

Object Detection using ESP 32 CAM

Ninad Mehendale

Abstract:

This paper covers ESP32 CAM Based Object Detection & Identification with OpenCV. OpenCV is an open-sourced image processing library that is very widely used not just in industry but also in the field of research and development. Here for object detection cvlib Library [1] is used. The library uses a pre-trained AI model on the COCO dataset to detect objects. The name of the pre-trained model is YOLOv3.


For hardware, the ESP32 Camera Module was used which can be programmed through FTDI Module. It is required to set up the Arduino IDE for the ESP32 Camera Module. It is important to upload the firmware and then work on the object detection & identification part. The script for object detection is written in the python programming language, thus people who want to use this code will also have to install Python and its required Libraries. This project makes use of OpenCV for Object Detection & Identification.



Figure 1: Concept diagram

Bill of Materials

The following is the list of Bill of Materials for building an ESP32 CAM-Based Object Detection & Identification System. The ESP32 CAM when combined with other hardware & firmware track and identify the object. One can purchase all these components from online sites such as Amazon.

S.N.	COMPONENTS	DESCRIPTION	QUANTITY	
1	ESP32-CAM Board	AI-Thinker ESP32 Camera Module	1	https://amzn.to/2RjsVnm
2	FTDI Module	USB-to-TTL Converter Module	1	https://amzn.to/3wWjQjD
3	USB Cable	5V Mini-USB Data Cable	1	https://amzn.to/3zag9cb
4	Jumper Wires	Female to Female Connectors	10	https://amzn.to/3z3wak3

ESP32 CAM Module

The ESP32 Based Camera Module was developed by AI-Thinker. The controller is based on a 32-bit CPU & has a combined Wi-Fi + Bluetooth/BLE Chip. It has a built-in 520 KB SRAM with an external 4M PSRAM. Its GPIO Pins have support like UART, SPI, I2C, PWM, ADC, and DAC. The module combines with the OV2640 Camera Module which has the highest Camera Resolution up to 1600×1200 . The camera connects to the ESP32 CAM Board using a 24 pins gold plated connector. The board supports an SD Card of up to 4GB. The SD Card stores capture images.

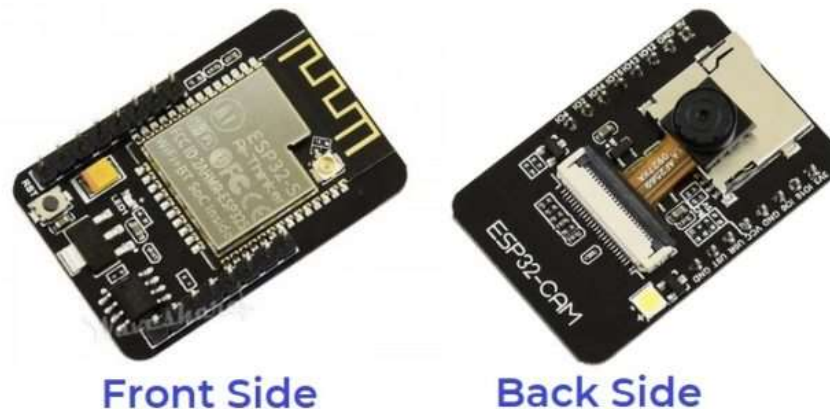


Figure 2: ESP 32 CAM module

ESP32-CAM FTDI Connection: The board doesn't have a programmer chip. So, to program this board, you can use any type of USB-to-TTL Module. There are so many FTDI Modules available based on CP2102 or CP2104 Chip or any other chip.

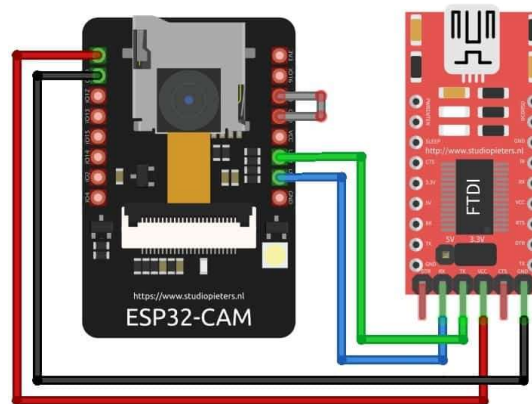


Figure 3: connection between FTDI Module and the ESP32 CAM module.

As shown in figure 3 Connect the 5V & GND Pin of ESP32 to the 5V & GND of the FTDI Module. Similarly, connect the Rx to U0T and Tx to U0R Pin. And the most important thing, you need to short the IO0 and GND Pin together. This is to put the device in programming mode. Once programming is done you can remove it.

Installing ESP32CAM Library: Here we will not use the general ESP webserver example but rather another streaming process. Therefore, we need to add another ESPCAM library. The esp32cam library provides an object-oriented API to use the OV2640 camera on the ESP32 microcontroller. It is a wrapper of the esp32-camera library.

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC
U0R	TX
U0T	RX
GPI00	GND

Table 2: Connection between FTDI Module and the ESP32 CAM module.

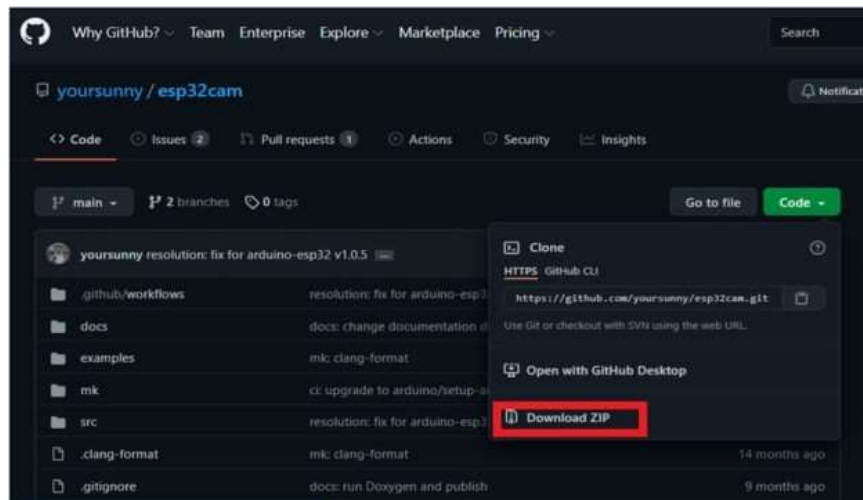


Figure 4: Screen shot of Github Link [2] where user can download the zip library as in the image

Once downloaded add zip library as shown in figure 4 to Arduino Library Folder.

To do so follow the following steps:

Open Arduino -> Sketch -> Include Library -> Add .ZIP Library -> Navigate to downloaded zip file -> add

Source Code/Program for ESP32 CAM Module:

```
#include <WebServer.h>
```

```
#include <WiFi.h>
```

```
#include <esp32cam.h>
```

```
const char* WIFI_SSID = "ssid e.g. NRL";
```

```
const char* WIFI_PASS = "password e.g. 9820805405";
```

```
WebServer server(80);
```

```
static auto loRes = esp32cam::Resolution::find(320, 240);
```

```
static auto midRes = esp32cam::Resolution::find(350, 530);
```

```
static auto hiRes = esp32cam::Resolution::find(800, 600);
```

```
void serveJpg()
```

```
{
```

```
    auto frame = esp32cam::capture();
```

```
    if (frame == nullptr) {
```

```
        Serial.println("CAPTURE FAIL");
```

```
        server.send(503, "", "");
```

```
        return;
```

```
    }
```

```
    Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
```

```
        static_cast<int>(frame->size()));
```

```
    server.setContentLength(frame->size());
```

```
    server.send(200, "image/jpeg");
```

```
    WiFiClient client = server.client();
```

```
    frame->writeTo(client);
```

```
}
```

```
void handleJpgLo()
```

```
{
```

```
    if (!esp32cam::Camera.changeResolution(loRes)) {
```

```
        Serial.println("SET-LO-RES FAIL");
```

```
    }
```

```
    serveJpg();
```

```
}
```

```
void handleJpgHi()
```

```

{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}

void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}

void setup(){
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
  }
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
}

```

```

WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
}
Serial.print("http://");
Serial.println(WiFi.localIP());
Serial.println(" /cam-lo.jpg");
Serial.println(" /cam-hi.jpg");
Serial.println(" /cam-mid.jpg");

server.on("/cam-lo.jpg", handleJpgLo);
server.on("/cam-hi.jpg", handleJpgHi);
server.on("/cam-mid.jpg", handleJpgMid);

server.begin();
}

void loop()
{
  server.handleClient();
}

```

Here is a source code for Object Detection & Identification with ESP32 Camera & OpenCV. Copy the code and paste it in the Arduino IDE.

Before Uploading the code, you have to make a small change to the code. Change the SSID and password variable and in accordance with your Wi-Fi network. Now compile and upload it to the ESP32 CAM Board. But during uploading, you have to follow few steps every time.

- Make sure the IO0 pin is shorted with the ground when you have pressed the upload button.
- If you see the dots and dashes while uploading press the reset button immediately
- Once the code is uploaded, remove the IO1 pin shorting with Ground and press the reset button once again.
- If the output in the Serial monitor is still not there then press the reset button again.
- Now you can see a similar output as in figure 5.

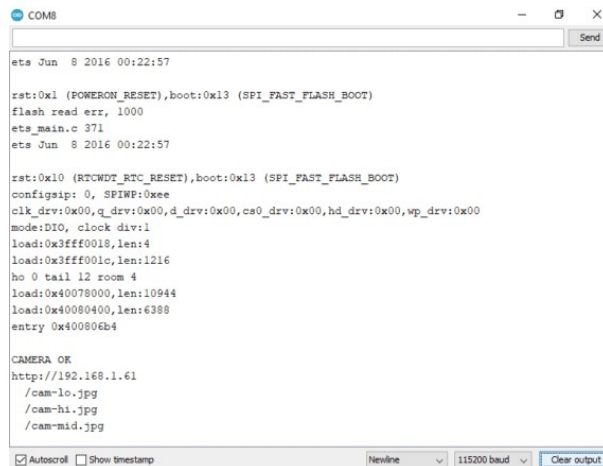
A screenshot of a terminal window titled 'COMS'. The window shows the boot process of an ESP32 microcontroller. It starts with a reset, followed by flash read error reporting and main program execution. The boot logs include hardware configuration details like SPI and I2C drivers, and a list of loaded partitions. At the bottom, it shows 'CAMERA OK' and a list of camera image files: /cam-lo.jpg, /cam-hi.jpg, and /cam-mid.jpg. The terminal interface includes a 'Send' button at the top right and checkboxes for 'Autoscroll' and 'Show timestamp' at the bottom left, along with a baud rate setting of 115200.

Figure 5: Output screen shot

As per figure 5, copy the IP address visible, the user will be using it to edit the URL in python code

Python Library Installation: For the live stream of video to be visible on our computer we need to write a Python script that will enable us to retrieve the frames of the video. The first step is to install Python. Go to python.org and download Python. Once download, install Python. Then Go to the command prompt and install NumPy, OpenCV and cvlib libraries [1].

type: **pip install numpy** and press enter. After the installation is done.

type: **pip install opencv-python** and press enter.

type: **pip install cvlib** and press enter, close the command prompt.

In our python code we have used urllib.request to retrieve the frames from the URL and the library for image processing is OpenCV. For Object detection, we have used the Cvlib library that uses an AI model for detecting objects. Since the whole process requires a good amount of processing power, thus we have used multiprocessing which utilizes multiple cores of our CPU.

Python Code for ESP32 CAM Object Detection/Identification: Now open Idle code editor or any other python code editor. Copy and paste the code from below and do the replacements as mentioned below.

```
import cv2
import matplotlib.pyplot as plt
import cvlib as cv
import urllib.request
import numpy as np
from cvlib.object_detection import draw_bbox
import concurrent.futures

url='http://192.168.10.162/cam-hi.jpg'
im=None

def run1():
    cv2.namedWindow("live transmission", cv2.WINDOW_AUTOSIZE)
    while True:
        img_resp=urllib.request.urlopen(url)
        imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
        im = cv2.imdecode(imgnp,-1)
```

```

cv2.imshow('live transmission',im)
key=cv2.waitKey(5)
if key==ord('q'):
    break

cv2.destroyAllWindows()

def run2():
    cv2.namedWindow("detection", cv2.WINDOW_AUTOSIZE)
    while True:
        img_resp=urllib.request.urlopen(url)
        imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
        im = cv2.imdecode(imgnp,-1)

        bbox, label, conf = cv.detect_common_objects(im)
        im = draw_bbox(im, bbox, label, conf)

        cv2.imshow('detection',im)
        key=cv2.waitKey(5)
        if key==ord('q'):
            break

    cv2.destroyAllWindows()

if __name__ == '__main__':
    print("started")
    with concurrent.futures.ProcessPoolExecutor() as executor:
        f1= executor.submit(run1)
        f2= executor.submit(run2)

```

Here we have to replace the IP address with the IP on Arduino Serial Monitor. For the first time, it will install a few files if they are not existing. Once we have done that, we can see two windows named live transmission and detected is visible. Now in the detected window, one can view different detected objects as around them different coloured boxes are visible. The video for this application is also available on <https://youtu.be/A1SPJSVra9I> [3]

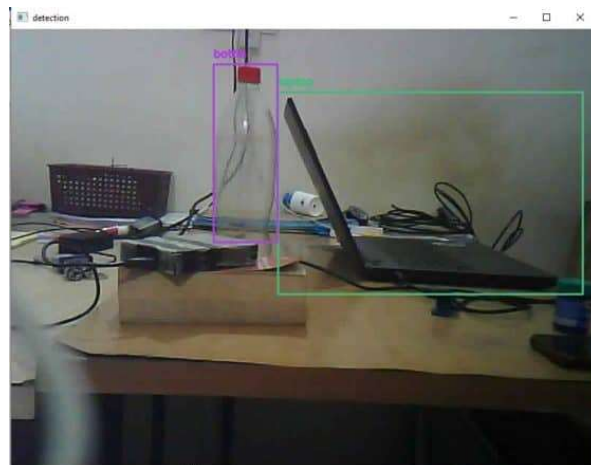


Figure 5: Result, where objects are marked with bounding box.

Conclusions: Object detection is having uses in almost all sorts of industries. It is used for tracking objects, people counting, automated CCTV surveillance, vehicle detection, etc. These are just some basic examples but in reality, the potential is tremendous.

References

- [1] <https://www.cvlb.net/>
- [2] <https://github.com/yoursunny/esp32cam>
- [3] <https://how2electronics.com/esp32-cam-based-object-detection-identification-with-opencv/>