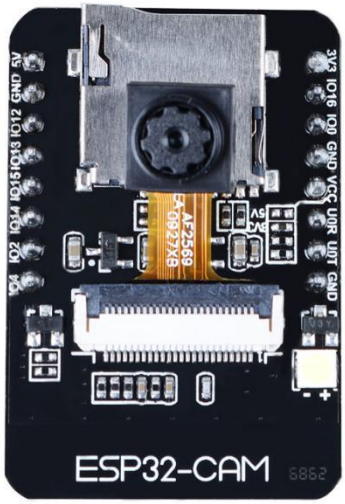
ESP 32 CAM Starter Kit V2.0



20220504V2

catalog

Project 1: DIY surveillance camera with ESP32 camera

In this project "DIY surveillance camera with ESP32-Camera", we will make a DIY surveillance camera for home use to monitor the surrounding environment using #ESP32-Camera Module. The ESP32 camera module will host a web server, where we can watch live surveillance, which will help us understand who will pass the home environment.

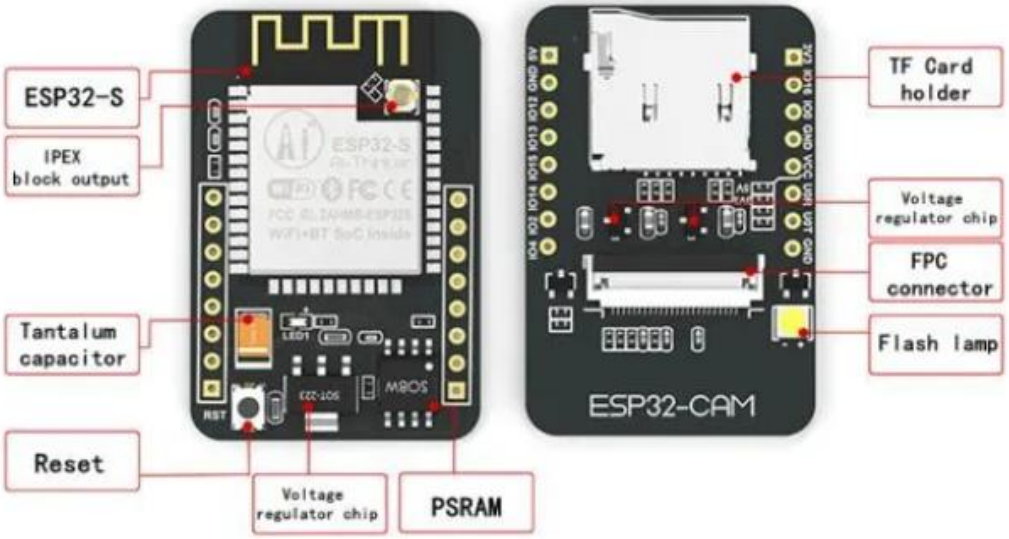
Required components:

ESP32-CAM with OV2640
FTDI programmer
Connecting line
5V power supply for ESP32-CAM

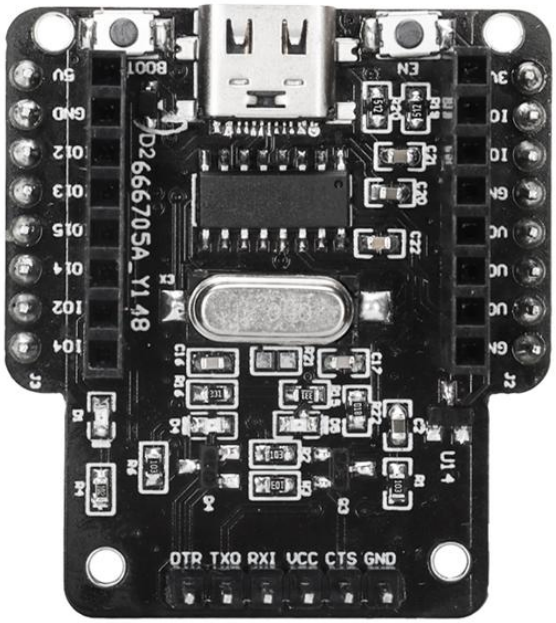
Introduction to ESP32-Cam:

The ESP32-CAM is a very small camera module with the ESP32-S chip, and the cost is less than $10. In addition to the OV2640 camera and several GPIOs for connecting peripherals, it also has a microSD card slot that can be used to store images taken with the camera or store files to be provided to the client.

The components of ESP32-Cam:



Unlike NodemCU-ESP8266, ESP32-CAM does not have its own micro USB onboard connector, so we specially build a burning code extension module for it.

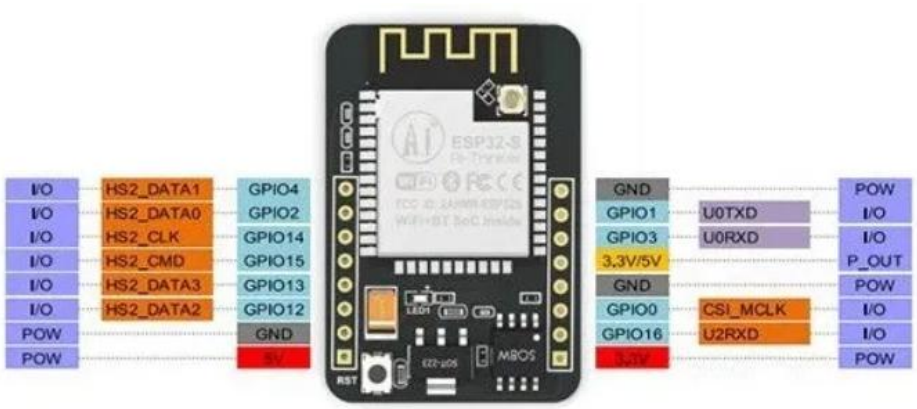


ESP32 camera features:

The following is a list of ESP32-Cam features:

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

ESP32-Cam Pinout (AI-Thinker module):



There are three GND pins and two pins for power supply: 3.3V or 5V.
GPIO 1 and GPIO 3 are serial pins. You need these pins to upload code to the board. In addition, GPIO 0 also plays an important role because it determines whether the ESP32 is in blinking mode. When GPIO 0 is connected to GND, ESP32 is in blinking 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 onboard LED)

GPIO 12: Data 2

GPIO 13: Data 3

Video streaming server configuration:

Please follow the steps below to build a video streaming web server using ESP32-Cam that you can access on the local network.
1. Install ESP32 plugin
In this example, we use the Arduino IDE to program the ESP32-Cam board. Therefore, you need to install the Arduino IDE and ESP32 add-on components. If you have not installed the ESP32 plug-in in your computer, please follow the tutorial to install it.

Refer to the link: "**Installing ESP32 board package to Arduino IDE**"

Source code: DIY surveillance camera with ESP32 camera

Copy and paste the following code into the new Arduino sketch.

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "esp_http_server.h"
```

```
//Replace with your network credentials
const char* ssid = "XXXX";
const char* password = "XXXXX";
#define PART_BOUNDARY "123456789000000000000987654321"
// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
// Not tested with this model
//#define CAMERA_MODEL_WROVER_KIT
#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #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      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22
#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
```

```
    #define VSYNC_GPIO_NUM       22
    #define HREF_GPIO_NUM        26
    #define PCLK_GPIO_NUM        21
#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
    #define PWDN_GPIO_NUM        -1
    #define RESET_GPIO_NUM       15
    #define XCLK_GPIO_NUM        27
    #define SIOD_GPIO_NUM        25
    #define SIOC_GPIO_NUM        23

    #define Y9_GPIO_NUM          19
    #define Y8_GPIO_NUM          36
    #define Y7_GPIO_NUM          18
    #define Y6_GPIO_NUM          39
    #define Y5_GPIO_NUM           5
    #define Y4_GPIO_NUM          34
    #define Y3_GPIO_NUM          35
    #define Y2_GPIO_NUM          17
    #define VSYNC_GPIO_NUM       22
    #define HREF_GPIO_NUM        26
    #define PCLK_GPIO_NUM        21
#elif defined(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
#else
    #error "Camera model not selected"
#endif
static const char* _STREAM_CONTENT_TYPE =
"multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
```

```
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";
httpd_handle_t stream_httpd = NULL;
static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];
  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }
  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
```

```
strlen(_STREAM_BOUNDARY));
        }
        if(fb){
            esp_camera_fb_return(fb);
            fb = NULL;
            _jpg_buf = NULL;
        } else if(_jpg_buf){
            free(_jpg_buf);
            _jpg_buf = NULL;
        }
        if(res != ESP_OK){
            break;
        }
        //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
    }
    return res;
}
void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();
    config.server_port = 80;
    httpd_uri_t index_uri = {
        .uri            = "/",
        .method         = HTTP_GET,
        .handler        = stream_handler,
        .user_ctx       = NULL
    };

    //Serial.printf("Starting web server on port: '%d'\n", config.server_port);
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(stream_httpd, &index_uri);
    }
}
void setup() {
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

    Serial.begin(115200);
    Serial.setDebugOutput(false);

    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()){
   config.frame_size = FRAMESIZE_UXGA;
   config.jpeg_quality = 10;
   config.fb_count = 2;
} else {
   config.frame_size = FRAMESIZE_SVGA;
   config.jpeg_quality = 12;
   config.fb_count = 1;
}

// Camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
   Serial.printf("Camera init failed with error 0x%x", err);
   return;
}
// Wi-Fi connection
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

Serial.print("Camera Stream Ready! Go to: http://");
Serial.print(WiFi.localIP());
```

```
  // Start streaming web server
  startCameraServer();
}
void loop() {
  delay(1);
}
```

Note: Need to be modified in the above sketch:

You need to make some modifications before uploading the code. Insert the network credentials into the following variables:

const char* ssid = "REPLACE_WITH_YOUR_WIFI_SSID";

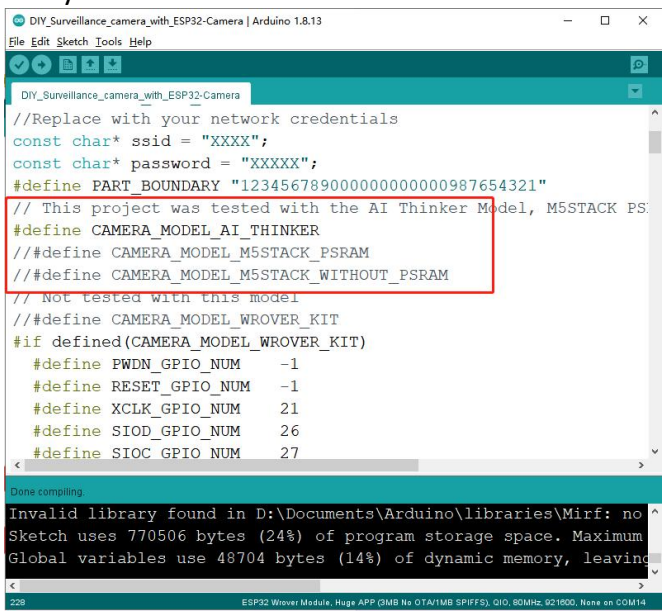
const char* password = "REPLACE_WITH_YOUR_WIFI_PASSWORD";

Then, make sure to select the correct camera module. Our correct choice is the AI-THINKER model.

Therefore, please comment all other models and uncomment the appropriate model as follows:

#define CAMERA_MODEL_AI_THINKER

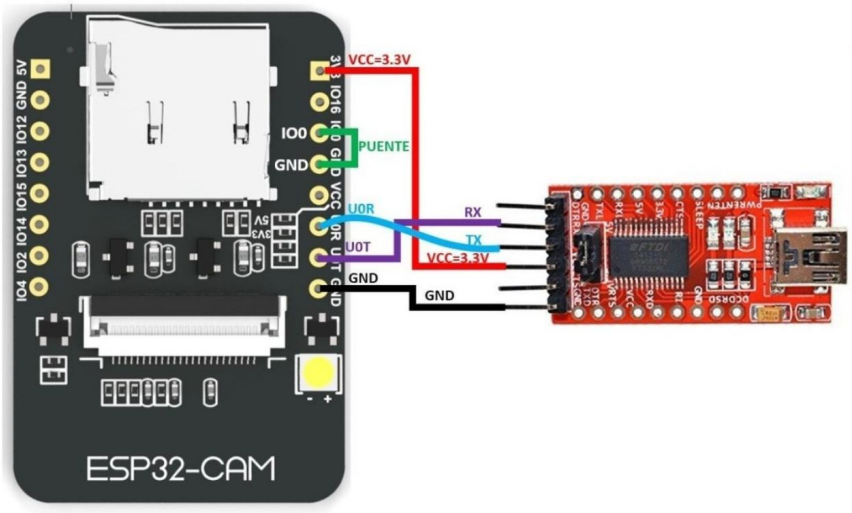
Now, the code is ready to flash into the ESP32-Cam module.



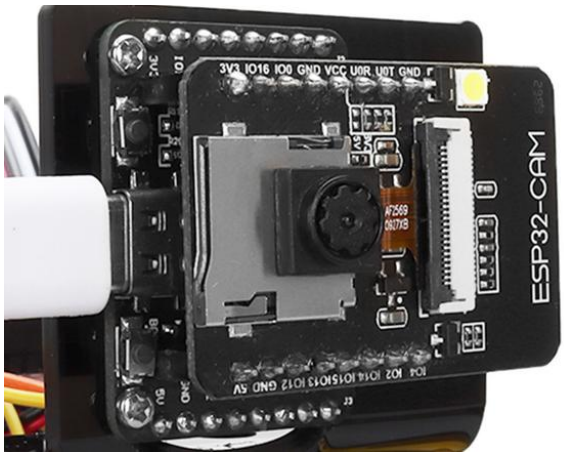
ESP32-CAM code upload:

Note: If you have an older version you can refer to the FTDI burn wiring diagram.

Use the FTDI programmer to connect the ESP32-CAM development board to the computer. The program interface of ESP32-cam and FTDI is shown in the figure below:



If it is the latest version, please refer to the following picture, insert USB data cable for installation to burn the code normally, without extra tedious wiring.



note:

GPIO 0 needs to be connected to GND to be able to upload code.

To upload the code, please follow the steps below:

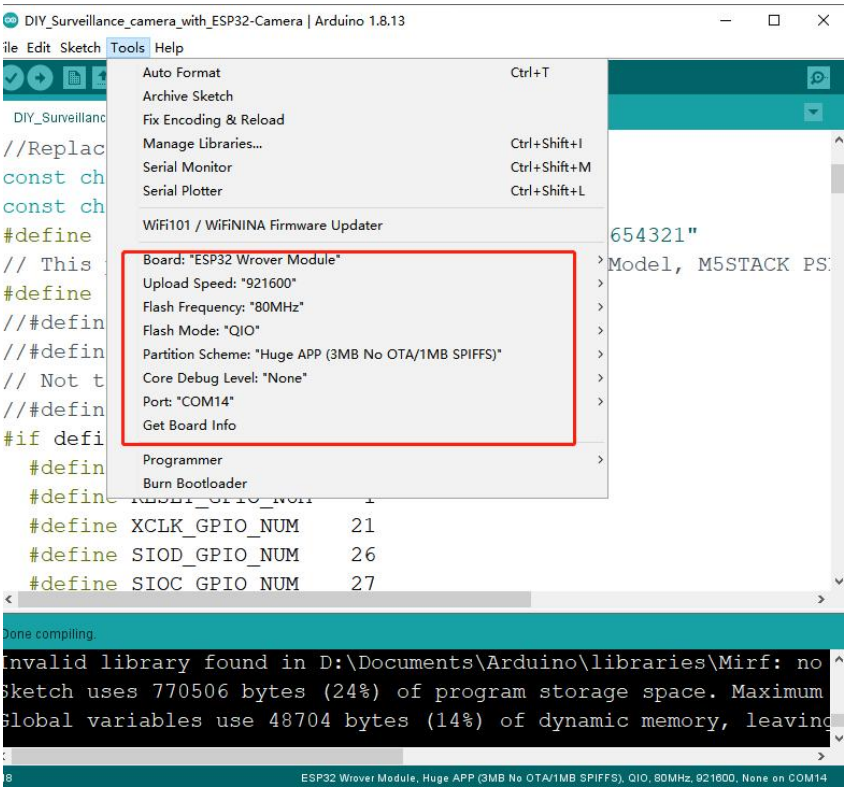
Go to Tools> Development Board and select ESP32 Dev Module

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

In "Tools"> "Partition Plan", select "Huge APP (3MB or OTA)"

Press the onboard RESET button of ESP32-CAM

Then, click the upload button to upload the code:



Note: If you cannot upload the code, please double check whether GPIO 0 is connected to GND and whether the correct setting is selected in the "Tools" menu. You should also press the Reset button on the board to restart the ESP32 in flashing mode.
When you start to see these points on the debug window, as shown below, please press the ESP32-CAM onboard button.

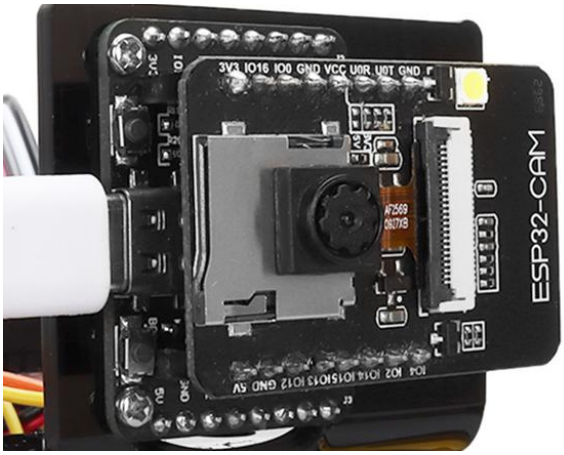
```
esptool.py v2.6-beta1
Serial port COM10
Connecting.....____.....____.....____
```

After a few seconds, the code should be successfully uploaded to your circuit board.

Upload complete: Hard reset via RTS pins...

Don't panic. It's not a mistake. Tell us that ESP32-CAM will reset once the code is successfully uploaded. This is a new feature for ArduinoIDE. This feature is available in 1.8.0 or later.

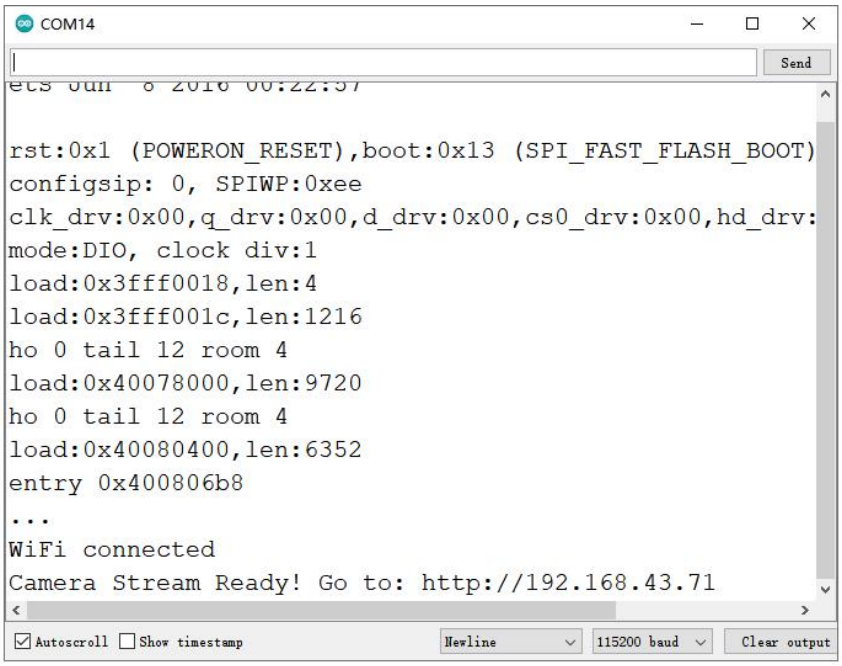
Obtain the local IP address of ESP32-Cam Server:



Normally, there is no need to manually press the reset switch when writing code by default. In special cases, you can manually press the reset switch when burning.

Turn on the serial monitor at a baud rate of 115200. Press the reset button on the ESP32-CAM board.

As shown below, the "WiFi connection status", the ESP32 IP address of the "Camera" module and the Web server will be printed in the "Serial Monitor".



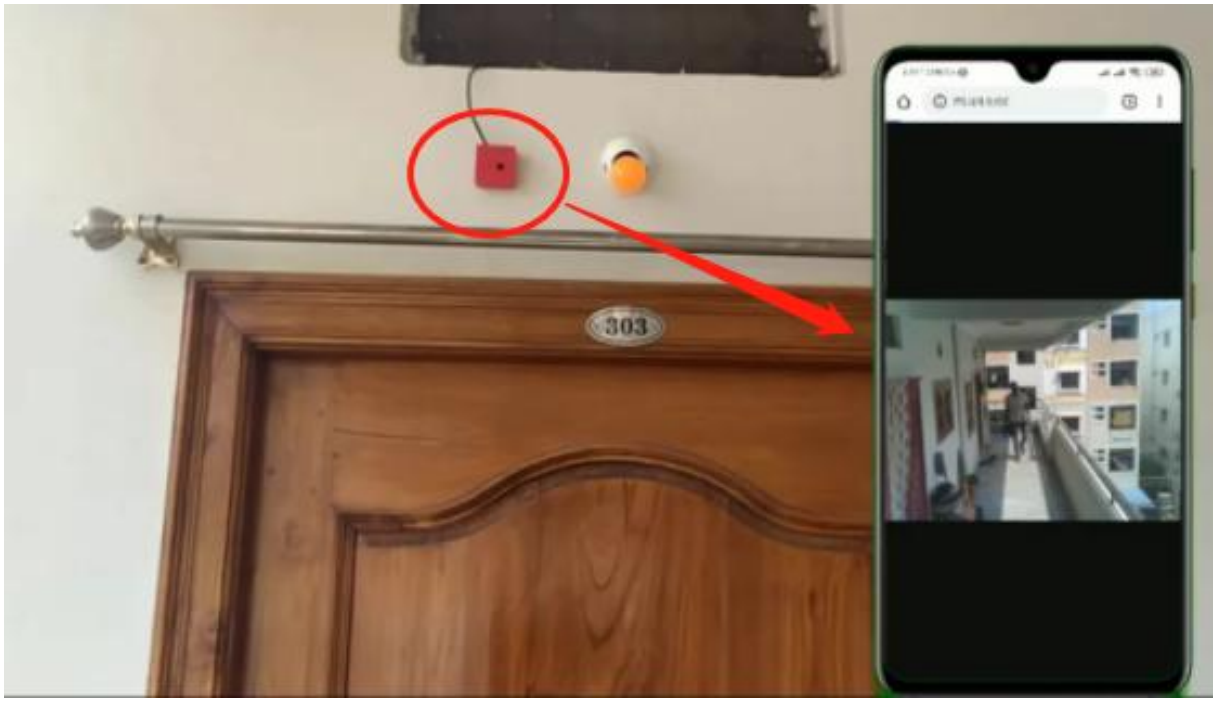
Video streaming server:

Now you can access the camera streaming server on the local network.

Open your favorite browser, enter the IP address of ESP32-CAM, and press Enter to load the Video Streaming page.

When we install the equipment in the monitoring area, to protect the equipment from environmental dust and moisture, you can also design your own interesting shell and print it through the 3D printer. Install the device in the monitoring area, as shown below, and you can see that the image quality is good.

Wish you a successful experiment!



Project 2:ESP32 Camera Surveillance Car

This project will introduce you to a highly interactive ESP32 CAM game. Use ESP32 CAM to make a camera programmable car with LED light control and motor control.

Required components:

- OV2640 ESP32 - CAM
- FTDI programmer
- Connecting line
- L298N Motor Drive Module
- 18650 battery box

Introduction to ESP32-Cam:

The ESP32-CAM is a very small camera module and ESP32-S chip. In addition to the OV2640 camera and several GPIOs for connecting peripherals, it also has a microSD card slot, which can be used to store images taken with the camera or storage The file to be provided to the client.

L298N Motor Drive Module:

Product parameters:

1. Driver chip: L298N dual H-bridge DC motor driver chip

2. The power supply range of the drive part of the terminal Vs: +5V～＋35V; if the board needs to be powered, the power supply range Vs: +7V～+35V

3. The peak current Io of the driving part: 2A

4. The power supply range Vss of the logic part terminal: +5V～＋7V (+5V can be taken from the board)

5. Logic part operating current range: 0～36mA

6. Control signal input voltage range:

Low level: -0.3V≤Vin≤1.5V

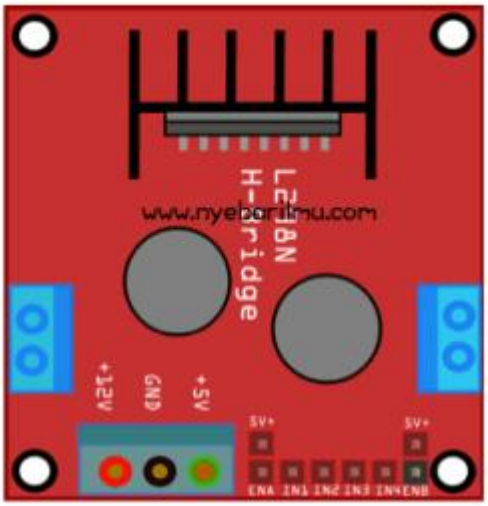
High level: 2.3V≤Vin≤Vss

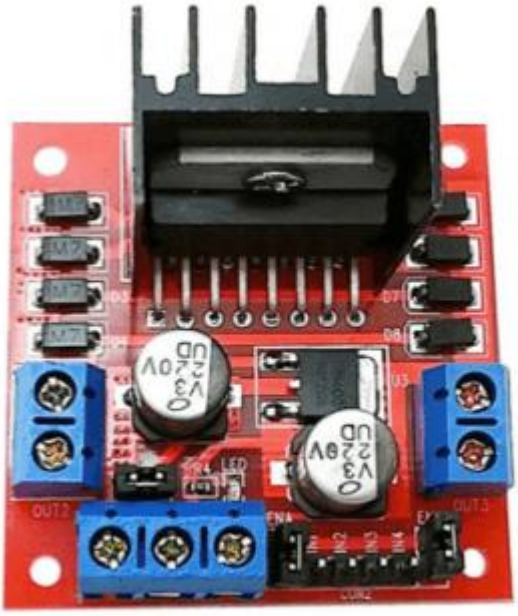
7. Enable signal input voltage range:

Low level: -0.3≤Vin≤1.5V (control signal is invalid)

High level: 2.3V≤Vin≤Vss (control signal is valid)

8. Maximum power consumption: 20W (when temperature T=75℃)





Source code:

```
#include "esp_camera.h"

#include <WiFi.h>

//

// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,

//                    or another board which has PSRAM enabled

//

// Adafruit ESP32 Feather

// Select camera model

//#define CAMERA_MODEL_WROVER_KIT

//#define CAMERA_MODEL_M5STACK_PSRAM

#define CAMERA_MODEL_AI_THINKER


const char* ssid = "    XXXX";      //Enter SSID WIFI Name

const char* password = "YYYYYYYYY";     //Enter WIFI Password


#if defined(CAMERA_MODEL_WROVER_KIT)

#define PWDN_GPIO_NUM       -1

#define RESET_GPIO_NUM      -1

#define XCLK_GPIO_NUM       21
```

```
#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         19
#define Y4_GPIO_NUM         18
#define Y3_GPIO_NUM          5
#define Y2_GPIO_NUM          4
#define VSYNC_GPIO_NUM      25
#define HREF_GPIO_NUM       23
#define PCLK_GPIO_NUM       22

#elif defined(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
#else
#error "Camera model not selected"
#endif

// GPIO Setting
extern int gpLb =   2; // Left 1
extern int gpLf = 14; // Left 2
extern int gpRb = 15; // Right 1
extern int gpRf = 13; // Right 2
extern int gpLed =   4; // Light
```

```
extern String WiFiAddr ="";


void startCameraServer();


void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();



  pinMode(gpLb, OUTPUT); //Left Backward
  pinMode(gpLf, OUTPUT); //Left Forward
  pinMode(gpRb, OUTPUT); //Right Forward
  pinMode(gpRf, OUTPUT); //Right Backward
  pinMode(gpLed, OUTPUT); //Light


  //initialize
  digitalWrite(gpLb, LOW);
  digitalWrite(gpLf, LOW);
  digitalWrite(gpRb, LOW);
  digitalWrite(gpRf, LOW);
  digitalWrite(gpLed, LOW);
```

```
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;
```

```
//init with high specs to pre-allocate larger buffers
if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
} else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
}

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
}

//drop down frame size for higher initial frame rate
sensor_t * s = esp_camera_sensor_get();
s->set_framesize(s, FRAMESIZE_CIF);
```

```
    WiFi.begin(ssid, password);


    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");


    startCameraServer();


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


void loop()
{
// put your main code here, to run repeatedly:


}
```

Note: In the sketch above, it needs to be modified:

You'll need to make some changes before you upload the code. Insert the network credentials into the following variables:

const char* ssid = " XXXX"; //Enter SSID WIFI Name

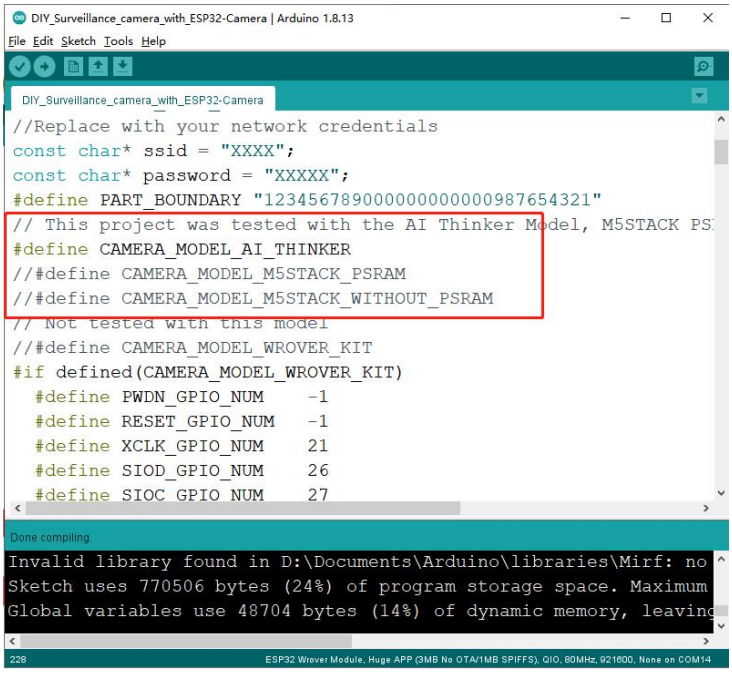
const char* password = "YYYYYYYYY"; //Enter WIFI Password

Then, make sure you choose the right camera module. Our correct choice was the AI-patience model.

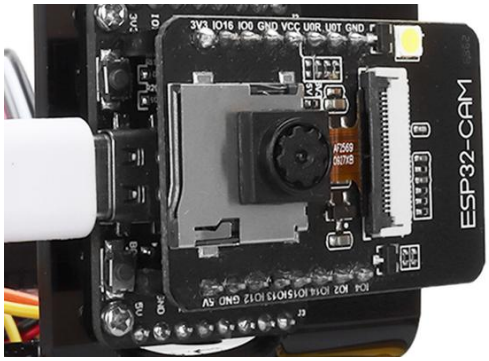
herefore, annotate all other models and unannotate the appropriate models, as shown below:

#define CAMERA_MODEL_AI_THINKER

Now the code is ready to be swiped into the ESP32-CAM module.



ESP32-CAM code upload:



Note:

GPIO 0 needs to be connected to GND to be able to upload code.

To upload the code, please follow the steps below:

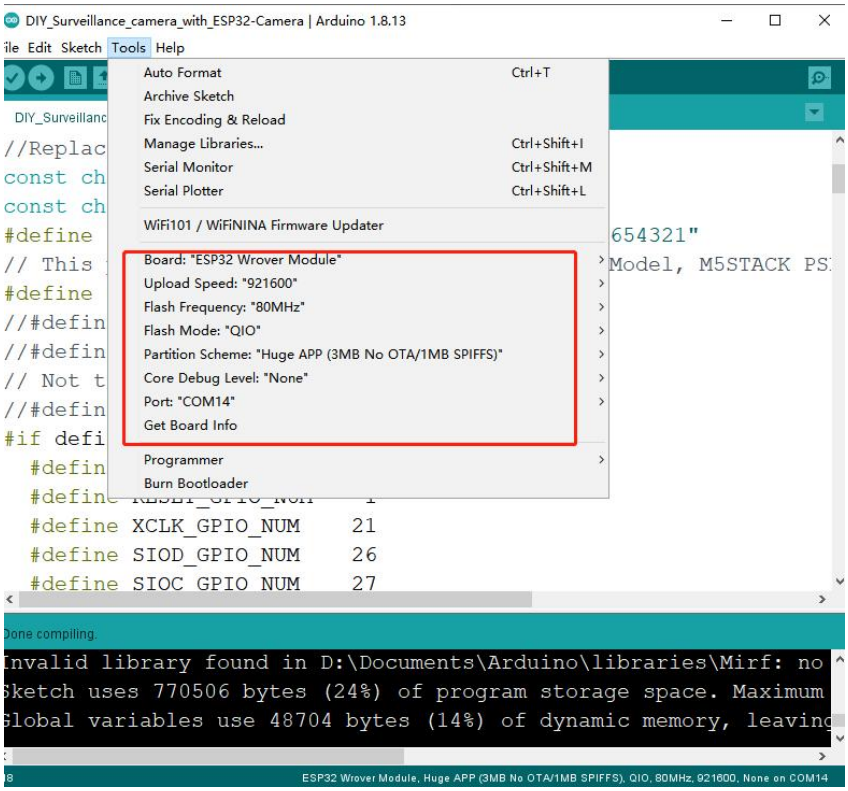
Go to Tools> Development Board and select ESP32 Dev Module

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

to In "Tools"> "Partition Plan", select "Huge APP (3MB or OTA)"

Press the onboard RESET button of ESP32-CAM

Then, click the upload button to upload the code:



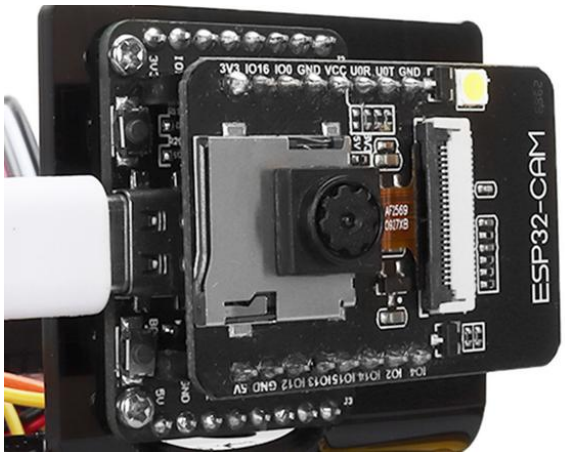
Note: If you cannot upload the code, please double check whether GPIO 0 is connected to GND and whether the correct setting is selected in the "Tools" menu. You should also press the Reset button on the board to restart the ESP32 in flashing mode.
When you start to see these points on the debug window, as shown below, please press the ESP32-CAM onboard button.

After a few seconds, the code should be successfully uploaded to your circuit board.

Upload complete: Hard reset via RTS pins...

Don't panic. It's not a mistake. Tell us that ESP32-CAM will reset once the code is successfully uploaded. This is a new feature for ArduinoIDE. This feature is available in 1.8.0 or later.

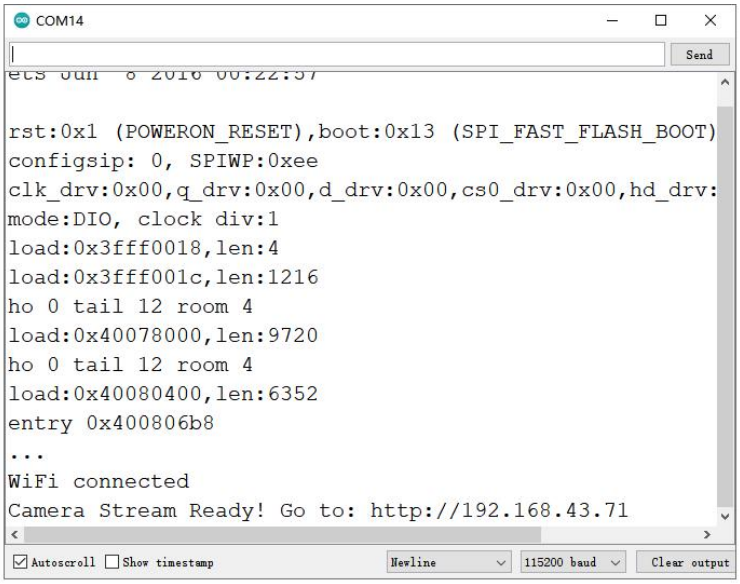
Get the local IP address of ESP32-CAM Server:



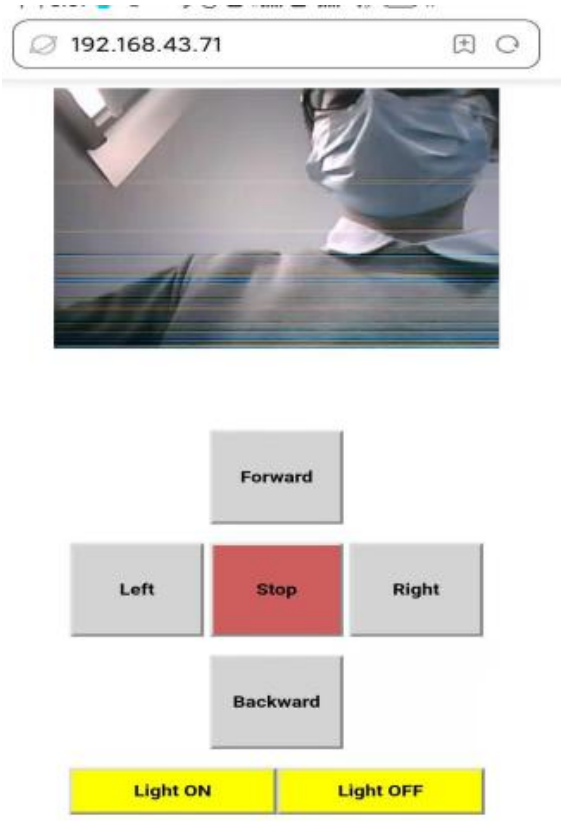
Normally, there is no need to manually press the reset switch when writing code by default. In special cases, you can manually press the reset switch when burning.

Turn on the serial monitor at a baud rate of 115200. Press the reset button on the ESP32-CAM board.

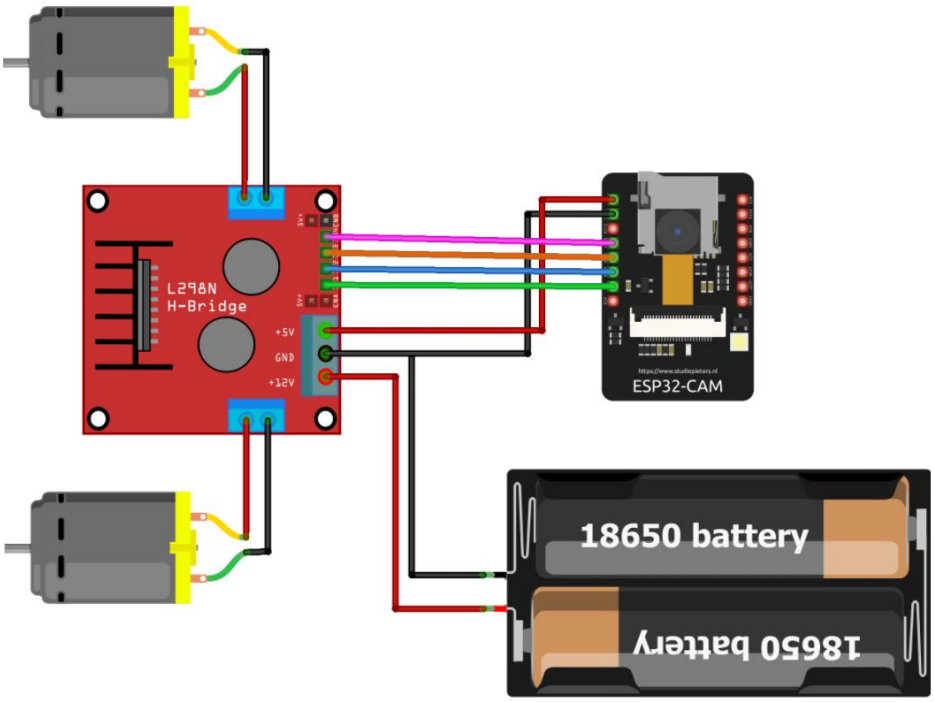
As shown below, the WiFi connection status, the Camera module, and the Web server's ESP32 IP address will be printed in the Serial Monitor.



You can now access the camera stream server on your local network. Open your favorite browser, Enter the IP address of ESP32-CAM, and then press Enter to load the Video Streaming page.



Trolley circuit wiring diagram:



Physical ma

Project3:ESP32-CAM Take Photo and Save to MicroSD Card

Learn how to take photos with the ESP32-CAM board and save them to a microSD card using Arduino IDE. When you press the ESP32-CAM RESET button, it wakes up, takes a photo and saves it in the microSD card.

We'll be using the ESP32-CAM board labelled as AI-Thinker module, but other modules should also work by making the correct pin assignment in the code.

Parts required:

To follow this tutorial, you need the following components:

ESP32 -CAM development board with OV2640

MicroSD 卡

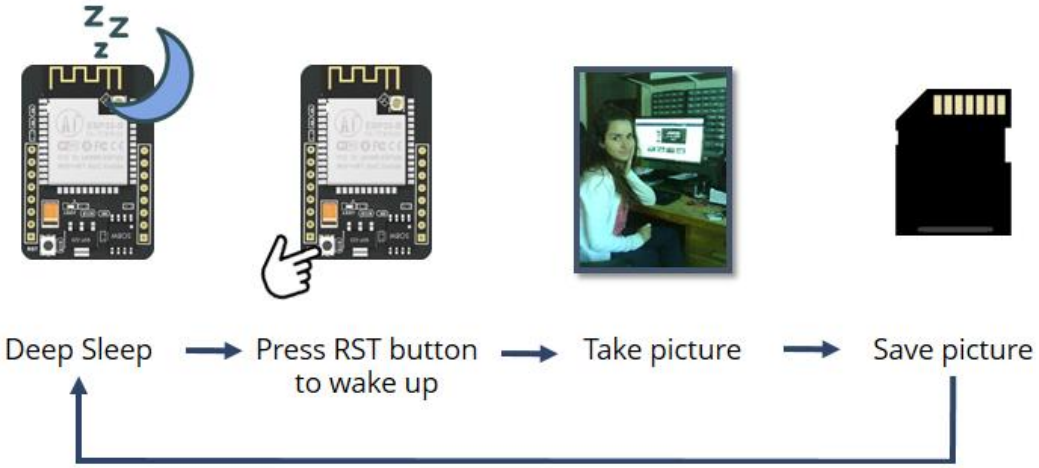
FTDI programmer

Female to female jumper

5V power supply for ESP32-CAM or mobile power supply (optional)

Project Overview：

Here is a quick overview on how the project works.



The ESP32-CAM is in deep sleep mode

Press the RESET button to wake up the board

The camera takes a photo

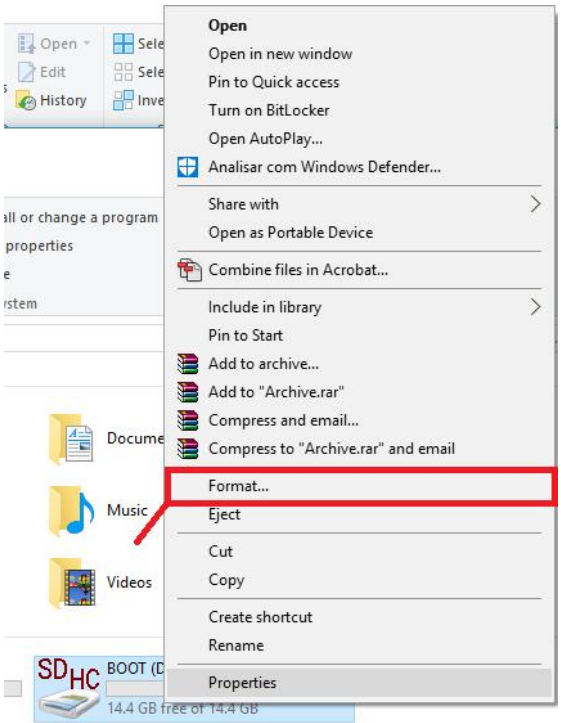
The photo is saved in the microSD card with the name: pictureX.jpg, where X corresponds to the picture number

The picture number will be saved in the ESP32 flash memory so that it is not erased during RESET and we can keep track of the number of photos taken.

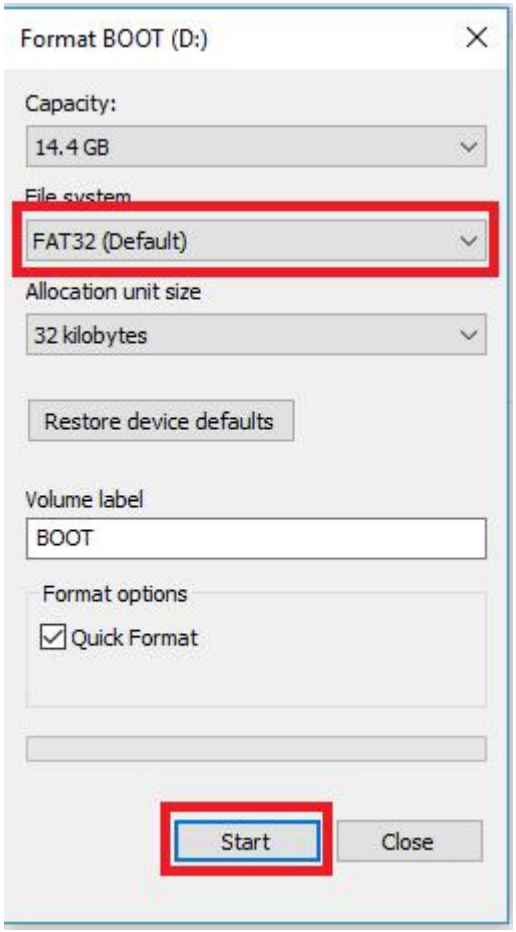
Formatting MicroSD Card：

The first thing we recommend doing is formatting your microSD card. You can use the Windows formatter tool or any other microSD formatter software.

1. Insert the microSD card in your computer. Go to My Computer and right click in the SD card. Select Format as shown in figure below.



2. A new window pops up. Select FAT32, press Start to initialize the formatting process and follow the onscreen instructions.



Installing the ESP32 add-on：

We'll program the ESP32 board using Arduino IDE. So you need the Arduino IDE installed as well as the ESP32 add-on. You can follow one of the next tutorials to install the ESP32 add-on, if you haven't already:

"**Installing ESP32 board package to Arduino IDE**"

Take and Save Photo Sketch

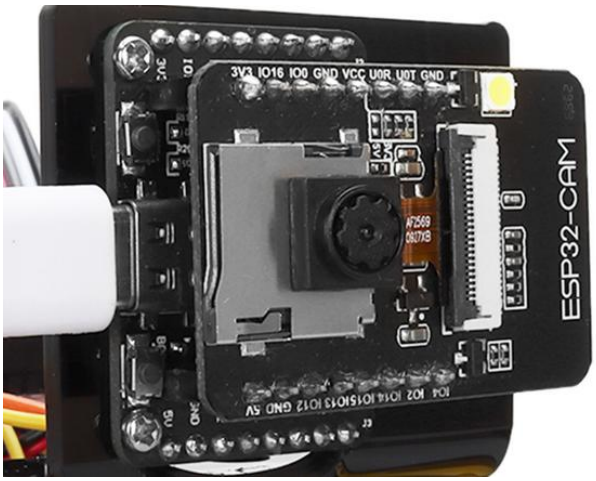
Copy the following code to your Arduino IDE.

```
/*********
***2020.11.20
*********/
#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
// 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);
  //Serial.println();

  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()){
    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");
  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();
  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);

  // 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(2000);
  Serial.println("Going to sleep now");
  delay(2000);
  esp_deep_sleep_start();
  Serial.println("This will never be printed");}

void loop() {
  }
```

ESP32-CAM Upload Code



Note: It is not necessary to manually press the reset switch by default when writing code normally, but it can be manually pressed in special cases such as code cannot be burned.

(Note: if you are using an older version, you can view the wiring instructions if you are using FTDI. If you are using a new version, you can ignore it.)

Many FTDI programmers have a jumper that allows you to select 3.3V or 5V. Make sure the jumper is in the right place to select 5V.

Important: GPIO 0 needs to be connected to GND so that you're able to upload code.

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

To upload the code, follow the next steps:

1) Go to Tools > Board and select AI-Thinker ESP32-CAM.

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

3) Then, click the upload button to upload the code.

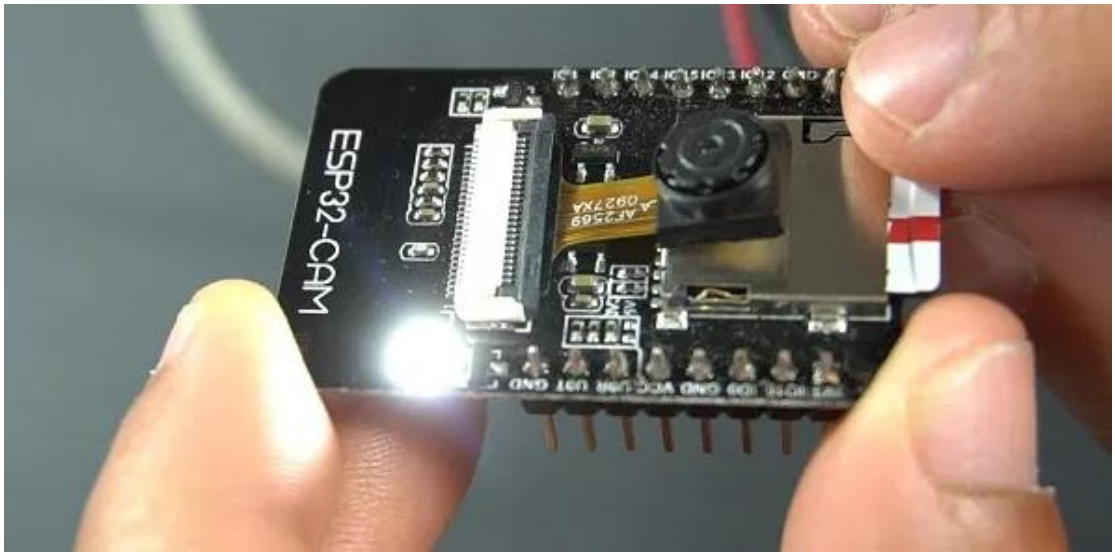
4) When you start to see these dots on the debugging window as shown below, press the ESP32-CAM on-board RST button.

After a few seconds, the code should be successfully uploaded to your board.

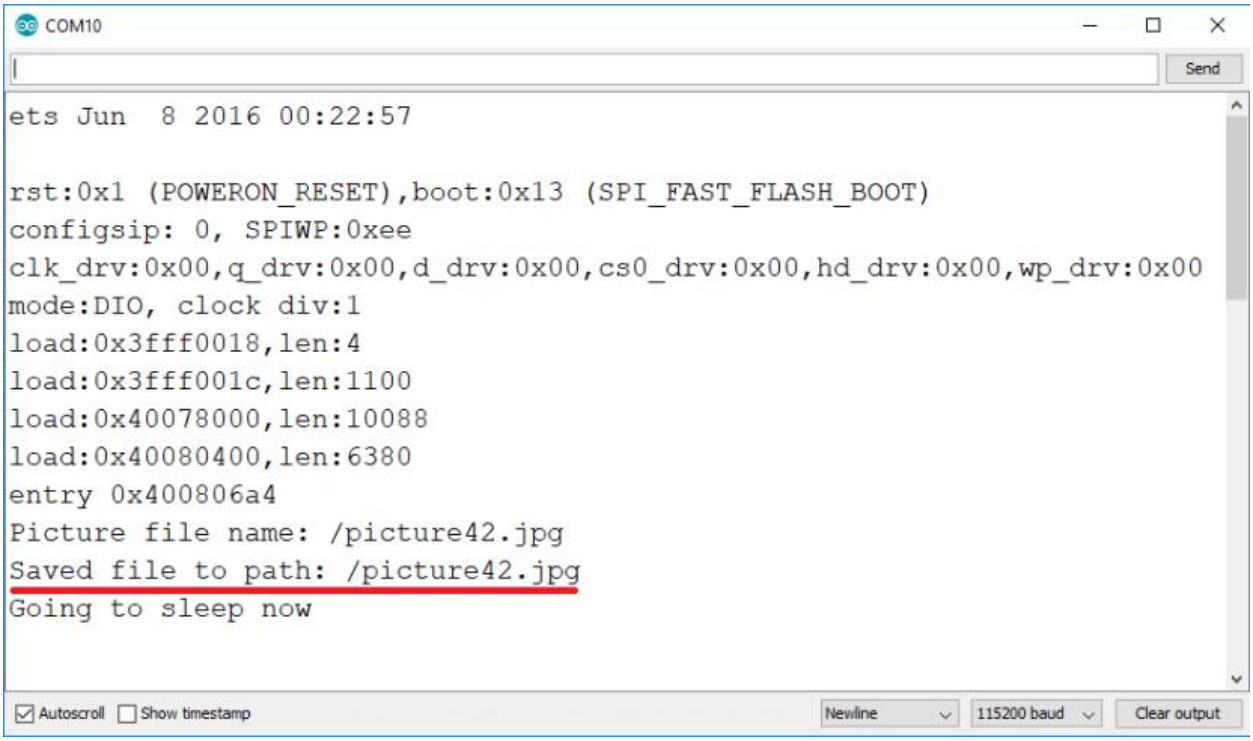
Demonstration：

After uploading the code, remove the jumper that connects GPIO 0 from GND.

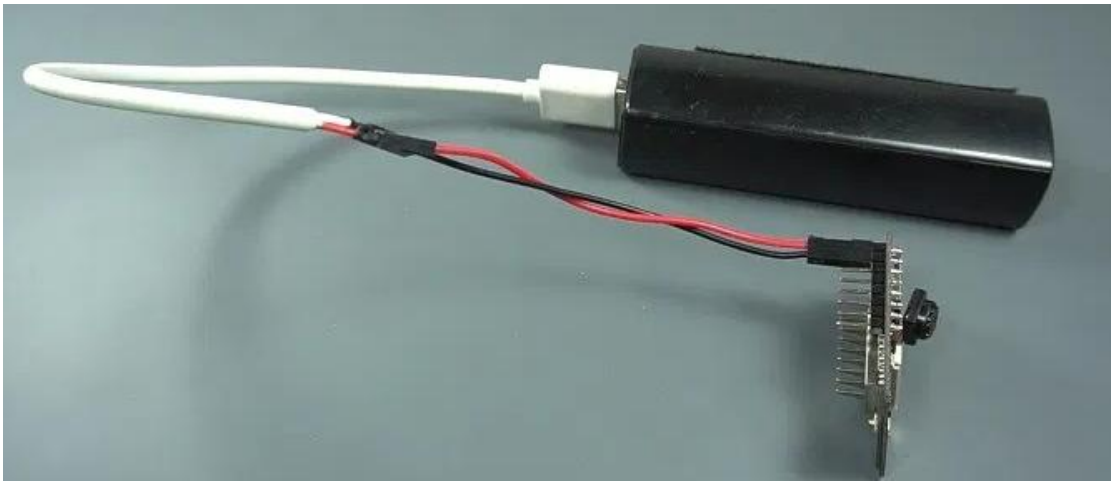
Open the Serial Monitor at a baud rate of 115200. Press the ESP32-CAM reset button. It should initialize and take a photo. When it takes a photo it turns on the flash (GPIO 4).



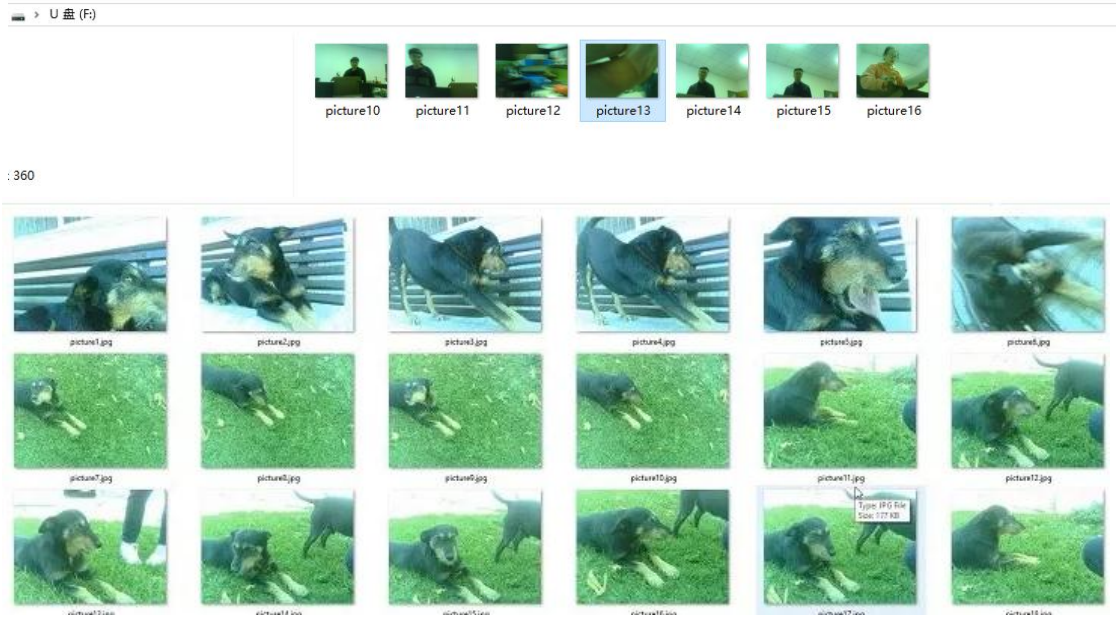
Check the Arduino IDE Serial Monitor window to see if everything is working as expected. As you can see, the picture was successfully saved in the microSD card.



After making sure that everything is working as expected, you can disconnect the ESP32-CAM from the FTDI programmer and power it using an independent power supply.



To see the photos taken, remove the microSD card from the microSD card slot and insert it into your computer. You should have all the photos saved.



The quality of your photo depends on your lighting conditions. Too much light can ruin your photos and dark environments will result in many black pixels.

Project4: ESP32-CAM PIR Motion Detector with Photo Capture (saves to microSD card)

In this project, we're going to make a motion sensor detector with photo capture using an ESP32-CAM. When your PIR sensor detects motion, it wakes up, takes a photo and saves it in the microSD card.

Parts required

For this project, you will need the following parts:

ESP32-CAM with OV2640

MicroSD card

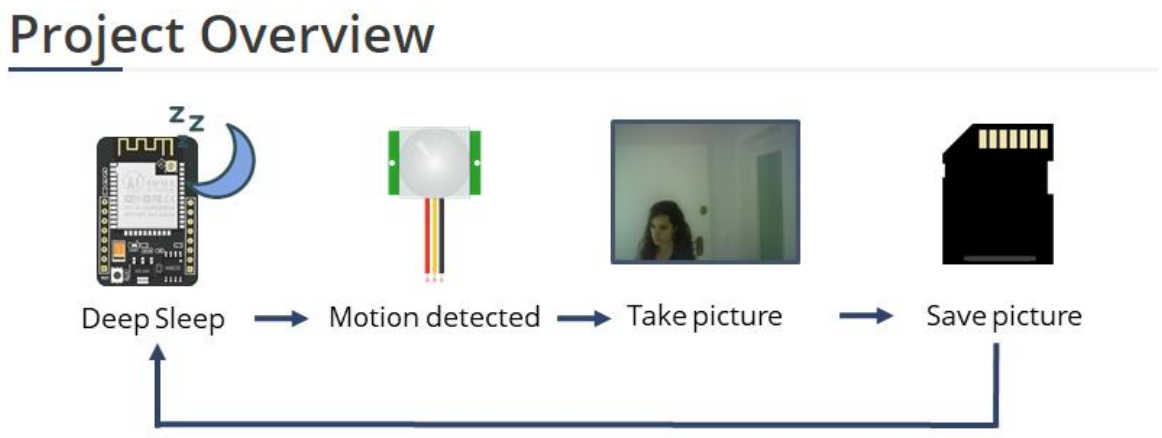
PIR motion sensor

2N3904 transistor

FTDI burner

Female to female jumper

5V power supply for ESP32-CAM or mobile power supply



Here is a quick overview on how the project works.

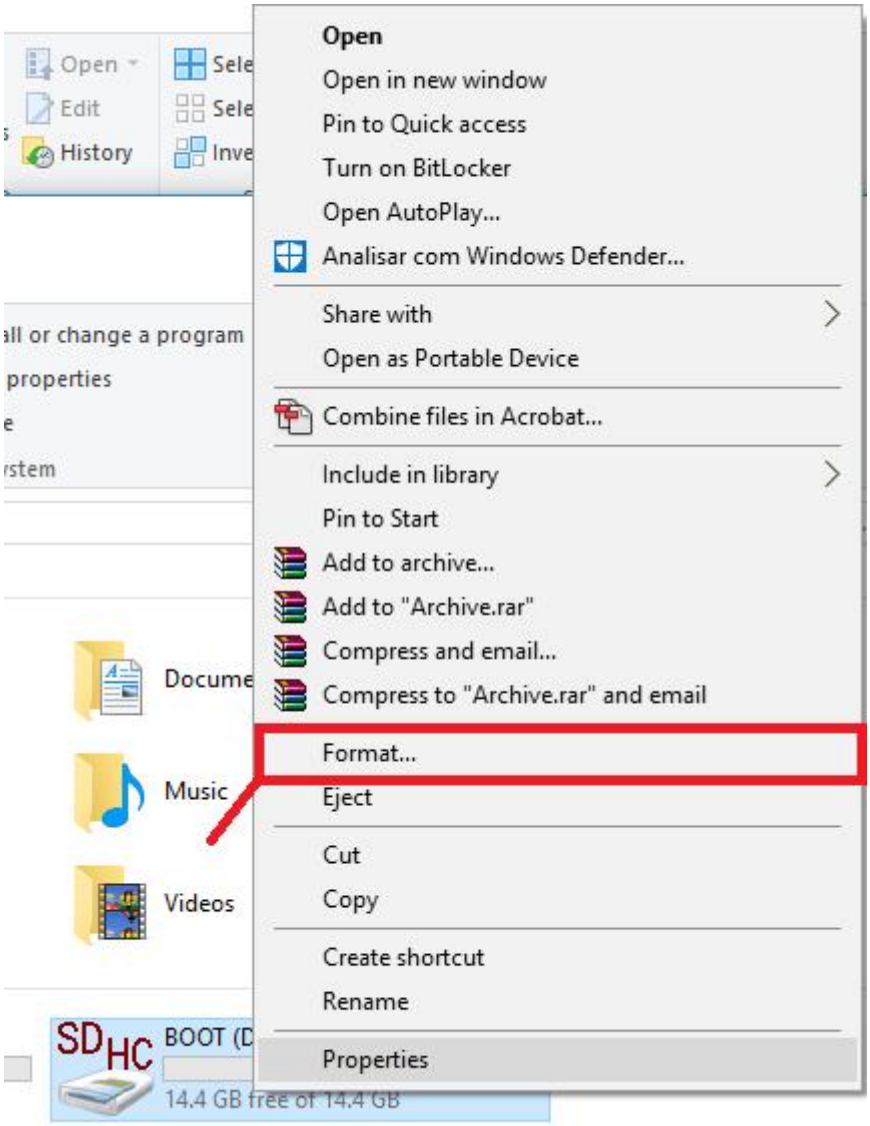
The ESP32-CAM is in deep sleep mode with external wake up enabled.
When motion is detected, the PIR motion sensor sends a signal to wake up the ESP32.

The ESP32-CAM takes a photo and saves it on the microSD card.
It goes back to deep sleep mode until a new signal from the PIR motion sensor is received.

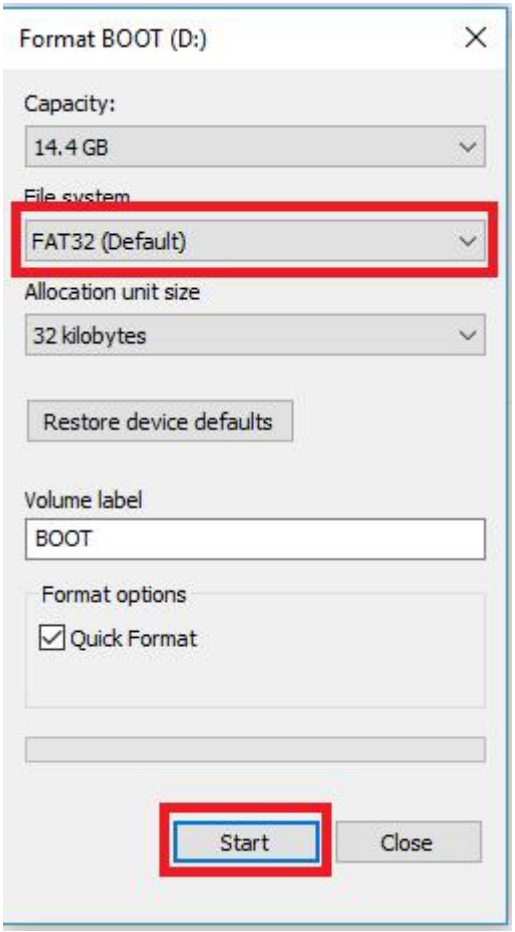
Formatting MicroSD Card

The first thing we recommend doing is formatting your microSD card. You can use the Windows formatter tool or any other microSD formatter software.

1. Insert the microSD card in your computer. Go to My Computer and right click in the SD card. Select Format as shown in figure below.



2. A new window pops up. Select FAT32, press Start to initialize the formatting process and follow the onscreen instructions.



ESP32-CAM Take Photo with PIR Sketch

Copy the following code to your Arduino IDE.

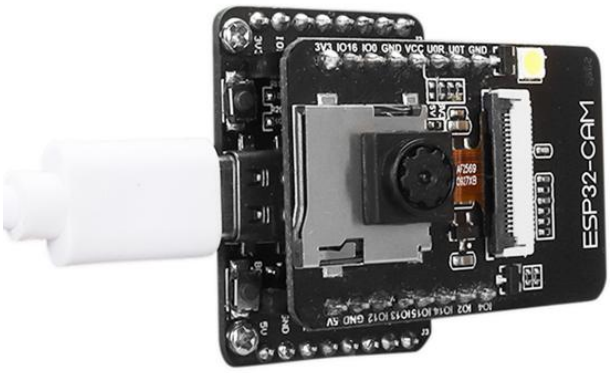
```
/*********
  **2020.11.21
*********/
 #include "esp_camera.h"#include "Arduino.h"#include "FS.h"                // SD Card ESP32#include "SD_MMC.h"                // SD Card ESP32#include "soc/soc.h"                // Disable brownour problems#include "soc/rtc_cntl_reg.h" // Disable brownour problems#include "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();
  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, 0);

  Serial.println("Going to sleep now");
  delay(1000);
  esp_deep_sleep_start();
  Serial.println("This will never be printed");}
void loop() {
}
```

ESP32-CAM Upload Code



Note: It is not necessary to manually press the reset switch by default when writing code normally, but it can be manually pressed in special cases such as code cannot be burned.

(Note: if you are using an older version, you can view the wiring instructions if you are using FTDI. If you are using a new version, you can ignore it.)

Many FTDI programmers have a jumper that allows you to select 3.3V or 5V. Make sure the jumper is in the right place to select 5V.

Important: GPIO 0 needs to be connected to GND so that you're able to upload code.

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

To upload the code, follow the next steps:

1) Go to Tools > Board and select AI-Thinker ESP32-CAM.

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

3) Then, click the upload button to upload the code.

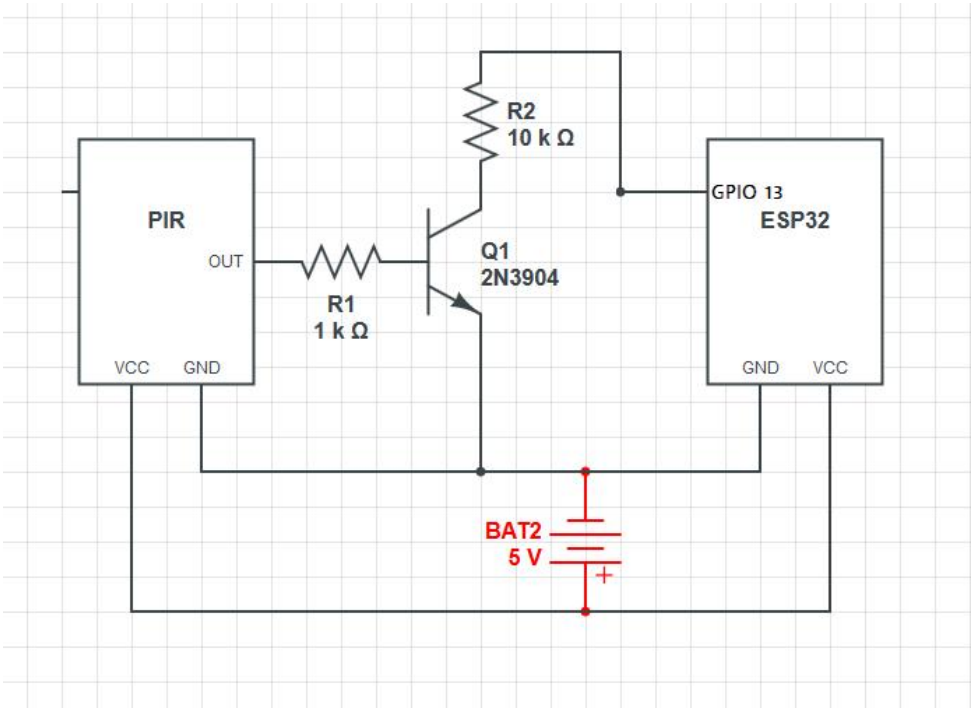
4) When you start to see these dots on the debugging window as shown below, press the ESP32-CAM on-board RST button.

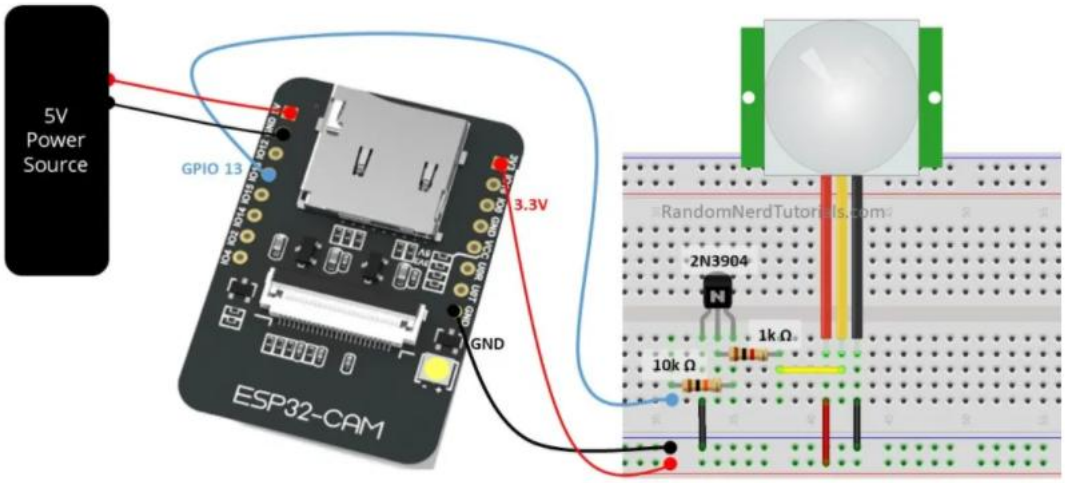
```
esptool.py v2.6-beta1
Serial port COM10
Connecting.....____.....____.....____
```

After a few seconds, the code should be successfully uploaded to your board.

schematic diagram：

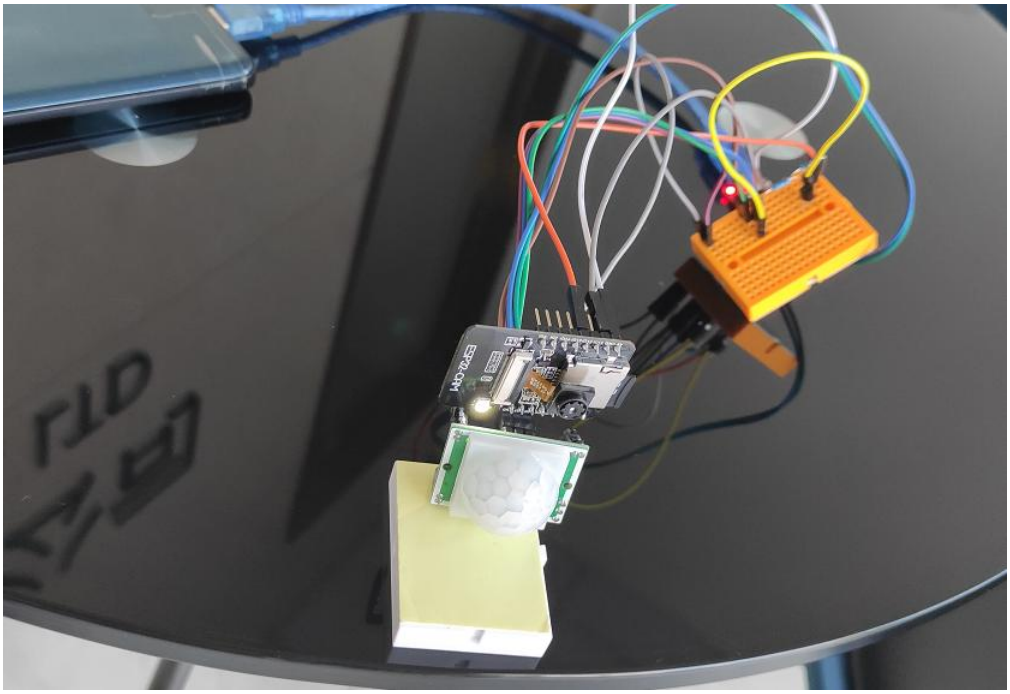
Assemble all parts as shown below.



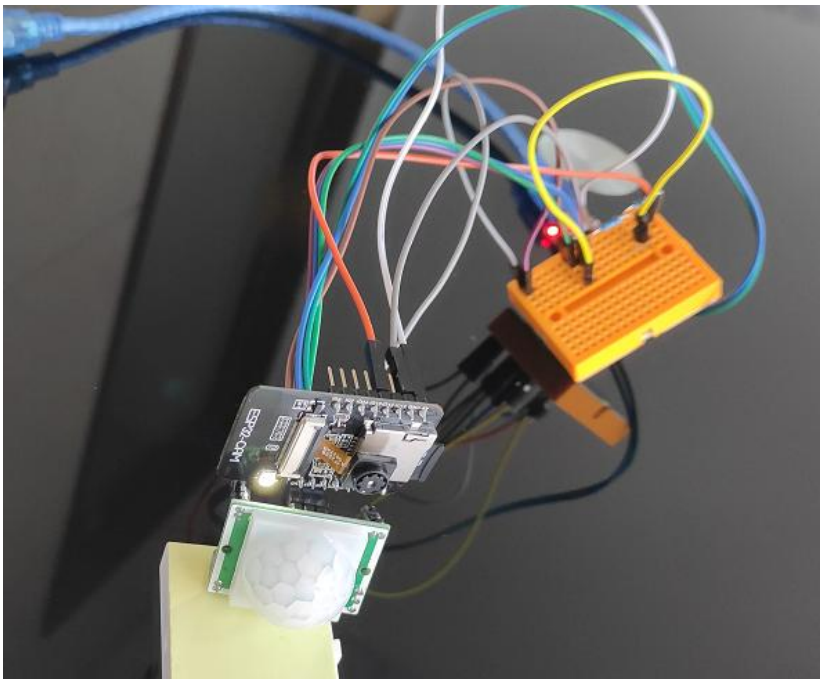


Physical map:

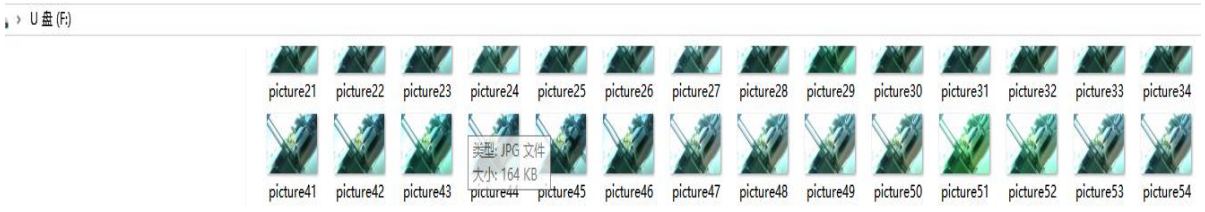
After uploading the code and assembling the circuit, insert a formatted microSD card and power the circuit - for example:



Then, press the Reset (RST) button and it should start working. When it detects movement, it turns on the flash, takes a picture and saves it on a microSD card.



The circuit was tested several times to ensure its normal operation. The microSD card is then inserted into the computer to view the captured photos.



This is a real case:

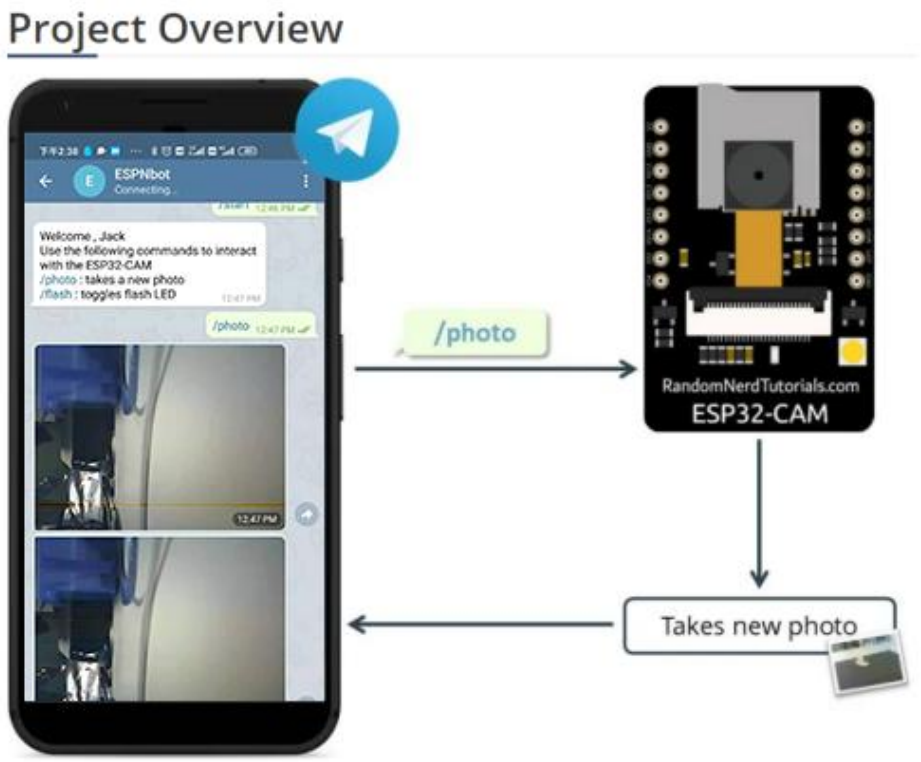
When we install the equipment in the surveillance area, in order to protect the equipment from environmental dust and moisture, you can also design an interesting shell by yourself and print it out through a 3D printer; install this equipment in the surveillance area, as shown in the picture below. The image quality is pretty good.

I wish you a successful experiment!

Project5:Telegram: ESP32-CAM Take and Send Photo (Arduino IDE)

In this tutorial, you'll create a Telegram bot to interact with the ESP32-CAM to request a new photo. You can request a new photo using your Telegram account from anywhere. You just need to have access to the internet in your smartphone.

Note: this project is compatible with any ESP32 Camera Board with the OV2640 camera. You just need to make sure you use the right pinout for th



Here's an overview of the project you'll build:

You'll create a Telegram bot for your ESP32-CAM;

You can start a conversation with the ESP32-CAM bot;

When you send the message /photo to the ESP32-CAM bot, the ESP32-CAM board receives the message, takes a new photo and responds with that photo;

You can send the message /flash to toggle the ESP32-CAM's LED flash;

You can send the /start message to receive a welcome message with the commands to control the board;

The ESP32-CAM will only respond to messages coming from your Telegram account ID.

This is a simple project, but shows how you can use Telegram in your IoT and Home Automation projects. The idea is to apply the concepts learned in your own projects.

Introducing Telegram：

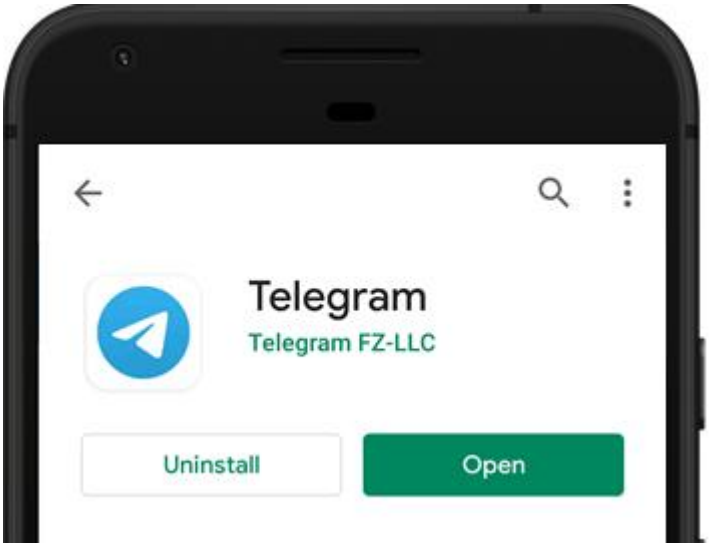
Telegram Messenger is a cloud-based instant messaging and voice over IP service. You can easily install it in your smartphone (Android and iPhone) or computer (PC, Mac and Linux). It's free and without any ads. Telegram allows you to create bots that you can interact with.

"Bots are third-party applications that run inside Telegram. Users can interact with bots by sending them messages, commands and inline requests. You control your bots using HTTPS requests to Telegram Bot API".

The ESP32-CAM will interact with the Telegram bot to receive and handle the messages, and send responses. In this tutorial, you'll learn how to use Telegram to send messages to your bot to request a new photo taken with the ESP32-CAM. You can receive the photo wherever you are (you just need Telegram and access to the internet).

Creating a Telegram Bot

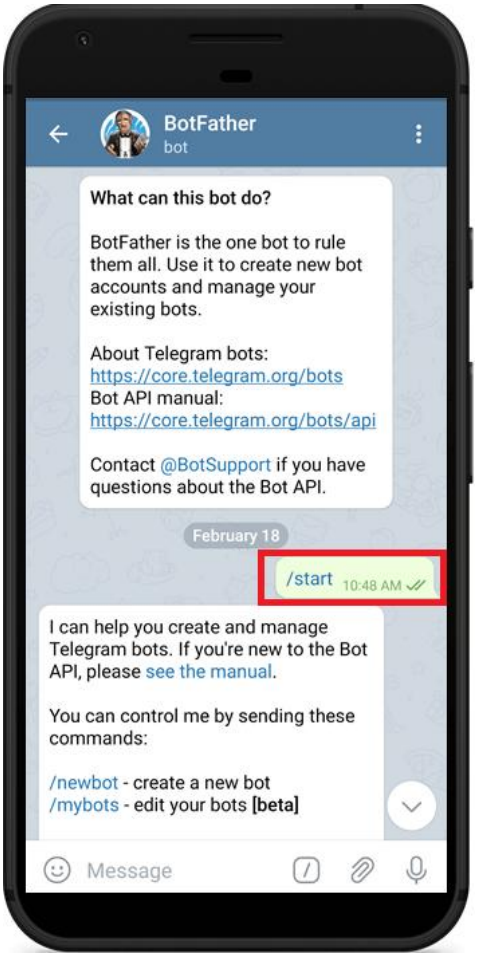
Go to Google Play or App Store, download and install Telegram.



Open Telegram and follow the next steps to create a Telegram Bot. First,

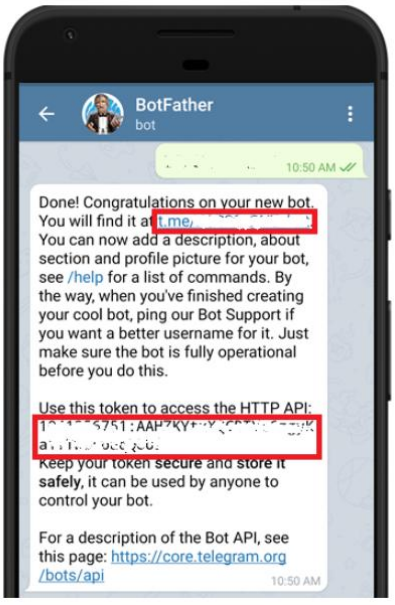
search for “botfather” and click the BotFather as shown below. Or open this link t.me/botfather in your smartphone.

The following window should open and you’ll be prompted to click the start button.



Type /newbot and follow the instructions to create your bot. Give it a name and username.

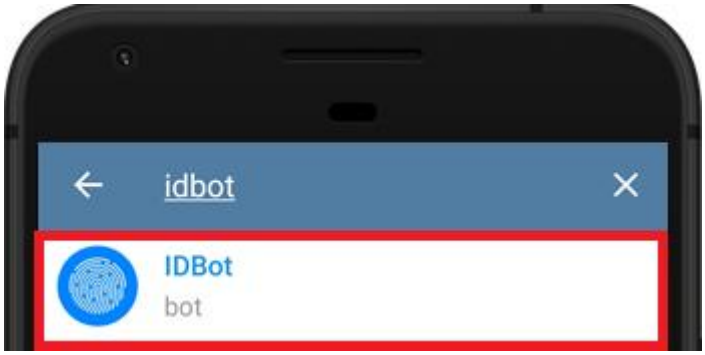
If your bot is successfully created, you'll receive a message with a link to access the bot and the bot token. Save the bot token because you'll need it so that the ESP32 can interact with the bot.



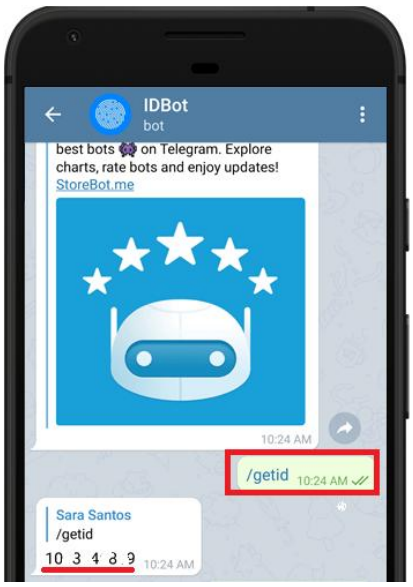
Get Your Telegram User ID:

Anyone that knows your bot username can interact with it. To make sure that we ignore messages that are not from our Telegram account (or any authorized users), you can get your Telegram User ID. Then, when your telegram bot receives a message, the ESP can check whether the sender ID corresponds to your User ID and handle the message or ignore it.

In your Telegram account, search for “IDBot” or open this link t.me/myidbot in your smartphone.



Start a conversation with that bot and type /getid. You will get a reply back with your user ID. Save that user ID, because you’ll need it later in this tutorial.



Preparing Arduino IDE

We'll program the ESP32 board using Arduino IDE, so make sure you have them installed in your Arduino IDE.

"**Installing ESP32 board package to Arduino IDE**"

Universal Telegram Bot Library

To interact with the Telegram bot, we'll use the Universal Telegram Bot Library created by Brian Lough that provides an easy interface for the Telegram Bot API.

Follow the next steps to install the latest release of the library.

1、Open our "Libraries" file; the "Universal Telegram Bot Library" inside

2、Go to Sketch > Include Library > Add.ZIP Library...

3、Add the library you've just downloaded.

Important: don't install the library through the Arduino Library Manager because it might install a deprecated version.

For all the details about the library, take a look at the Universal Arduino Telegram Bot Library GitHub page.

ArduinoJson Library：

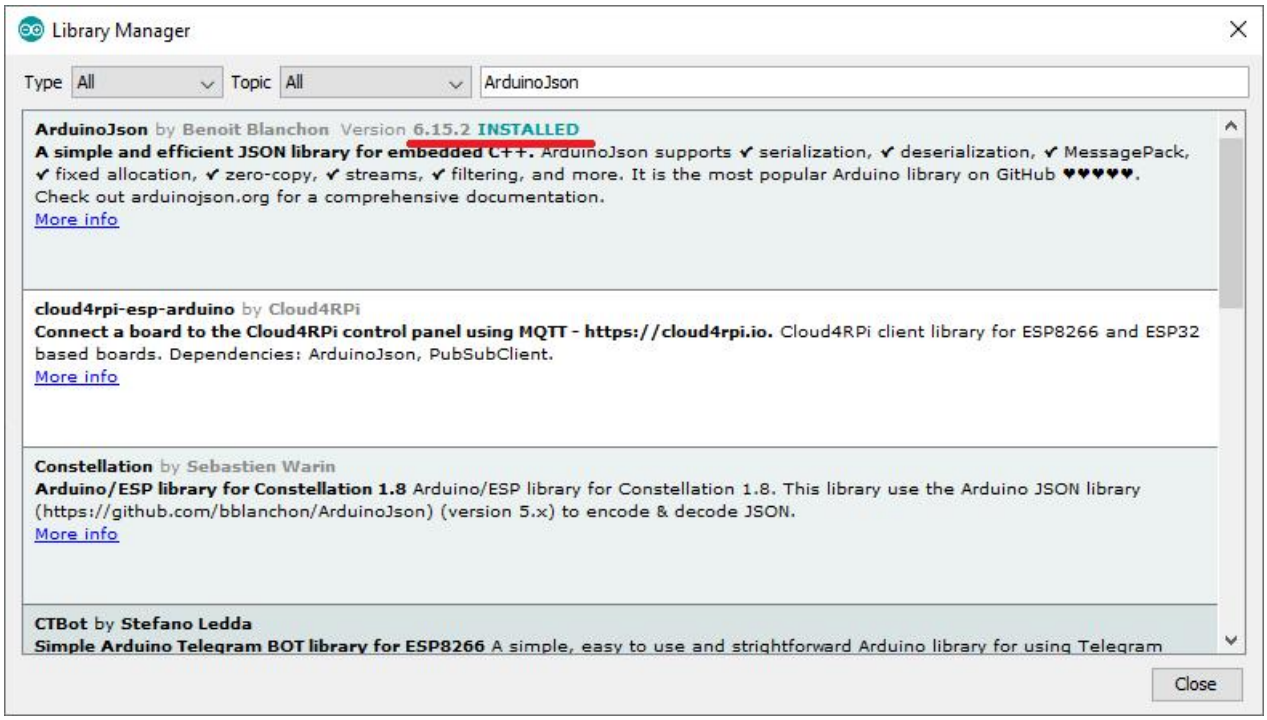
You also have to install the ArduinoJson library. Follow the next steps to install the library.

1、Go to Sketch > Include Library > Manage Libraries.

2、Search for “ArduinoJson”.

3、Install the library.

We’re using ArduinoJson library version 6.5.12.



Code ：

Copy the following code the Arduino IDE. To make this sketch work for you, you need to insert your network credentials (SSID and password), the Telegram Bot token and your Telegram user ID. Additionally, check the pin assignment for the camera board that you’re using.

```
#include <Arduino.h>
#include <WiFi.h>
#include<WiFiClientSecure.h>
#include "soc/soc.h"#include "soc/rtc_cntl_reg.h"#include "esp_camera.h"#include <UniversalTelegramBot.h>#include <ArduinoJson.h>
const char* ssid = "REPLACE_WITH_YOUR_SSID";const char* password = "REPLACE_WITH_YOUR_PASSWORD";
// Initialize Telegram BOT
String BOTtoken = "XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"; // your Bot Token (Get from Botfather)
// Use @myidbot to find out the chat ID of an individual or a group// Also note that you need to click "start" on a bot before it can// message you
String CHAT_ID = "XXXXXXXXXX";

bool sendPhoto = false;

WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);
#define FLASH_LED_PIN 4
bool flashState = LOW;
//Checks for new messages every 1 second.int botRequestDelay = 1000;unsigned long lastTimeBotRan;
//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

void configInitCamera(){
  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;

  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;    //0-63 lower number means higher quality
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;    //0-63 lower number means higher quality
    config.fb_count = 1;
  }

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    delay(1000);
    ESP.restart();
  }

  // Drop down frame size for higher initial frame rate
  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s, FRAMESIZE_CIF);
 // UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA}
void handleNewMessages(int numNewMessages) {
  Serial.print("Handle New Messages: ");
  Serial.println(numNewMessages);

  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID){
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
```

```
      }

      // Print the received message
      String text = bot.messages[i].text;
      Serial.println(text);

      String from_name = bot.messages[i].from_name;
      if (text == "/start") {
        String welcome = "Welcome , " + from_name + "\n";
        welcome += "Use the following commands to interact with the ESP32-CAM \n";
        welcome += "/photo : takes a new photo\n";
        welcome += "/flash : toggles flash LED \n";
        bot.sendMessage(CHAT_ID, welcome, "");
      }
      if (text == "/flash") {
        flashState = !flashState;
        digitalWrite(FLASH_LED_PIN, flashState);
        Serial.println("Change flash LED state");
      }
      if (text == "/photo") {
        sendPhoto = true;
        Serial.println("New photo request");
      }
    }}

String sendPhotoTelegram() {
  const char* myDomain = "api.telegram.org";
  String getAll = "";
  String getBody = "";

  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    delay(1000);
    ESP.restart();
    return "Camera capture failed";
  }

  Serial.println("Connect to " + String(myDomain));


  if (clientTCP.connect(myDomain, 443)) {
```

```
    Serial.println("Connection successful");

    String head = "--RandomNerdTutorials\r\nContent-Disposition: form-data; name=\"chat_id\"; \r\n\r\n" + CHAT_ID + "\r\n--RandomNerdTutorials\r\nContent-Disposition: form-data; name=\"photo\"; filename=\"esp32-cam.jpg\"\r\nContent-Type: image/jpeg\r\n\r\n";
    String tail = "\r\n--RandomNerdTutorials--\r\n";

    uint16_t imageLen = fb->len;
    uint16_t extraLen = head.length() + tail.length();
    uint16_t totalLen = imageLen + extraLen;

    clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto HTTP/1.1");
    clientTCP.println("Host: " + String(myDomain));
    clientTCP.println("Content-Length: " + String(totalLen));
    clientTCP.println("Content-Type: multipart/form-data; boundary=RandomNerdTutorials");
    clientTCP.println();
    clientTCP.print(head);

    uint8_t *fbBuf = fb->buf;
    size_t fbLen = fb->len;
    for (size_t n=0;n<fbLen;n=n+1024) {
      if (n+1024<fbLen) {
        clientTCP.write(fbBuf, 1024);
        fbBuf += 1024;
      }
      else if (fbLen%1024>0) {
        size_t remainder = fbLen%1024;
        clientTCP.write(fbBuf, remainder);
      }
    }

    clientTCP.print(tail);

    esp_camera_fb_return(fb);

    int waitTime = 10000;      // timeout 10 seconds
    long startTimer = millis();
    boolean state = false;

    while ((startTimer + waitTime) > millis())
    {
      Serial.print(".");
```

```
        delay(100);
        while (clientTCP.available())
        {
              char c = clientTCP.read();
              if (c == '\n')
              {
                 if (getAll.length()==0) state=true;
                 getAll = "";
              }
              else if (c != '\r')
                 getAll += String(c);
              if (state==true) getBody += String(c);
              startTimer = millis();
         }
         if (getBody.length()>0) break;
      }
      clientTCP.stop();
      Serial.println(getBody);
   }
   else {
      getBody="Connected to api.telegram.org failed.";
      Serial.println("Connected to api.telegram.org failed.");
   }
   return getBody;}
void setup(){
   WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
   // Init Serial Monitor
   Serial.begin(115200);

   // Set LED Flash as output
   pinMode(FLASH_LED_PIN, OUTPUT);
   digitalWrite(FLASH_LED_PIN, flashState);

   // Config and init the camera
   configInitCamera();

   // Connect to Wi-Fi
   WiFi.mode(WIFI_STA);
   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(ssid);
   WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED) {
      Serial.print(".");
```

```
    delay(500);
  }
  Serial.println();
  Serial.print("ESP32-CAM IP Address: ");
  Serial.println(WiFi.localIP()); }
void loop() {
  if (sendPhoto) {
    Serial.println("Preparing photo");
    sendPhotoTelegram();
    sendPhoto = false;
  }
  if (millis() > lastTimeBotRan + botRequestDelay)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while (numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }}
```

How the Code Works

This section explains how the code works. Continue reading to learn how the code works or skip to the Demonstration section.

Importing Libraries

Start by importing the required libraries.

```
#include<Arduino.h>

#include<WiFi.h>

#include<WiFiClientSecure.h>

#include"soc/soc.h"

#include"soc/rtc_cntl_reg.h"

#include"esp_camera.h"

#include <UniversalTelegramBot.h>
```

#include <ArduinoJson.h>

Network Credentials

Insert your network credentials in the following variables.

const char* ssid = "REPLACE_WITH_YOUR_SSID";

const char* password = "REPLACE_WITH_YOUR_PASSWORD";

Telegram User ID

Insert your chat ID. The one you've got from the IDBot.

String CHAT_ID = "XXXXXXXXXX";

Telegram Bot Token

Insert your Telegram Bot token you've got from Botfather on the BOTtoken variable.

String BOTtoken="XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

The sendPhoto boolean variable indicates whether it is time to send a new photo to your telegram account. By default, it is set to false.

bool sendPhoto = false;

Create a new WiFi client with WiFiClientSecure.

WiFiClientSecure clientTCP;

Create a bot with the token and client defined earlier.

UniversalTelegramBot bot(BOTtoken, clientTCP);

Create a variable to hold the flash LED pin (FLASH_LED_PIN). In the ESP32-CAM AI Thinker, the flash is connected to GPIO 4. By default, set it to LOW.

define FLASH_LED_PIN 4

bool flashState = LOW;

The botRequestDelay and lasTimeBotRan variables are used to check for new Telegram messages every x number of seconds. In this case, the code will check for new messages every second (1000 milliseconds). You can change that delay time in the botRequestDelay variable.

int botRequestDelay = 1000;

unsigned long lastTimeBotRan;

Demonstration ：

Upload the code to your ESP32-CAM board. Don’t forget to go to Tools > Board and select the board you’re using. Go to Tools > Port and select the COM port your board is connected to.

After uploading the code, press the ESP32-CAM on-board RST button so that it starts running the code. Then, you can open the Serial Monitor to check what’s happening in the background.

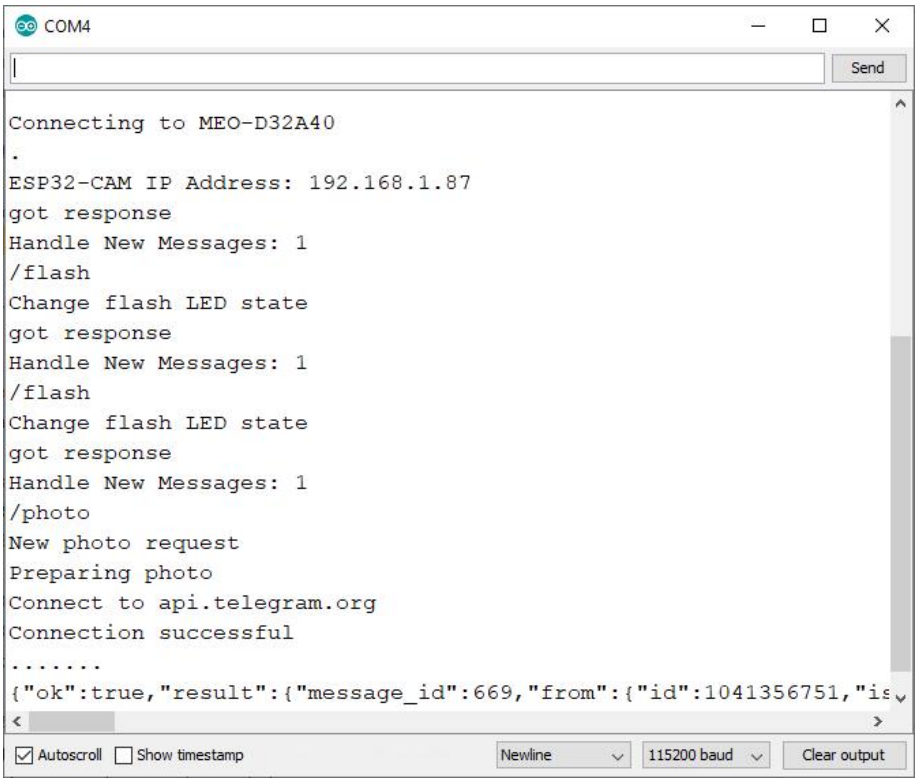
Go to your Telegram account and open a conversation with your bot. Send the following commands and see the bot responding:

/start shows the welcome message with the valid commands;

/flash inverts the state of the LED flash;

/photo takes a new photo and sends it to your Telegram account.

At the same time, on the Serial Monitor, you should see that the ESP32-CAM is receiving the messages.



In this tutorial you've learned how to send a photo from the ESP32-CAM to your Telegram account. As long as you have access to the internet in your smartphone, you can request a new photo no matter where you are. This is great to monitor your ESP32-CAM from anywhere in the world.