

TensorFlow

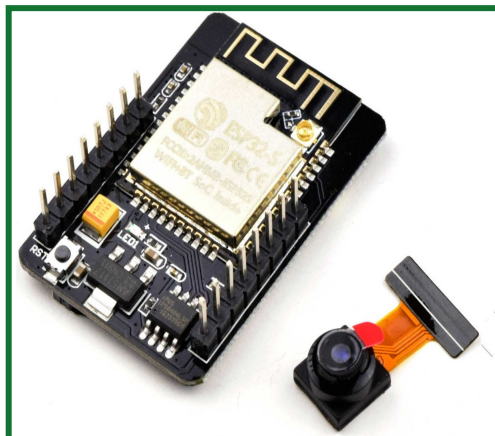
TensorFlow allows developers to create data flow diagrams, structures that describe how data moves through diagrams or a series of processing nodes. Each node in the diagram represents a mathematical operation, and each connection or edge between nodes is a multidimensional or tensile data array. TensorFlow provides all of this process through the Python language. It is easy to learn and work with Python and provides convenient ways to express high-level abstractions together. But real math is not done in Python. The libraries available for TensorFlow are written in high-performance C binary. Python only directs traffic between components and provides high-level programming abstracts to connect them.

Pre-trained models

In machine learning, there is a topic called model, which is actually the objects and things that we want to recognize, for example, recognizing the human face, so for this we have to teach our machine that the appearance of a human, female or male, Small or large, old or young. For this we have to introduce different images to our car to increase the ability to detect our car. This result of this process is called modelling, which is the hallmark of our machine, and then all of these models must be labelled so that the result is visually understandable to the user.

ESP32-CAM

- The board is powered by an ESP32-S SoC from Espressif, a powerful, programmable MCU with out-of-the-box WIFI and Bluetooth.
- It's the cheapest (around \$7) ESP32 dev board that offers an onboard camera module, MicroSD card support, and 4MB PSRAM at the same time.
- Adding an external wifi antenna for signal boosting requires extra soldering work.
- The board does not have a conventional USB port, you will have to use either an FTDI programmer, or an add-on HAT, or an Arduino UNO along with the Arduino IDE/ESP-IDF DEV tools to upload codes to it.
- Being a low-cost enough form factor popular for many applications.



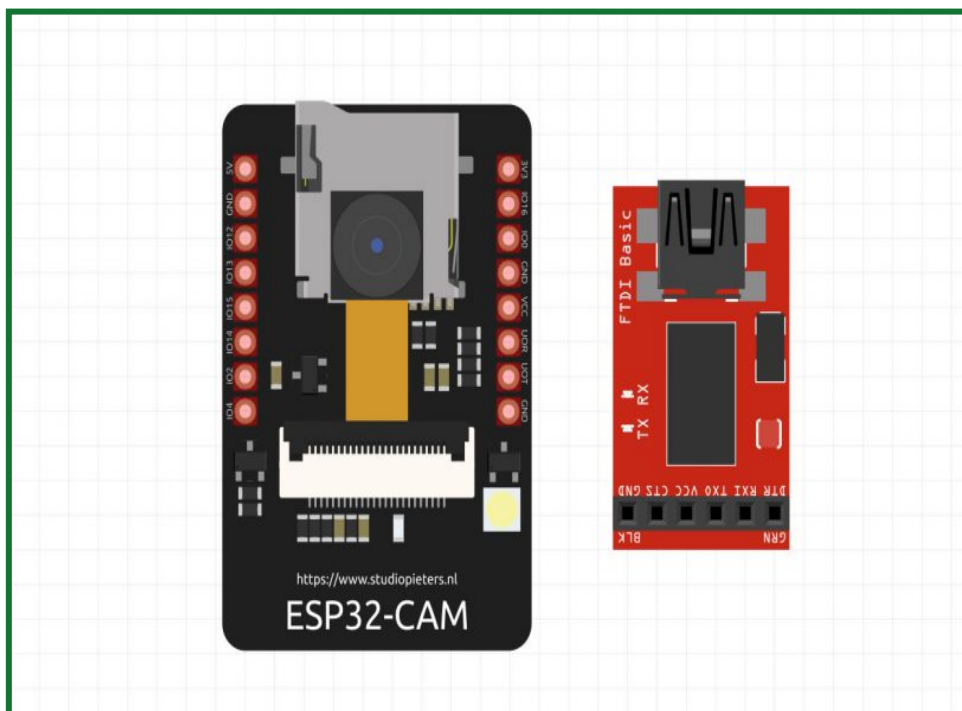
board in a small has made it extremely IoT and machine vision

- The outdated spec sheet and many tutorial pages say that the ESP32-CAM only supports two camera modules (OV2640 & OV7670), while in fact you can use many cameras with it, simply scroll down for more details.

How to detect objects with the camera

In this project, by streaming images using the ESP32-CAM board and receiving and displaying them in the browser, we will also use Tensorflow.JS to process images using the default models applied. As soon as the image is received by the web server running in the browser, it will anticipate and analyze the items in the image. In this project, only the default objects are recognizable. As you may know, Tensorflow has several pre-trained models that we can use to easily start learning the machine. COCO-SSD is an ML model used to localize and identify objects in an image. The same model is used in this tutorial.

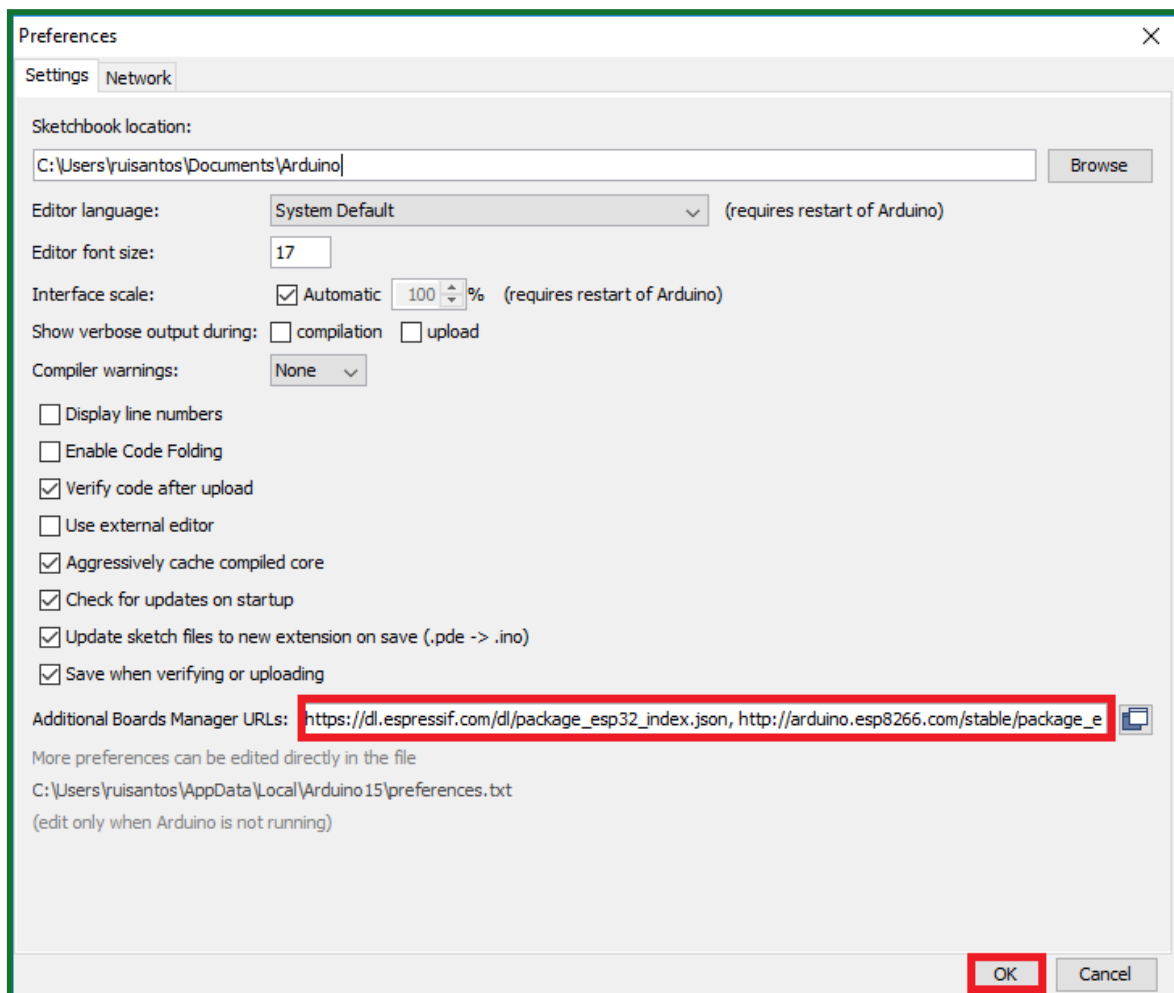
ESP32-CAM module FT232 converter module for program



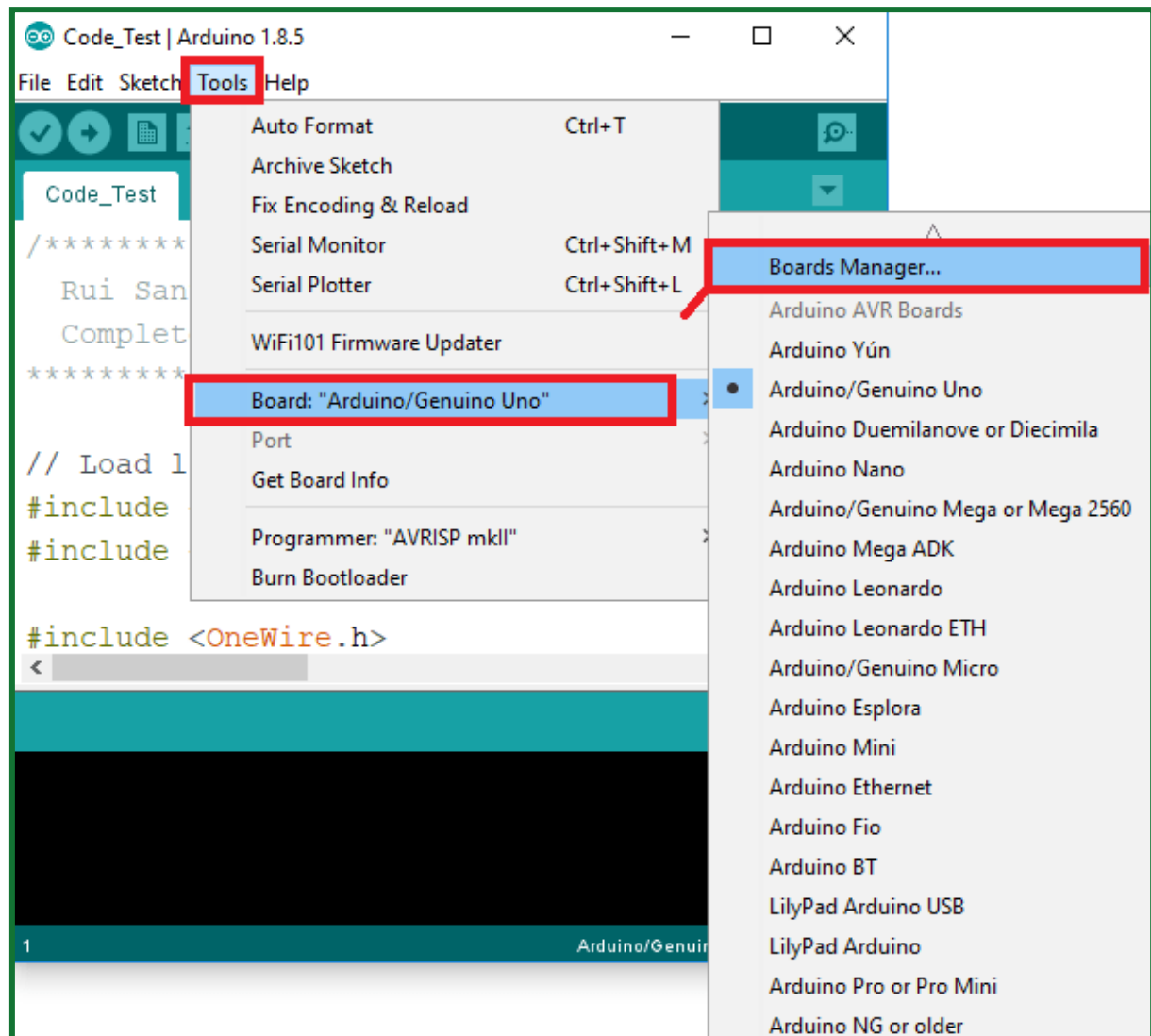
Install ESP32 plugin on Arduino

To get started, the library for this module must be installed in the Arduino IDE software. First, the following link must be copied and pasted to **File> Preferences**.

https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json

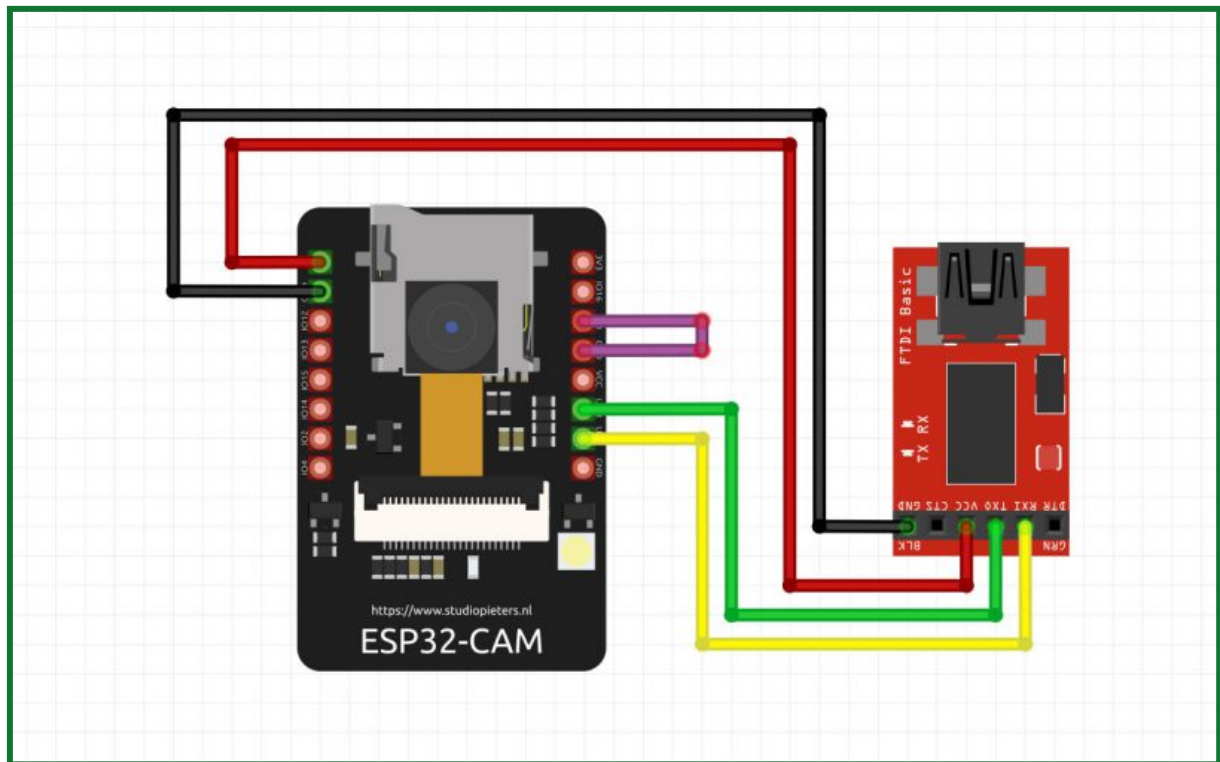


Then you will be able to download the relevant library by going to the Boards section in **Tools> Board> Boards Manager** and ESP32 Search.



Programming ESP32-cam

To program the ESP-CAM board, we need Arduino-IDE software, and of course, download the relevant board in the software environment, as well as install the required libraries. For information on these cases, you can refer to this tutorial. Make the connections according to the schematic below according to the USB TO TTL used. Note that when programming code, ie after compiling the code, the two pins GPIO 0 and GND are connected to each other, and after successfully compiling the code, you must disconnect this connection to run the project.



- The connections are as follows.

ESP32-CAM	USB TO TTL
U0T	RXD
U0R	TXD
5V	VCC
GND	GND

Launching and implementing the project

The first step in using ESP32-CAM in conjunction with Tensorflow.js is to identify the objects that make up the web page where the conclusion occurs. To use the Tensorflow Javascript library, we need to follow these steps: first import the Tensorflow JavaScript libraries, then load the model, in this project the COCO-SSD trained ML model will be used. And create labels for processed objects, which are displayed using the COCO-SSD model on the input video of the identified items by drawing rectangles around the objects.

In this part of the action code, we will introduce Tensorflow.js to analyze the received images in the browser. In the next line, we will also import COCO-SSD models along with Tensorflow.js

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.3.1/dist/tf.min.js"> </script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/coco-ssd@2.1.0"> </script>
```

- In the next section, we will load the introduced models so that they can be

```
function ObjectDetect() {
  result.innerHTML = "Please wait for loading model.";
  cocoSsd.load().then(cocoSsd_Model => {
    Model = cocoSsd_Model;
    result.innerHTML = "";
    getStill.style.display = "block";
  });
}
```

- After our machine detects an object in the image, it should now be visible to the user in the form that squares around the detected image are usually used in image processing, the following code is for this.

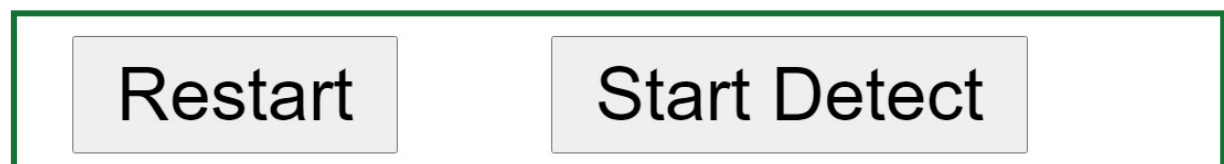
```
if (Predictions.length>0) {
  result.innerHTML = "";
  for (var i=0;i<Predictions.length;i++) {
    const x = Predictions[i].bbox[0];
    const y = Predictions[i].bbox[1];
    const width = Predictions[i].bbox[2];
    const height = Predictions[i].bbox[3];
    context.lineWidth = Math.round(s/200);
```

```
context.strokeStyle = "#00FFFF";
context.beginPath();
context.rect(x, y, width, height);
context.stroke();
context.lineWidth = "2";
context.fillStyle = "red";
context.font = Math.round(s/30) + "px Arial";
context.fillText(Predictions[i].class, x, y);
```

- Complete Object Detection system design project code with ESP32-CAM
- In this part of the action code, we will introduce Tensorflow.js to analyze the received images in the browser. In the next line, we will also import COCO-SSD models along with Tensorflow.js.
- In the next section, we will load the introduced models so that they can be recognized as a result of processing the received image for our machine.
- After our machine detects an object in the image, it should now be visible to the user in the form that squares around the detected image are usually used in image processing, the following code is for this.

Final result

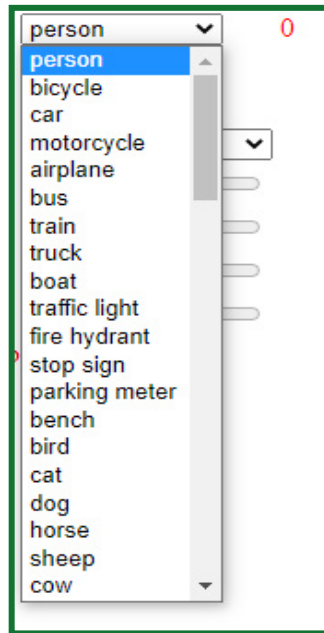
After uploading the desired code, I reset the board in the serial ip of the web server will be displayed for us, such as 192.168.1.103, which we will search for in the browser to enter the web server page, we have to wait for a while until Load the related models, then click on the StartDetect icon to display the images and recognize the defined items. Also, if there is a problem with the image processing or other parts, click on the Restart option.



In the web server, it is possible to count it by specifying the desired item.






By selecting the desired object, the number of detected items will be displayed in front of it.



By selecting the desired object, the number of detected items will be displayed in front of it.

ScoreLimit	0	▼
MirrorImage	yes	▼
Resolution	VGA(640x480)	▼

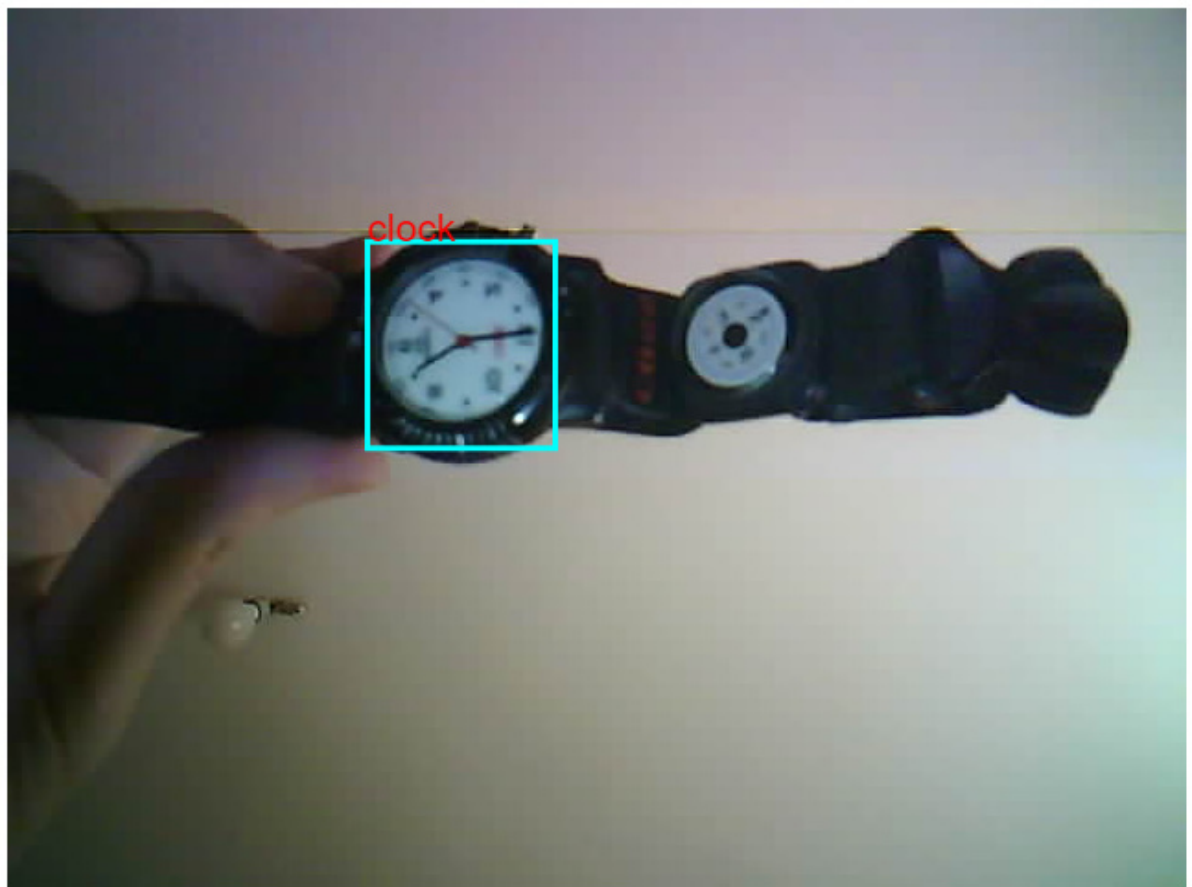
In the web server, it is possible to count it by specifying the desired item.

Quality	
Brightness	
Contrast	

If the object is detected, its name will be displayed in this section, and also the percentage of probability that our machine analysis is correct, as well as the coordinates and number of detected objects can be displayed in this section.

[0] person, 55%, 276, 139, 363, 317

Finally, we have the received image, which will inform the user if he detects an object with a square and also enters its name.



Conclusion

We described how to implement object detection in ESP32-CAM using Tensorflow.js. This tutorial introduces how to stream video from ESP32-CAM and use the Javascript Tensorflow library to identify and classify objects. Do not forget that the inference process runs inside the browser and is not in ESP32-CAM. And with Tensorflow models already trained as COCO-SSD, we are able to identify objects.