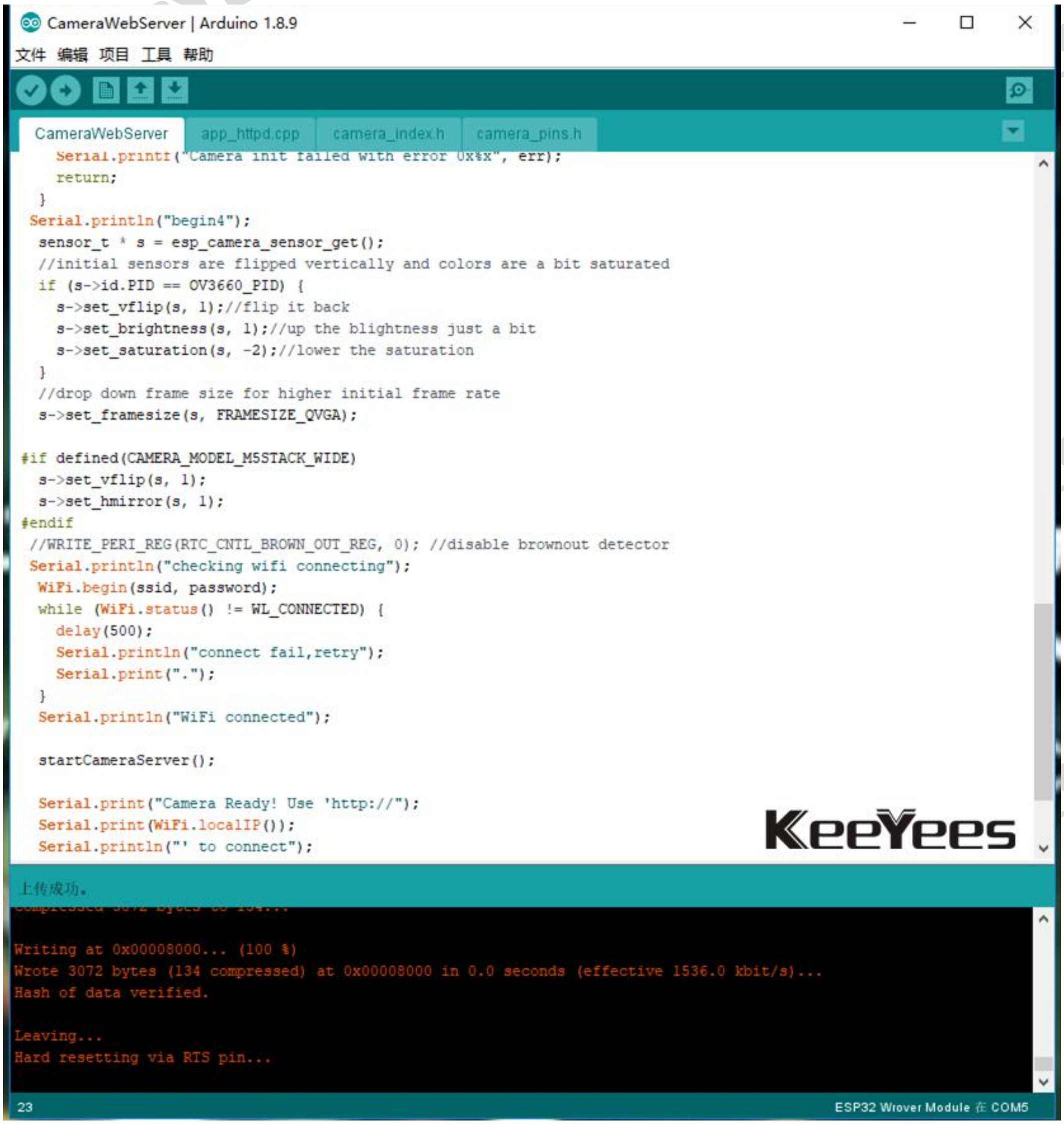
```
esptool.py v2.6
Serial port COM5
Connecting........_____....._____....._____....._____....._____....._____....._____

A fatal error occurred: Failed to connect to ESP32: Timed out waiting for packet header
A fatal error occurred: Failed to connect to ESP32: Timed out waiting for packet header
```

Make sure that the RXD and TXD pins of the ESP32-S module are correctly connected to the FTDI232 module pins.

Press the reset button to confirm that the light is on.



After the upload is successful, disconnect the GND and GPIO0. Back to CameraWebServer to restart.



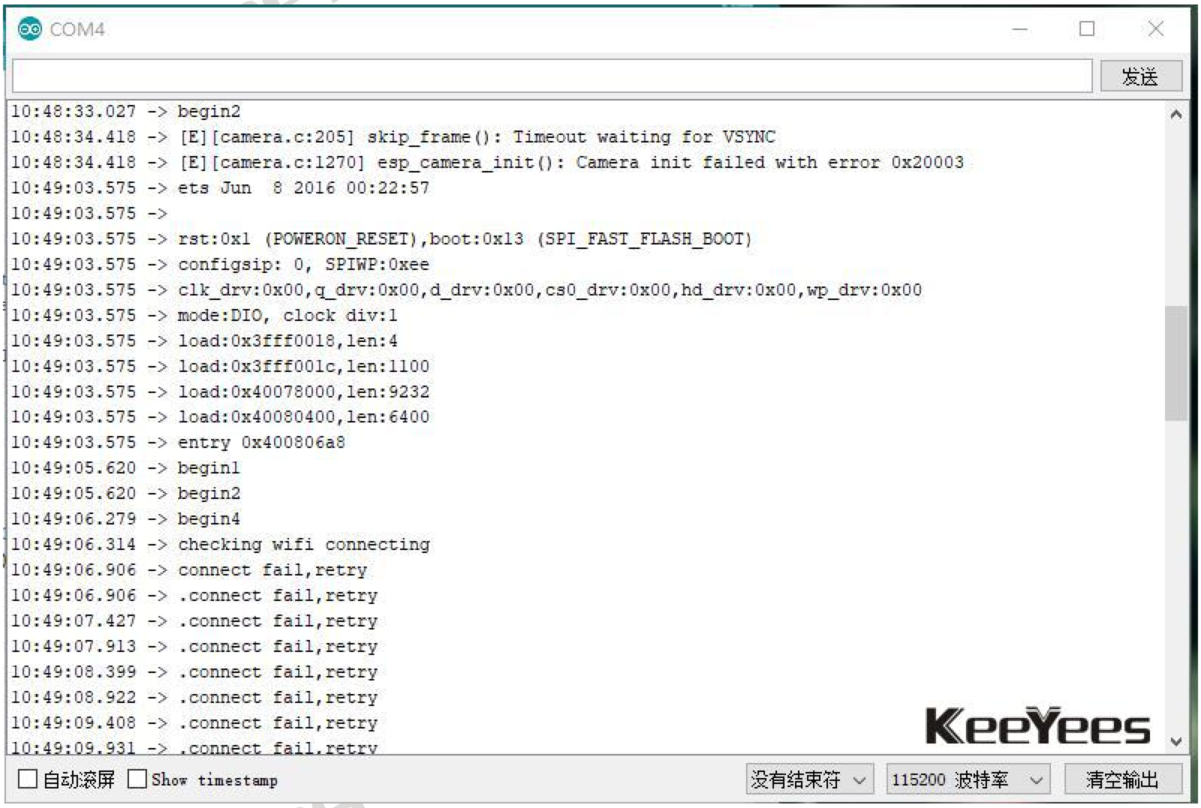
```
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
 Serial.println("begin4");
  sensor_t * s = esp_camera_sensor_get();
  //initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);//flip it back
    s->set_brightness(s, 1);//up the blightness just a bit
    s->set_saturation(s, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif
 //WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
 Serial.println("checking wifi connecting");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("connect fail,retry");
    Serial.print(".");
  }
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
```

上传成功。

```
Compressed 3072 bytes to 134...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (134 compressed) at 0x00008000 in 0.0 seconds (effective 1536.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

23   ESP32 Wrover Module 在 COM5

If still failed, change the usb-TTl to 5v and the voltage on the ESP32-CAM to 5V. Then reset.



```
10:48:33.027 -> begin2
10:48:34.418 -> [E][camera.c:205] skip_frame(): Timeout waiting for VSYNC
10:48:34.418 -> [E][camera.c:1270] esp_camera_init(): Camera init failed with error 0x20003
10:49:03.575 -> ets Jun  8 2016 00:22:57
10:49:03.575 ->
10:49:03.575 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
10:49:03.575 -> configsip: 0, SPIWP:0xee
10:49:03.575 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
10:49:03.575 -> mode:DIO, clock div:1
10:49:03.575 -> load:0x3fff0018,len:4
10:49:03.575 -> load:0x3fff001c,len:1100
10:49:03.575 -> load:0x40078000,len:9232
10:49:03.575 -> load:0x40080400,len:6400
10:49:03.575 -> entry 0x400806a8
10:49:05.620 -> begin1
10:49:05.620 -> begin2
10:49:06.279 -> begin4
10:49:06.314 -> checking wifi connecting
10:49:06.906 -> connect fail,retry
10:49:06.906 -> .connect fail,retry
10:49:07.427 -> .connect fail,retry
10:49:07.913 -> .connect fail,retry
10:49:08.399 -> .connect fail,retry
10:49:08.922 -> .connect fail,retry
10:49:09.408 -> .connect fail,retry
10:49:09.931 -> .connect fail,retry
```