

# ESP32-S3 STICK with TinyML

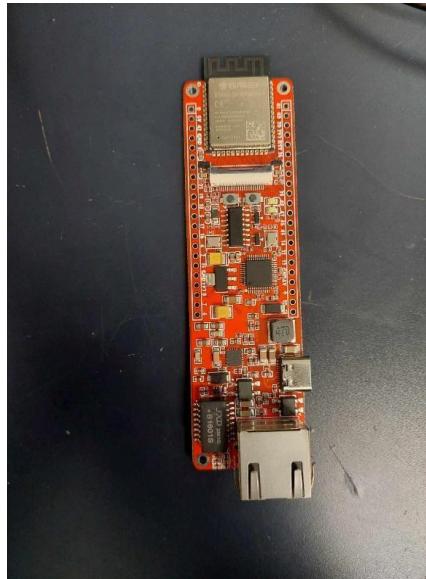
## INTRODUCTION:

Hello everyone! In this tutorial, I will demonstrate how to utilize our pre-trained machine learning model with custom data. Specifically, we will be using Edge Impulse to detect the colors. In our previous tutorial, we relied on a PC and a Python script for face detection. However, in this tutorial, we'll shift to using S3 SoC instead of a PC.

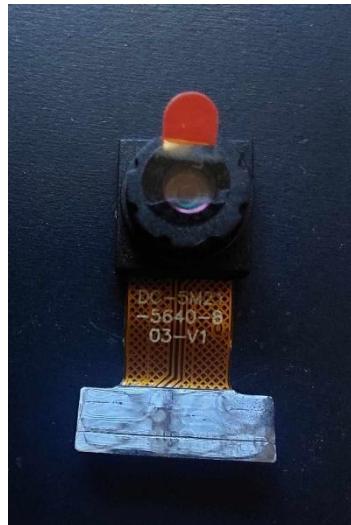
## List of hardware you need:

- 1x USB-C cable
- ESP32-S3 stick
- Camera Ov2640 or Ov5640

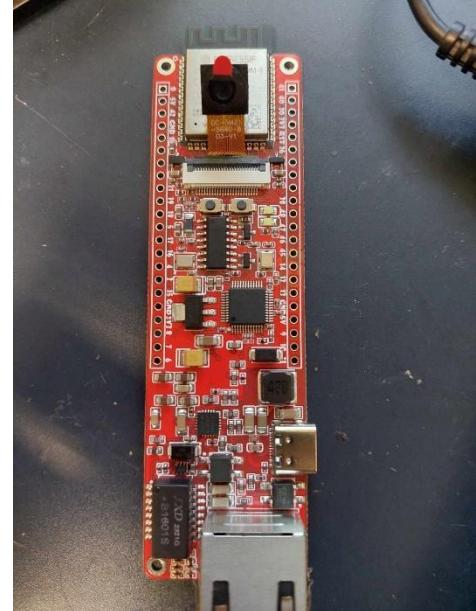
ESP32 STICK



Ov5640 camera



Connected camera to stick

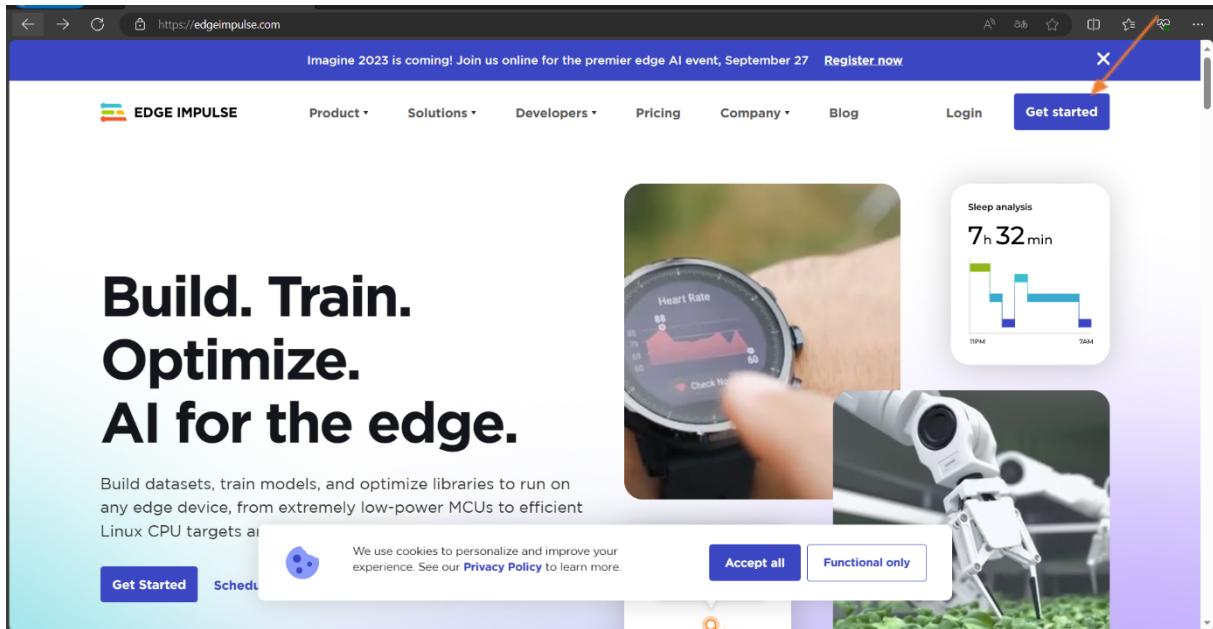


## 1. Software

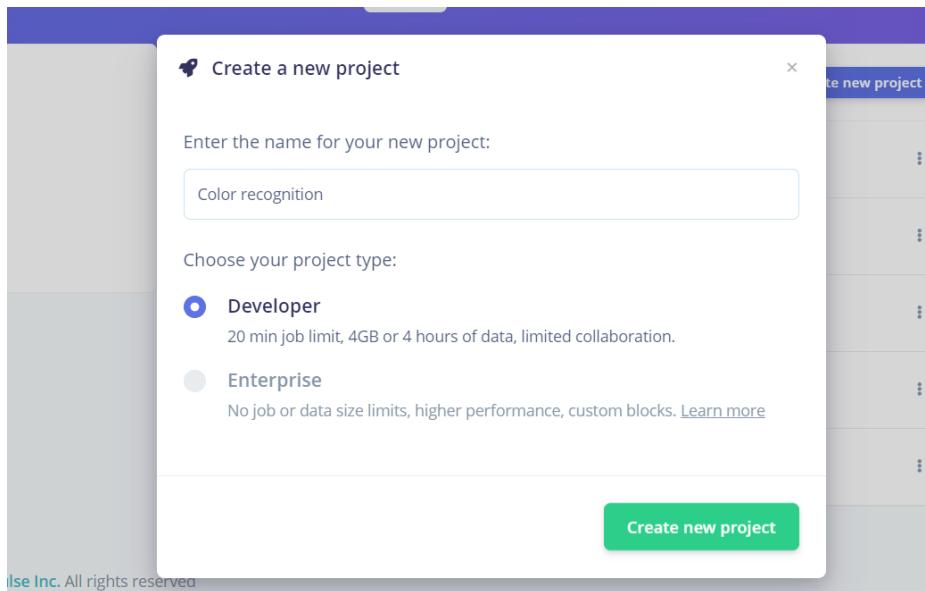
To set up TinyML, we'll be using the Arduino IDE. While we could use VScode as we did in the previous tutorial, Arduino IDE offers a simpler approach because it allows us to include libraries in the form of ZIP files. Additionally, please note that for our pre-trained model, you'll need to register on the Edge Impulse website.

## 2. Registration

On Edge Impulse website click on button 'Get started' and register



Create a project, enter the name of a project for me it's 'color recognition'

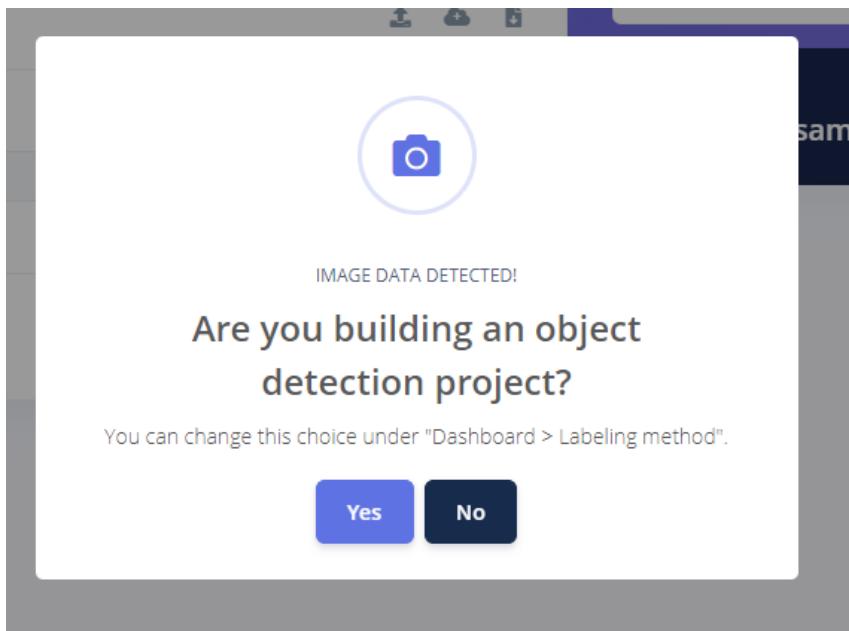


### 3. Uploading data

After creating project you should be on dashboard, click on ‘Collect new data’, if you see window with collecting data by computer, phone or board close it, but if you want to you can. Click on ‘add data’ and ‘upload data’.

The screenshot shows the Edge Impulse dashboard. On the left is a sidebar with various project management and development tools. The main area is divided into sections: 'Getting started' (with options for adding existing data, collecting new data, and uploading models), 'Sharing' (indicating the project is private), 'Collaborators' (listing one collaborator named 'man' who is the owner), 'Summary' (showing 0 devices connected and no data collected), and 'Project info'. In the 'Getting started' section, there's a 'Start with a tutorial' section featuring three circular icons for Motion: Gesture recognition, Images: Object detection, and Audio: Audio classification. A prominent orange arrow points from the text above to the 'Collect new data' button in this section.

you see this while uploading data click ‘No’



You need to save photo in Labeling queue

The screenshot shows the 'Data sources' interface. At the top, there are three tabs: 'Dataset' (selected), 'Data sources', and 'Labeling queue (0)'. An orange arrow points from the text above to the 'Labeling queue (0)' tab. Below the tabs, there are two main sections: 'DATA COLLECTED' (30 items) and 'TRAIN / TEST SPLIT' (empty). The 'DATA COLLECTED' section has a large blue background. The 'TRAIN / TEST SPLIT' section has a white background. At the bottom, there is a 'Dataset' card with 'Training (24)' and 'Test (6)' counts, and standard filtering and sorting icons.

## 4. Distribution

Once you have uploaded your data, click on the icon that allows you to select multiple items. From there, you can edit the names of the labels, such as "red," "green," and "blue." If you wish to create a testing dataset, you can move two photos of each color into the testing dataset.

The screenshot shows the 'Dataset' interface. At the top, there is a 'Dataset' tab and a row of icons: upload, download, and edit. Below that, it says 'Training (24)' and 'Test (6)'. There are filtering and sorting icons. A horizontal bar with buttons for 'Delete (0)', 'Edit labels (0)', 'Move to test set (0)' (which is highlighted with an orange arrow), 'Enable (0)', and 'Disable (0)'. The main area is a table with columns: 'SAMPLE NAME', 'LABEL', and 'ADDED'. The table lists 15 samples: green3 (green, Today, 14:31:11), red10 (red, Today, 14:30:32), red9 (red, Today, 14:30:27), red8 (red, Today, 14:30:19), red6 (red, Today, 14:30:13), red5 (red, Today, 14:30:07), red4 (red, Today, 14:30:04), red3 (red, Today, 14:30:00), red1 (red, Today, 14:29:52), green10 (green, Today, 14:29:36), green9 (green, Today, 14:29:34), and green7 (green, Today, 14:29:26). At the bottom, there are navigation arrows and page numbers (1, 2).

## 5. Training model

Next on sidebar click on 'Create impulse'

The screenshot shows the Edge Impulse web interface. On the left, a sidebar lists various options: Data acquisition, Impulse design (highlighted with a red arrow), Create impulse, EON Tuner, Retrain model, Live classification, Model testing, Versioning, Deployment, and GETTING STARTED. Below these are Documentation and Forums links. A 'Try Enterprise Free' button is also present. The main workspace is divided into several panels: 'Image data' (red background) showing input axes for image width (96) and height (96), and a note about resize mode; 'Image' (white background) showing input axes (1) and a checked 'Image' checkbox; 'Transfer Learning (Images)' (purple background) showing name 'Transfer learning', input features (Image checked), and output features (3 (blue, green, red)); and 'Output features' (green background) showing '3 (blue, green, red)'. At the bottom, there are dashed boxes for 'Add a processing block' and 'Add a learning block'.

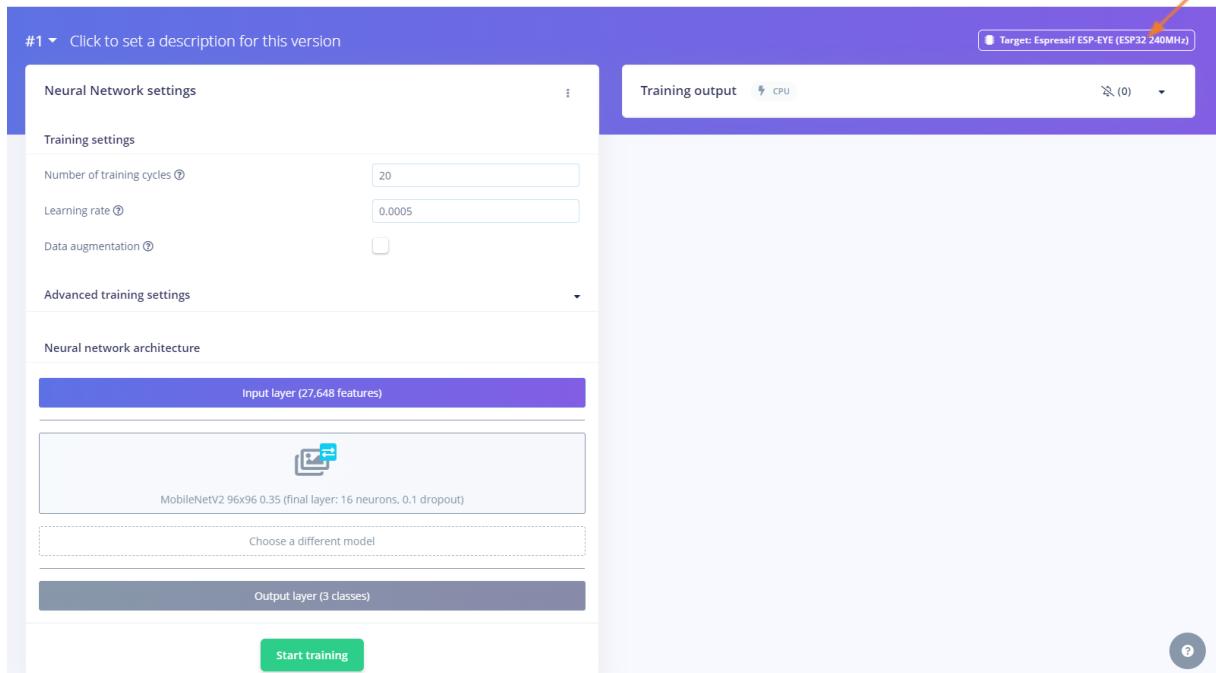
Save parameters, color RGB

The screenshot shows the 'Raw data' section of the Edge Impulse interface. It displays a green car image and various parameters. A red arrow points to the 'Save parameters' button at the bottom of the 'Parameters' section. The 'DSP result' and 'On-device performance' sections are also visible.

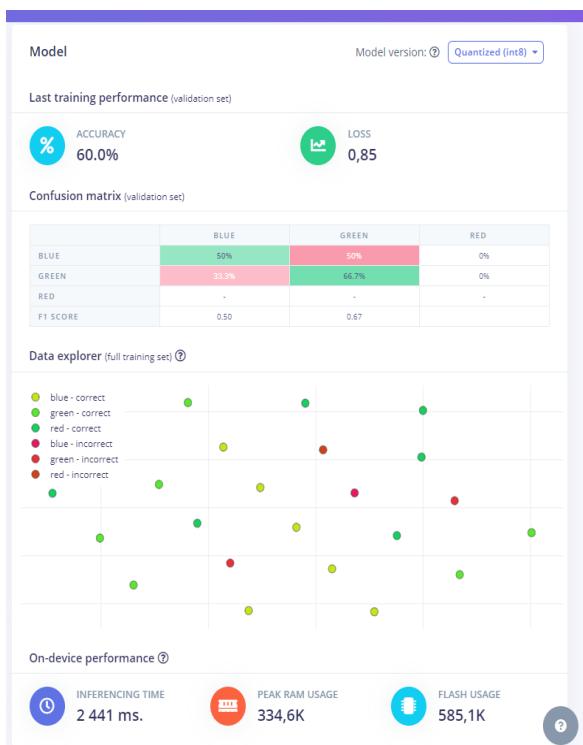
Generate features, and after generating click on sidebar 'Transfer learning'

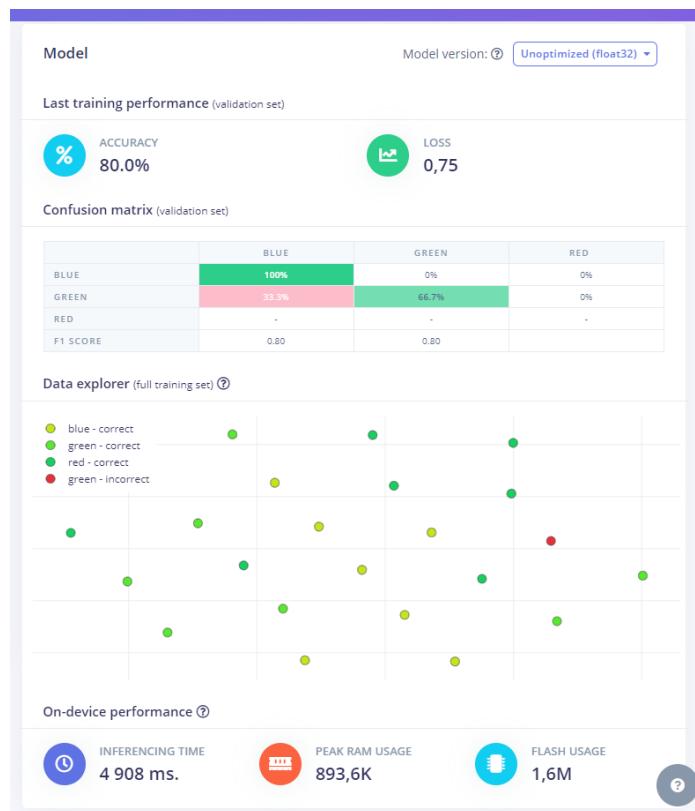
The screenshot shows the 'Training set' and 'Feature explorer' sections. The 'Training set' table includes 'Data in training set' (24 items) and 'Classes' (3 (blue, green, red)). The 'Feature explorer' panel states 'No features generated yet.' A red arrow points to the 'Generate features' button at the bottom of the 'Training set' section. The top bar shows '#1 Click to set a description for this version' and 'Parameters Generate features' buttons.

After that select target for ESP-EYE because it's similar to our stick and click 'start training'



Accuracy is 60% for optimized and 80% for unoptimized but **inference time** is twice as much as optimized. Accuracy could be better, but for training we had only 24 pictures and for testing 6 pictures.





## 6. Deployment

In sidebar click on ‘Deployment’, choose Arduino library, turn off EON compiler because for ESP32-S3 Edge Impulse hasn’t released SDK yet and click on build

## Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)



Arduino library X



SELECTED DEPLOYMENT



Arduino library

An Arduino library with examples that runs on most Arm-based Arduino development boards.

### MODEL OPTIMIZATIONS

Model optimizations can increase on-device performance but may reduce accuracy.

[Enable EON™ Compiler](#) *Same accuracy, up to 50% less memory.* [Learn more](#)

Quantized (int8)

[Selected ✓](#)

	IMAGE	TRANSFER LEARNING	TOTAL
LATENCY	15 ms.	2 441 ms.	2 456 ms.
RAM	4.0K	353.1K	353.1K
FLASH	-	683.4K	-
ACCURACY			-

Unoptimized (float32)

[Select](#)

	IMAGE	TRANSFER LEARNING	TOTAL
LATENCY	15 ms.	4 908 ms.	4 923 ms.
RAM	4.0K	1.1M	1.1M
FLASH	-	1.7M	-
ACCURACY			-

To compare model accuracy, run model testing for all available optimizations.

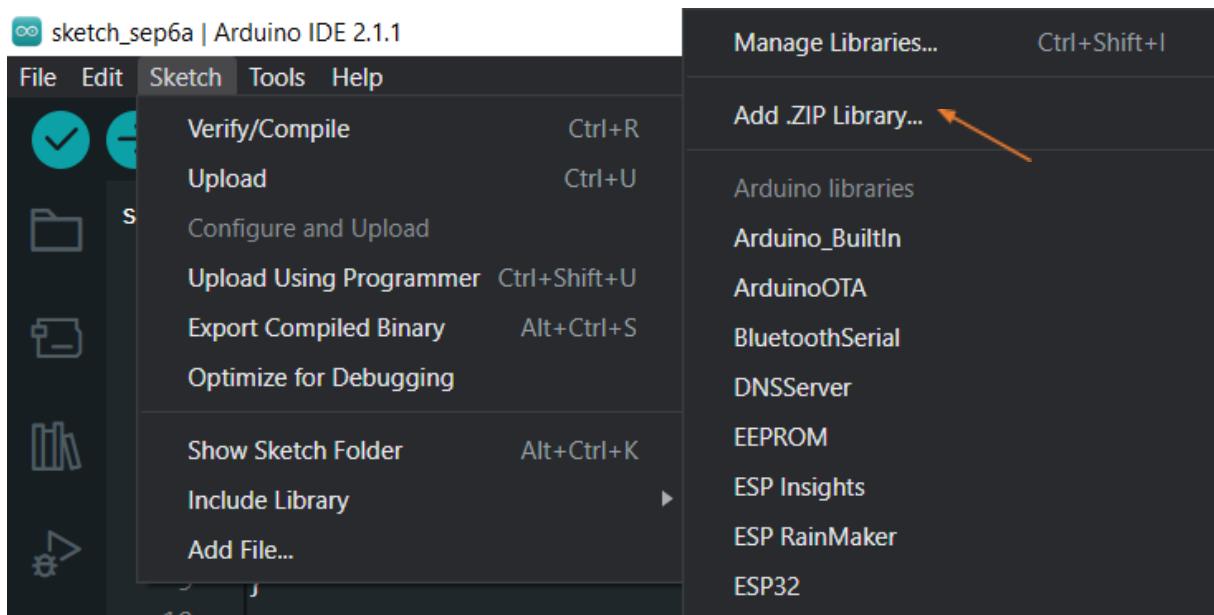
[Run model testing](#)

Estimate for Espressif ESP-EYE (ESP32 240MHz) · [Change target](#)

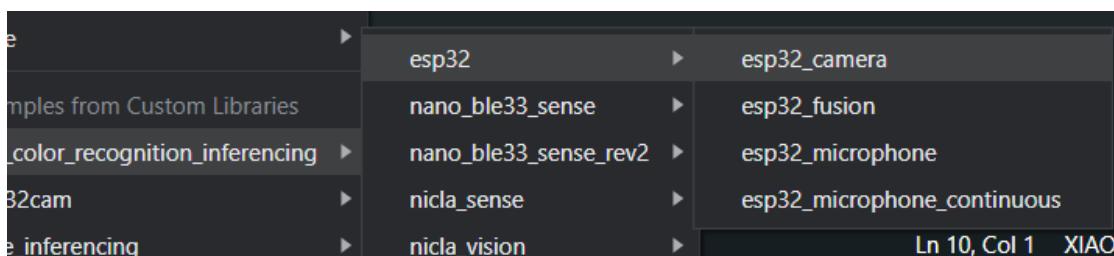
[Build](#)

## 7. Including to the library

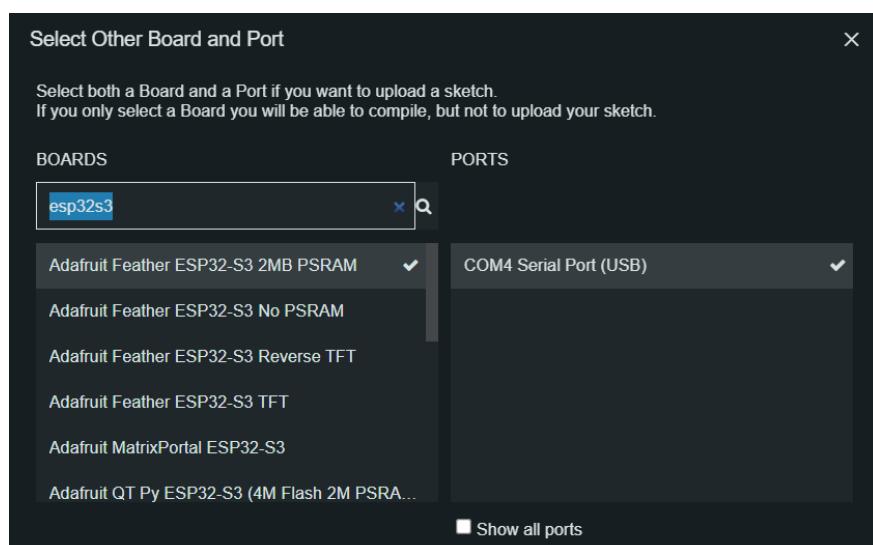
After downloading zip file include file in Arduino IDE



After library is installed go to File>Examples and scroll to the bottom for examples from custom libraries. Library name should be same name as project and choose camera



You can select this board with PSRAM or XIAO\_ESP32S3, it's same SoC



## 8. Configuration

You must configurate pins, in this screenshot you can delete it and set up like this below:

```
#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//define CAMERA_MODEL_AI_THINKER // Has PSRAM
#if defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM      -1
#define RESET_GPIO_NUM     -1
#define XCLK_GPIO_NUM       4
#define SIOD_GPIO_NUM      18
#define SIOC_GPIO_NUM       23
#define Y9_GPIO_NUM         36
#define Y8_GPIO_NUM         37
#define Y7_GPIO_NUM         38
#define Y6_GPIO_NUM         39
#define Y5_GPIO_NUM         35
#define Y4_GPIO_NUM         14
#define Y3_GPIO_NUM         13
#define Y2_GPIO_NUM         34
#define VSYNC_GPIO_NUM      5
#define HREF_GPIO_NUM       27
#define PCLK_GPIO_NUM       25
#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
```

```
#include "esp_camera.h"

// Select camera model - find more camera
// https://github.com/espressif/arduino-esp32

#define CAMERA_MODEL_XIAO_ESP32S3 // Has

#define PWDN_GPIO_NUM      -1
#define RESET_GPIO_NUM     -1
#define XCLK_GPIO_NUM      15
#define SIOD_GPIO_NUM      4
#define SIOC_GPIO_NUM      5

#define Y9_GPIO_NUM         16
#define Y8_GPIO_NUM         17
#define Y7_GPIO_NUM         18
#define Y6_GPIO_NUM         12
#define Y5_GPIO_NUM         10
#define Y4_GPIO_NUM         8
#define Y3_GPIO_NUM         9
#define Y2_GPIO_NUM         11
#define VSYNC_GPIO_NUM      6
#define HREF_GPIO_NUM       7
#define PCLK_GPIO_NUM       13
```

Don't forget to disable the ESP NN acceleration. Failure to do so may result in the following error message:

Go to:

Documents\Arduino\libraries\Car\_color\_recognition\_inferencing\src\edge-impulse-sdk\classifier\ei\_classifier\_config.h

And change this line from 1 to 0 and save:

```
#define EI_CLASSIFIER_TFLITE_ENABLE_ESP_NN 0
```

You must do some changes in this code because generated code be EI it doesn't work for this stick, I mean code works but in serial monitor it don't have any outputs

Change every: Serial > Serial0

ei\_printf > Serial0.printf

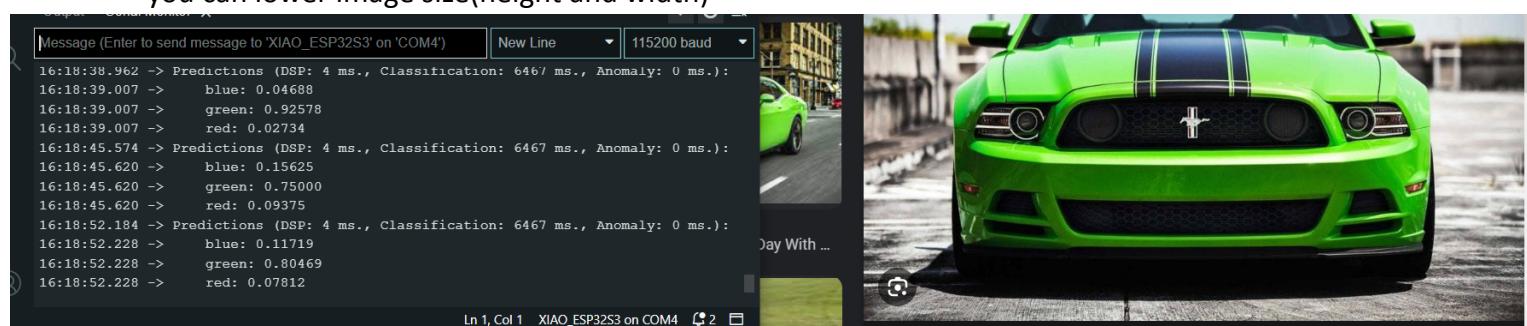
it should look like this:

```
void setup()
{
    // put your setup code here, to run once:
    Serial0.begin(115200);
    //comment out the below line to start inference immediately after upload
    while (!Serial0);
    Serial0.println("Edge Impulse Inferencing Demo");
    if (ei_camera_init() == false) {
        Serial0.printf("Failed to initialize Camera!\r\n");
    }
    else {
        Serial0.printf("Camera initialized\r\n");
    }

    Serial0.printf("\nStarting continious inference in 2 seconds...\r\n");
    ei_sleep(2000);
}
```

## 9 . Test

You can see it, I pointed with camera on image of green car, it took 6 seconds which is slow you can lower image size(height and width)



Accurate of around 60% for blue

The screenshot shows a terminal window titled "Serial Monitor X" with the baud rate set to 115200. The output window displays a series of predictions for a blue car. The log entries show the following data:

```
16:25:02.744 -> Predictions (DSP: 4 ms., Classification: 6466 ms., Anomaly: 0 ms.):
16:25:02.744 ->     blue: 0.67578
16:25:02.744 ->     green: 0.22656
16:25:02.744 ->     red: 0.09766
16:25:09.319 -> Predictions (DSP: 4 ms., Classification: 6466 ms., Anomaly: 0 ms.):
16:25:09.366 ->     blue: 0.62500
16:25:09.366 ->     green: 0.23828
16:25:09.366 ->     red: 0.13672
16:25:15.927 -> Predictions (DSP: 4 ms., Classification: 6466 ms., Anomaly: 0 ms.):
16:25:15.972 ->     blue: 0.57031
16:25:15.972 ->     green: 0.21875
16:25:15.972 ->     red: 0.21094
```

To the right of the terminal is a large image of a blue Lexus GS sedan. Below the image is a caption: "Lexus GS Blue Car PNG Image | Lexus, Blue car, Car".

Here's another model for recognition with an accuracy rate of 75%. This model incorporates a larger dataset with approximately 65 images, evenly distributed between men and women.

The screenshot shows a terminal window titled "Serial Monitor X" with the baud rate set to 115200. The output window displays predictions for a man's face. The log entries show the following data:

```
16:28:49.525 ->     man: 0.39844
16:28:49.525 ->     woman: 0.60156
16:28:52.050 -> Predictions (DSP: 4 ms., Classification: 2388 ms., Anomaly: 0 ms.):
16:28:52.050 ->     man: 0.88281
16:28:52.050 ->     woman: 0.11719
16:28:54.568 -> Predictions (DSP: 4 ms., Classification: 2388 ms., Anomaly: 0 ms.):
16:28:54.568 ->     man: 0.58594
16:28:54.568 ->     woman: 0.41406
16:28:57.105 -> Predictions (DSP: 4 ms., Classification: 2388 ms., Anomaly: 0 ms.):
16:28:57.105 ->     man: 0.64453
16:28:57.105 ->     woman: 0.35547
```

To the right of the terminal is a large image of a young man's face. Below the image is a caption: "That Make You Attr...".

It takes some time to recognize if it's a woman or a man

The screenshot shows a terminal window titled "Serial Monitor X" with the baud rate set to 115200. The output window displays predictions for a woman's face. The log entries show the following data:

```
16:30:17.823 ->     man: 0.55078
16:30:17.823 ->     woman: 0.44922
16:30:20.332 -> Predictions (DSP: 4 ms., Classification: 2388 ms., Anomaly: 0 ms.):
16:30:20.332 ->     man: 0.52344
16:30:20.332 ->     woman: 0.47656
16:30:22.855 -> Predictions (DSP: 4 ms., Classification: 2388 ms., Anomaly: 0 ms.):
16:30:22.855 ->     man: 0.28516
16:30:22.855 ->     woman: 0.71484
16:30:25.357 -> Predictions (DSP: 4 ms., Classification: 2388 ms., Anomaly: 0 ms.):
16:30:25.404 ->     man: 0.07812
16:30:25.404 ->     woman: 0.92188
```

To the right of the terminal is a large image of a woman's face. Below the image is a caption: "These women showcase the true diversity of Indian beauty | VOGUE India | Vogue India".