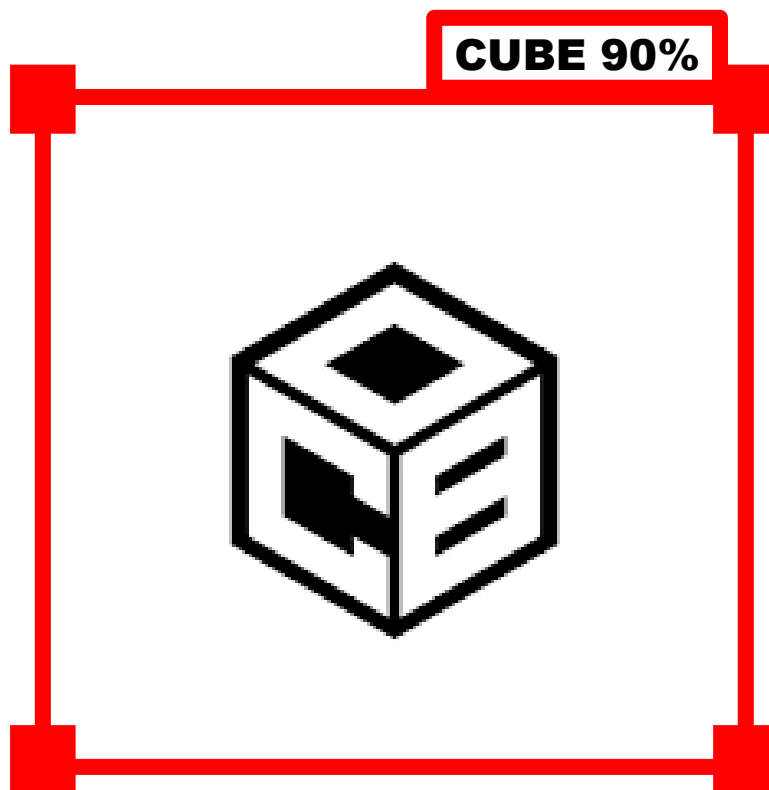


Reconocimiento de objetos con IA usando un ESP32-CAM



Unidades
Tecnológicas
de Santander



Objetivo

La detección de objetos con inteligencia artificial ya no requiere equipos costosos. Gracias al **ESP32-CAM**, ahora es posible capturar imágenes y reconocer objetos en tiempo real desde un dispositivo pequeño y económico.

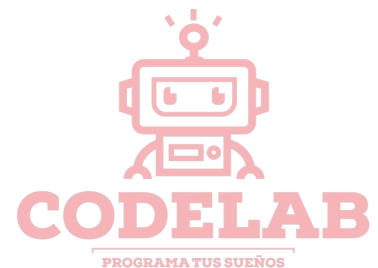
Esta guía te enseñará paso a paso cómo implementar un sistema de detección de objetos usando **IA y un ESP32**, combinando herramientas como **Edge Impulse**. No necesitas experiencia previa en IA: con este proyecto aprenderás los conceptos básicos, entrenarás tu propio modelo y lo pondrás en marcha en el ESP32.

Ideal para estudiantes, docentes o entusiastas, este es tu punto de partida para entrar al mundo de la visión artificial embebida.

Agradecemos a las Unidades Tecnológicas de Santander por permitir la realización de esta actividad, brindando un espacio de aprendizaje e innovación que impulsa el desarrollo de habilidades tecnológicas en los estudiantes.



Unidades
Tecnológicas
de Santander



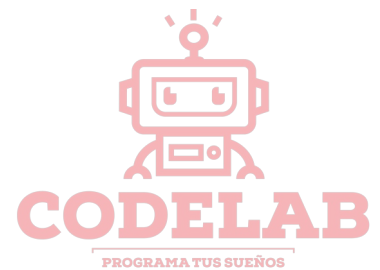
Partes para realizar el ejercicio

A continuación se mostrarán los pasos a seguir para poder realizar el reconocimiento de objetos mediante IA utilizando un ESP32-CAM:

- Tomar fotos para entrenamiento.
- Realizar etiquetado de imágenes.
- Crear impulso y configuración de las imágenes.
- Realizar entrenamiento de la IA.
- Construir librería necesaria para subir al ESP32-CAM
- Probar resultado del entrenamiento.



Unidades
Tecnológicas
de Santander

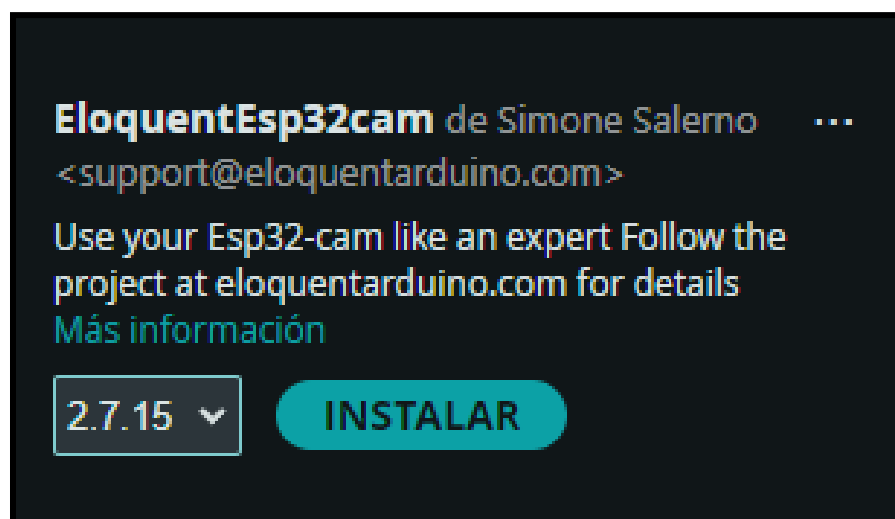


Toma de fotos para entrenamiento

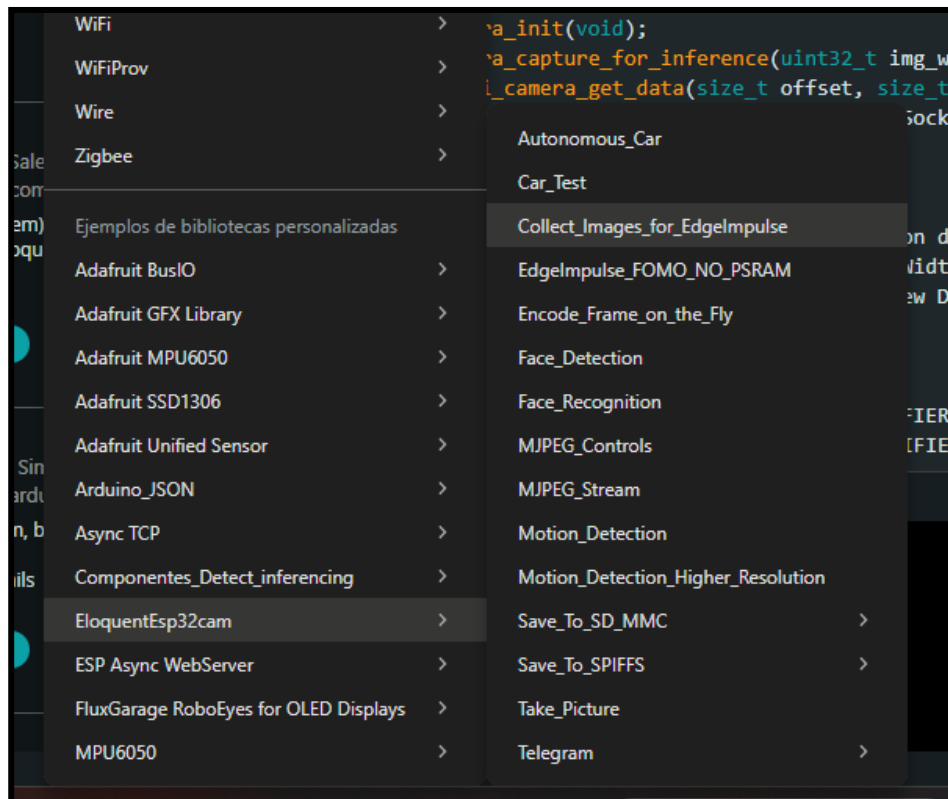
Para poder detectar objetos debemos enseñarle a nuestra IA que objetos vamos a reconocer, por lo cual vamos a tomarle fotos en todas las direcciones desde el mismo ángulo, idealmente se debe tener el mismo fondo y si es posible tener la misma cantidad de luz y misma distancia a cámara.

Descarga de librería

En el IDE de Arduino iremos al gestor de bibliotecas y buscaremos la biblioteca "EloquentEsp32cam", la instalaremos ya que nos brinda un código que lo usaremos mas adelante.



El siguiente paso es buscar entre los ejemplos de la librería que acabamos de escoger, el código de "Collect_images_for_edgeimpulse"



En las líneas 15 y 16 de este código pondremos la red a la cual nos vamos a conectar tanto en el ESP32-CAM como en nuestro computador ya que deben estar en la misma red para que esto funcione.

```
15  #define WIFI_SSID "SSID"
16  #define WIFI_PASS "PASSWORD"
17  #define HOSTNAME "esp32cam"
```

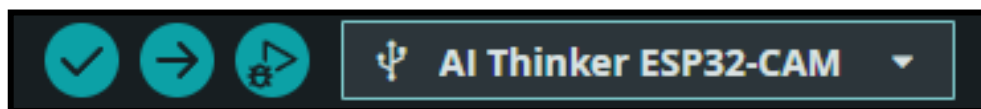
Luego la línea 36 cambiaremos un poco el código de la siguiente manera:

```
35 // replace with your own model!  
36 camera.pinout.wroom_s3();
```



```
35 // replace with your own model!  
36 camera.pinout.aithinker();
```

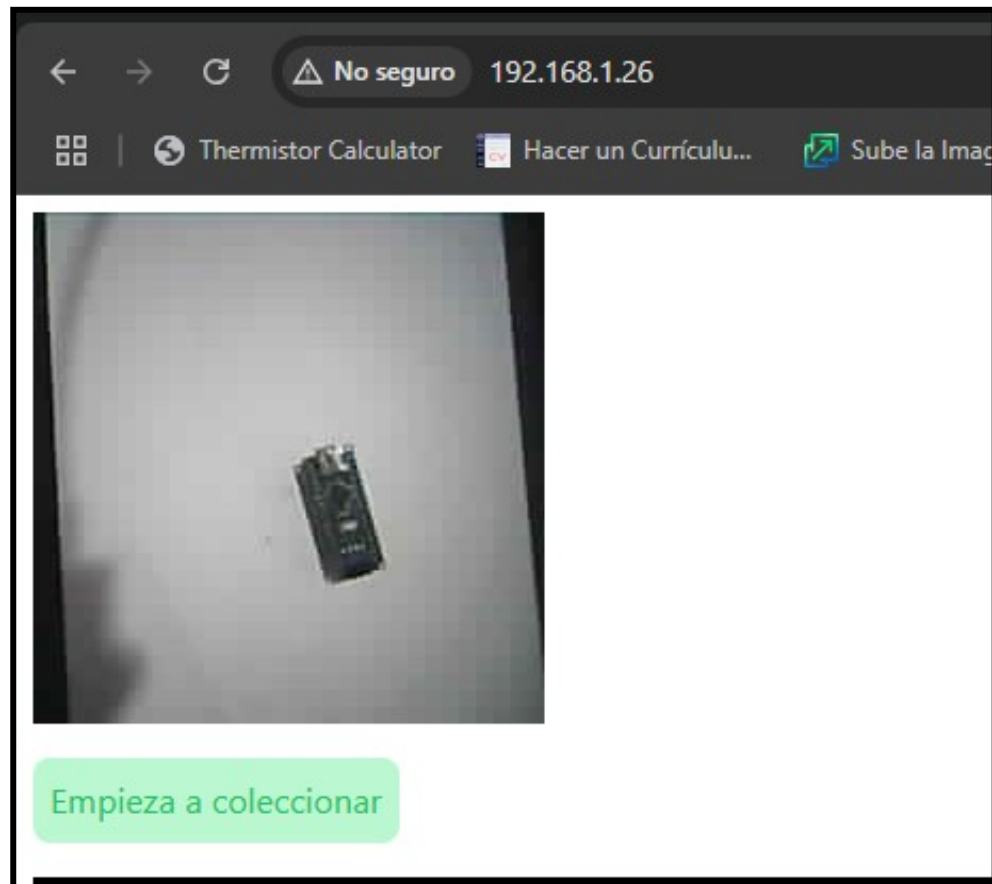
Seleccionamos el puerto y modelo de nuestra placa.



Una vez puesta la red WIFI correctamente y el modelo de nuestro ESP32-CAM, procedemos a subir el código y ver nuestro monitor serial

```
load:0x40080404,len:3476  
entry 0x400805b4  
__IMAGE_COLLECTION_SERVER__  
Camera OK  
WiFi OK  
Image Collection Server OK  
Image Collection Server is available at http://192.168.1.26
```

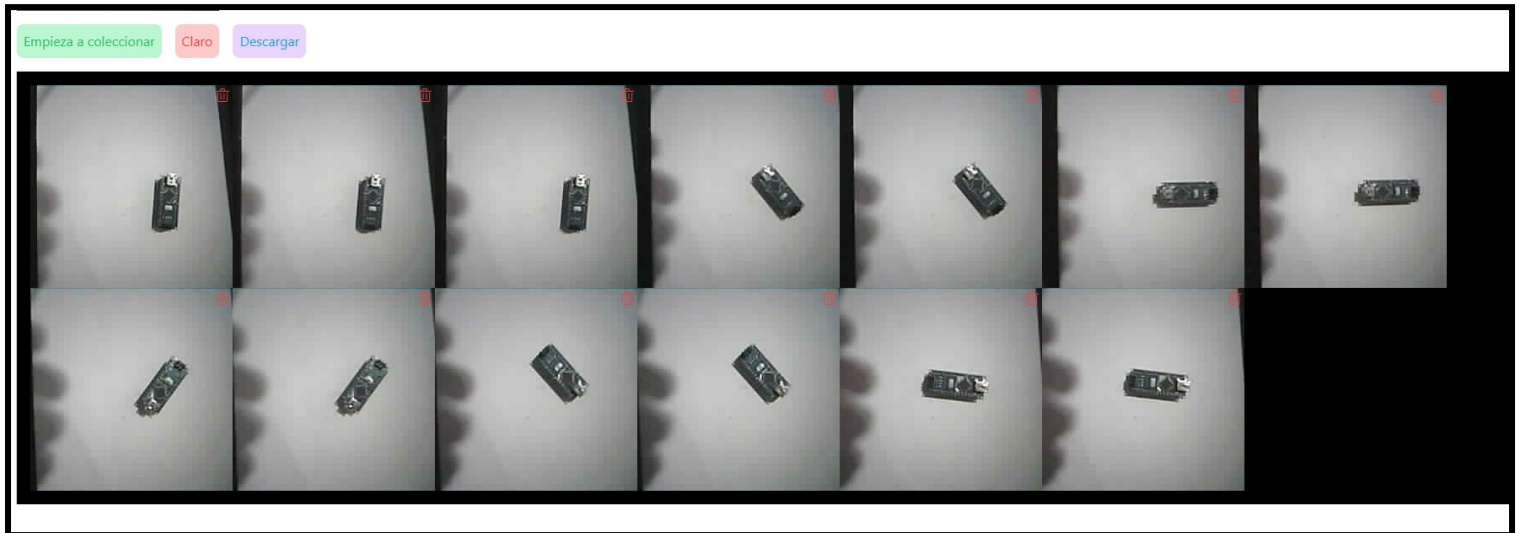
En esta nos brinda una dirección web, la cual pegaremos en nuestro navegador de confianza.



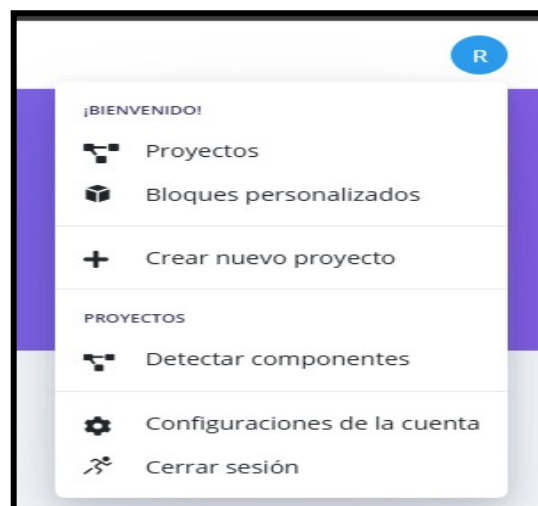
Luego de esto posicionamos la cámara en un lugar fijo y daremos clic a “empieza a seleccionar”, tomaremos alrededor de 19 imágenes o más, entre mayor sea la cantidad de imágenes mejor será la precisión del entrenamiento, recuerda girar el objeto para que lo pueda detectar en todos los sentidos

Una vez obtenidas todas las fotos necesarias daremos clic en descargar y le daremos nombre, en este caso le pondré "NANO", y lo hacemos con los demás objetos

Imagen solo de referencia, en la prueba real se realizaron mas fotografías



Luego de ello ingresaremos a la plataforma de Edge impulse y nos registraremos



Daremos clic en crear nuevo proyecto

Crear un nuevo proyecto

Introduzca el nombre para su nuevo proyecto:

Componentes_Detect

Elige tu tipo de proyecto:

☒ **Personal**
Límite de trabajo de 60 minutos, 4 GB o 4 horas de datos, colaboración limitada.

☐ **Empresa**
Sin límites de tamaño de trabajo o datos, mayor rendimiento, bloques personalizados.

Elige la configuración de tu proyecto:

☒ **Público**
Cualquier persona en internet puede ver y clonar este proyecto bajo la [licencia BSD de 3 cláusulas](#). Solo los usuarios invitados podrán editarlo.

☐ **Privado** (Quedan 3 de 3)
Sólo los usuarios invitados pueden editar y ver su proyecto.

¿Quieres acceso completo a todas las funciones y proyectos ilimitados? [Prueba Enterprise gratis.](#)

Crear nuevo proyecto

Le damos un nombre y creamos un nuevo proyecto, luego de ello en el panel principal de nuestro proyecto daremos clic en agregar datos existentes:

Detectar componentes

Este es tu proyecto Edge Impulse. Desde aquí, adquieres nuevos datos de entrenamiento, diseñas impulsos y entrenas modelos.

+ Nueva etiqueta

Empezando

Comience a crear su conjunto de datos o valide el rendimiento de su modelo en el dispositivo:



Agregar datos existentes



Recopilar nuevos datos



Sube tu modelo

Empezar con un tutorial

Luego daremos clic en subir datos

Agregar datos existentes



Subir datos



Agregar depósito de almacenamiento

Allí cargaremos todas las imágenes que hemos tomado de todos los objetos

Modo de carga

☐ Seleccionar archivos individuales ?

☒ Seleccione una carpeta ?

Seleccionar archivos

Sin archivos seleccionados

Subir a la categoría

☒ Dividir automáticamente entre entrenamiento y prueba ?

☐ Capacitación

☐ Pruebas

Etiqueta

☒ Inferir a partir del nombre del archivo ?

☐ Dejar los datos sin etiquetar ?

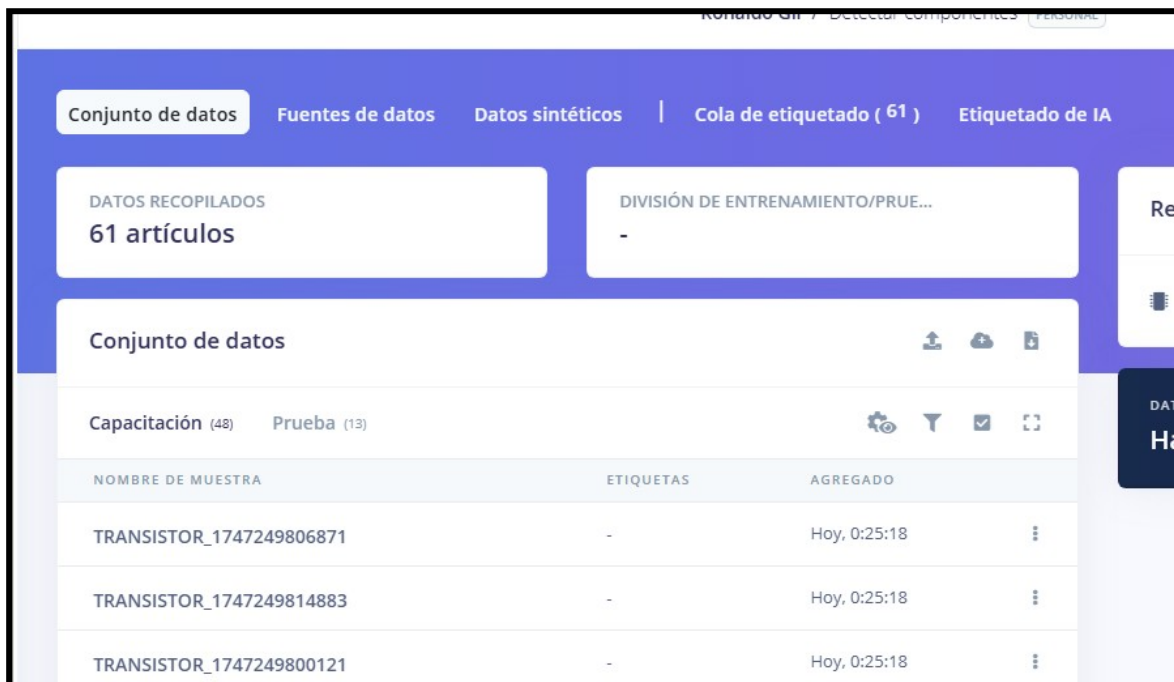
☐ Introduzca la etiqueta:

Debes subir absolutamente todas las imágenes para poder realizar el entrenamiento, mantén las configuraciones que se observan en la imagen.

Etiquetado de imágenes

Para que la IA los aprenda a diferenciar, debemos decirle que objeto es y donde se encuentra ubicado, a esto le llamamos etiquetado.

Daremos clic en la opción de cola de etiquetado



Es aquí donde realizaremos la acción imagen por imagen para que la IA pueda entrenarse de forma correcta.



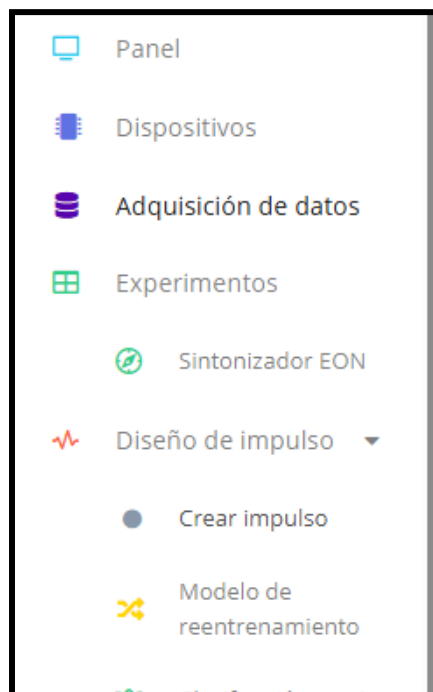
Vamos a encerrar y dar nombre a la etiqueta de forma correcta, recuerda que los objetos deben llevar la misma etiqueta con exactamente el mismo nombre a este yo le llamaré “NANO” por lo cual no pueden haber otras como “Nano” porque lo toma como un objeto diferente.



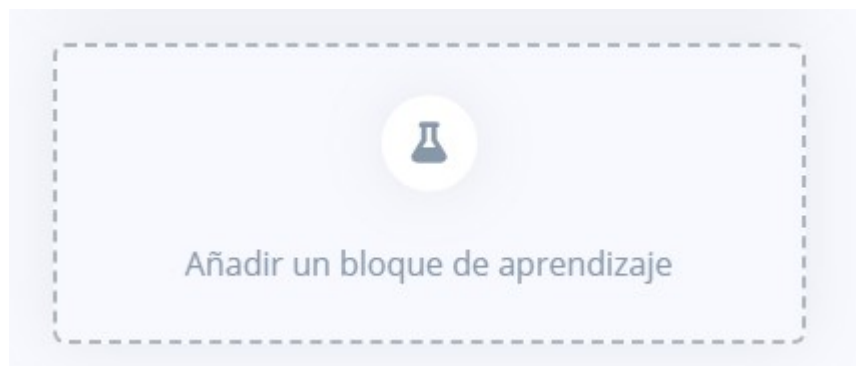
Guardamos la etiqueta y hacemos este paso con todas las imágenes.

Creación de impulso y configuración de imágenes.

En el panel lateral izquierdo iremos a la opción de crear impulso



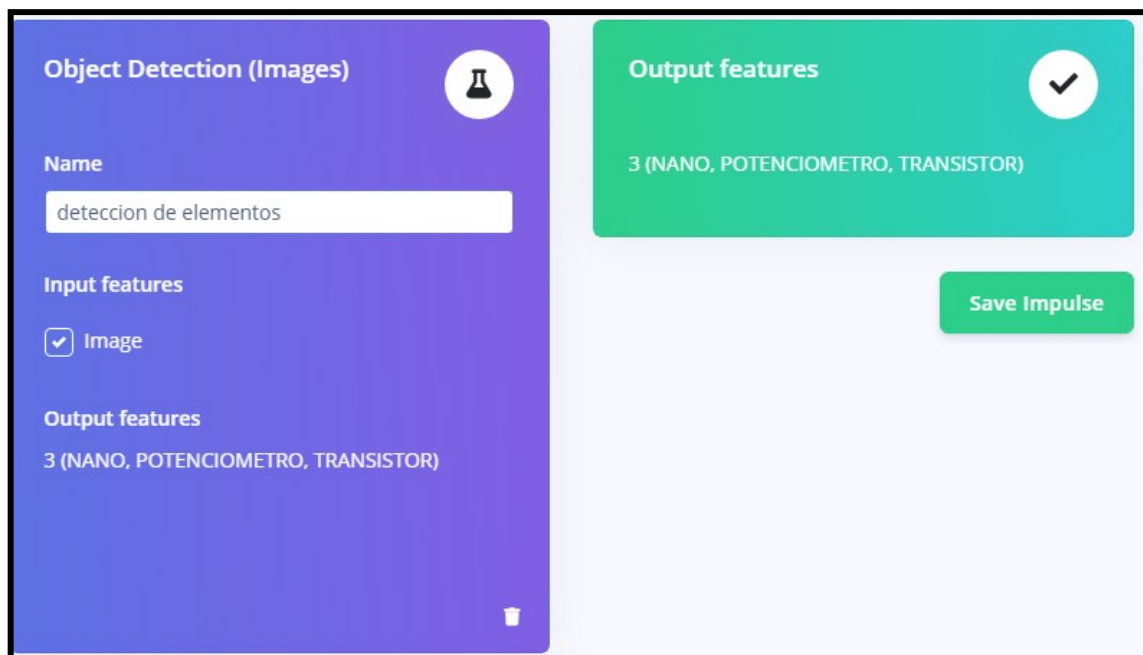
Cerramos la ventana emergente y damos clic en añadir un bloque de aprendizaje



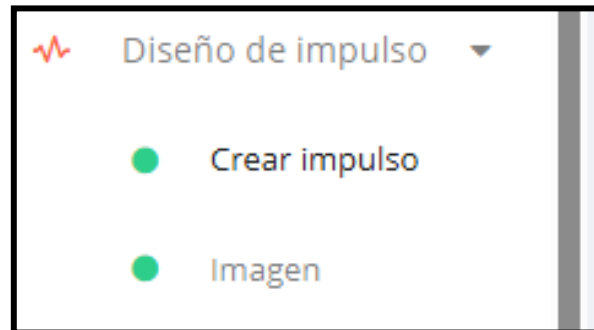
Luego de ello agregamos el de detección de objetos



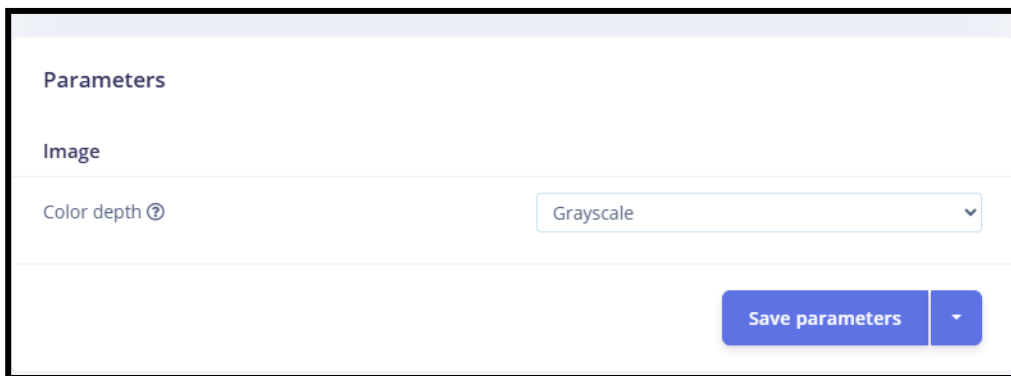
Y damos clic en guardar impulso.



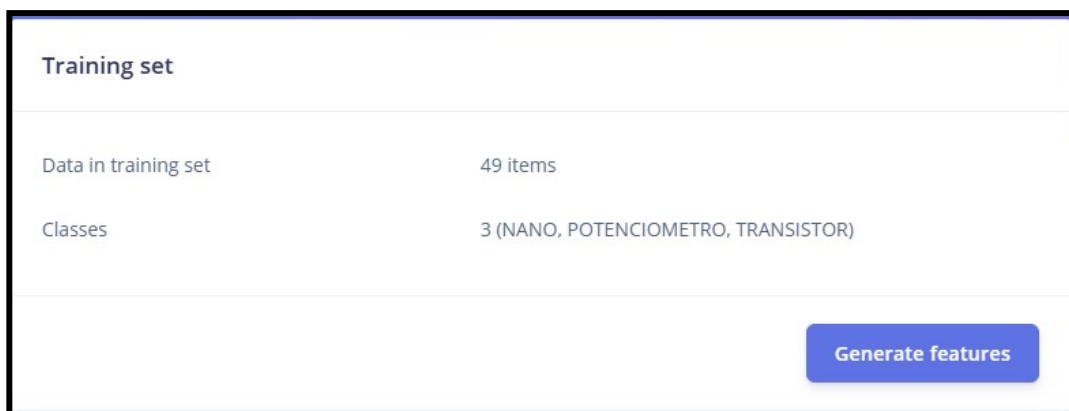
Luego en el panel izquierdo daremos clic en la opción de imagen.



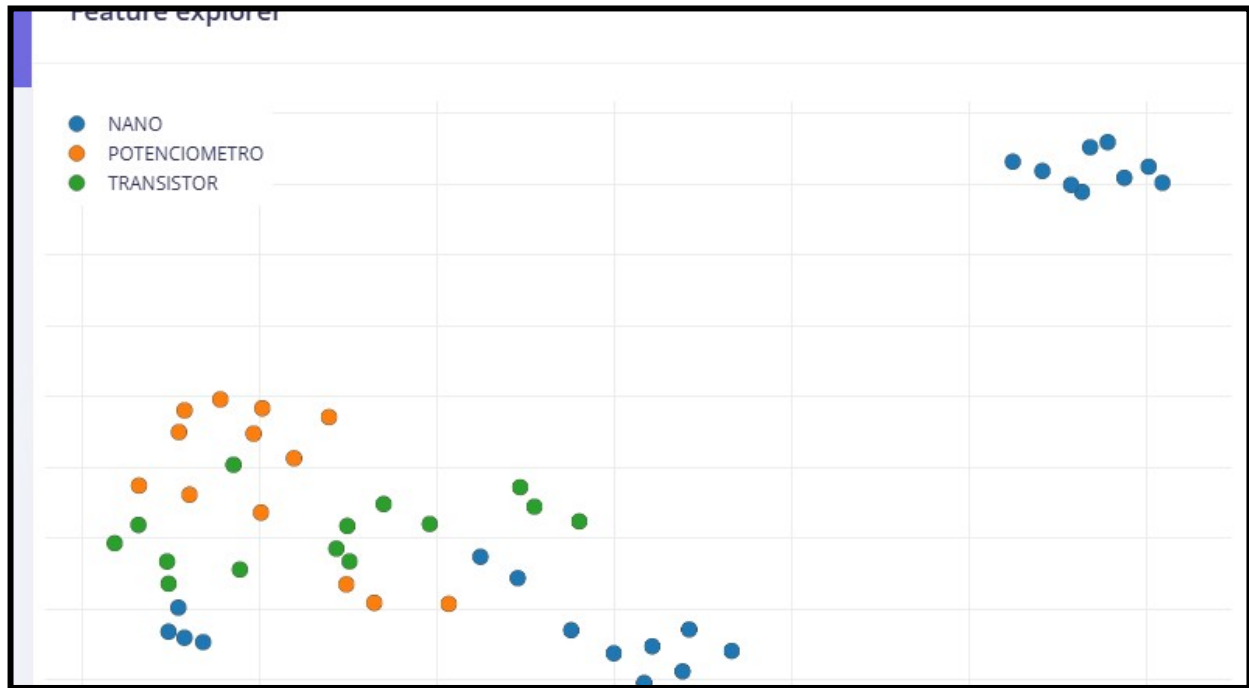
Lo dejaremos en escala de grises y guardamos los parámetros



Luego daremos clic en generate features



Debe generarnos un mapa así con las etiquetas que hemos creado, en mi caso son 3

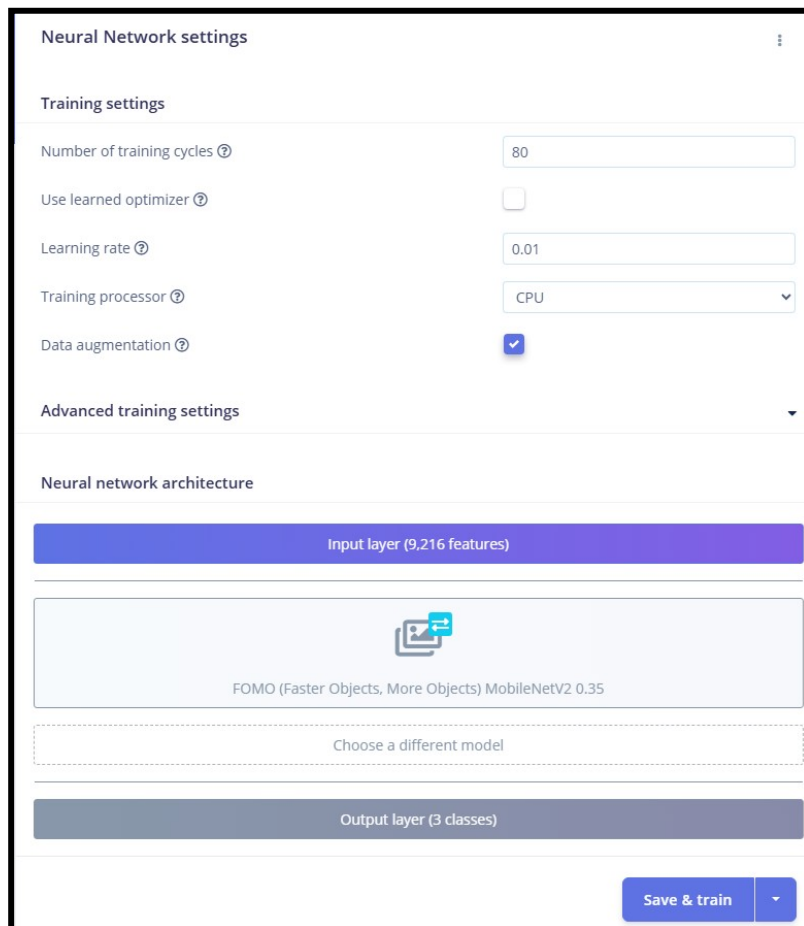


Luego de esto, el sistema está listo para entrenarse!

Entrenamiento de reconocimiento de objetos.

En el panel lateral izquierdo iremos a la opción de detección de elementos.

Pondremos la siguiente configuración para nuestra primera prueba y daremos clic en guardar y entrenar.



The image shows a 'Neural Network settings' panel. It is divided into three main sections: 'Training settings', 'Advanced training settings', and 'Neural network architecture'. In the 'Training settings' section, there are five items: 'Number of training cycles' set to 80, 'Use learned optimizer' which is unchecked, 'Learning rate' set to 0.01, 'Training processor' set to CPU, and 'Data augmentation' which is checked. The 'Advanced training settings' section is currently collapsed. The 'Neural network architecture' section shows a diagram with three layers: an 'Input layer (9,216 features)' in a blue bar, a middle layer labeled 'FOMO (Faster Objects, More Objects) MobileNetV2 0.35' with a small icon, and an 'Output layer (3 classes)' in a grey bar. Below the middle layer is a dashed box with the text 'Choose a different model'. At the bottom right of the panel is a blue button labeled 'Save & train'.

Neural Network settings

Training settings

Number of training cycles 80

Use learned optimizer ☐

Learning rate 0.01

Training processor CPU

Data augmentation ☒

Advanced training settings

Neural network architecture

Input layer (9,216 features)

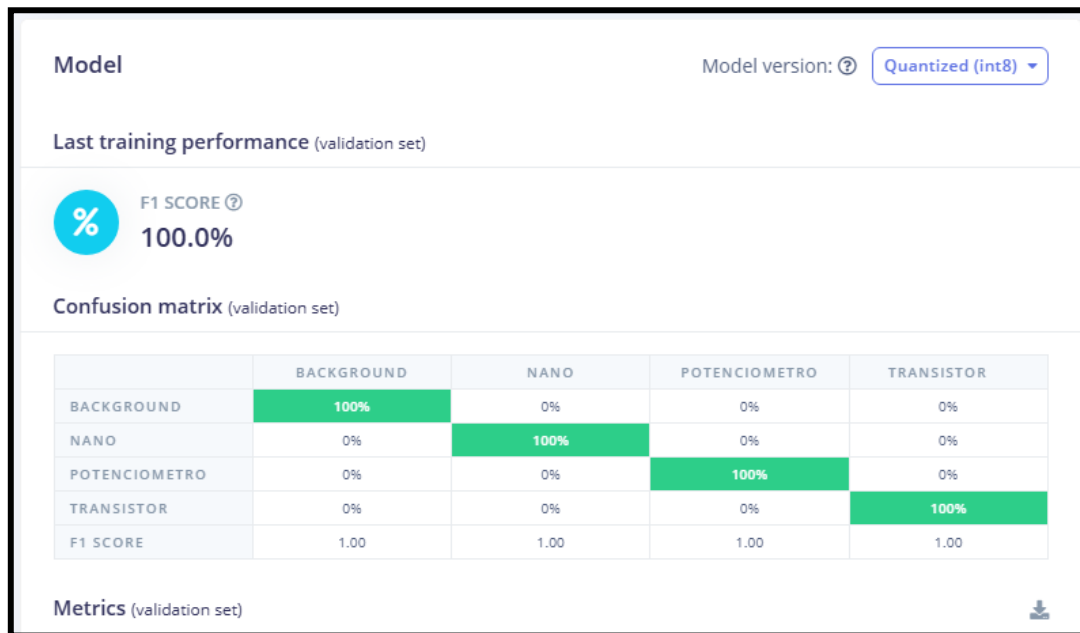
FOMO (Faster Objects, More Objects) MobileNetV2 0.35

Choose a different model

Output layer (3 classes)

Save & train

Si el entrenamiento se realiza al 100% significa que se ha realizado correctamente, de no ser así, trata aumentando el numero de épocas y entrena nuevamente



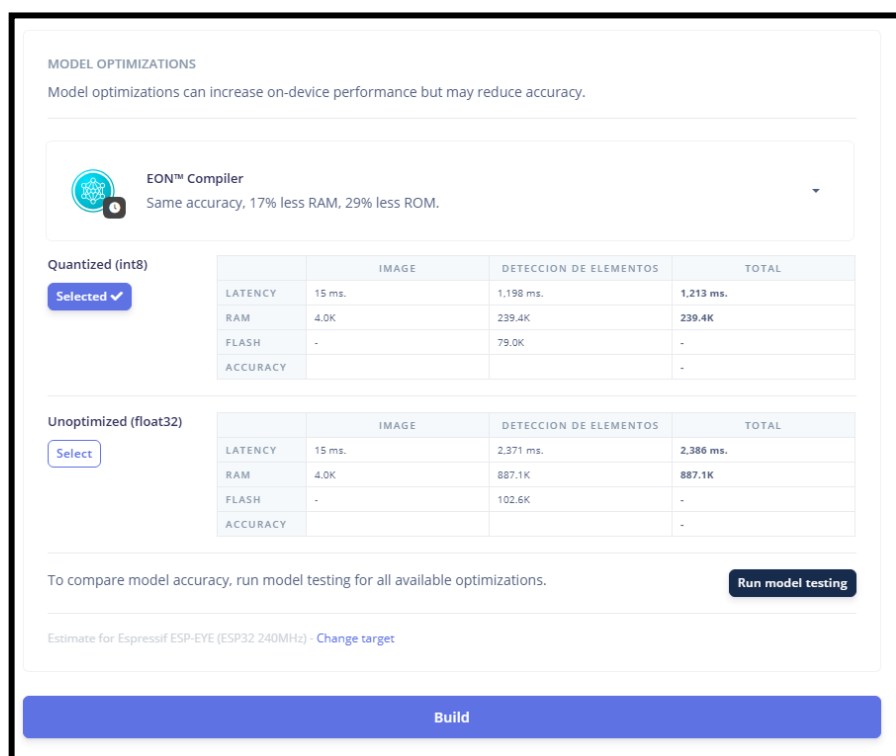
Una vez haber realizado correctamente el entrenamiento es hora de crear la librería que subiremos a nuestro ESP32-CAM para que pueda realizar la detección de los objetos en tiempo real.

Construir la librería para el ESP32-CAM

En el panel lateral izquierdo iremos a la opción de deployment.



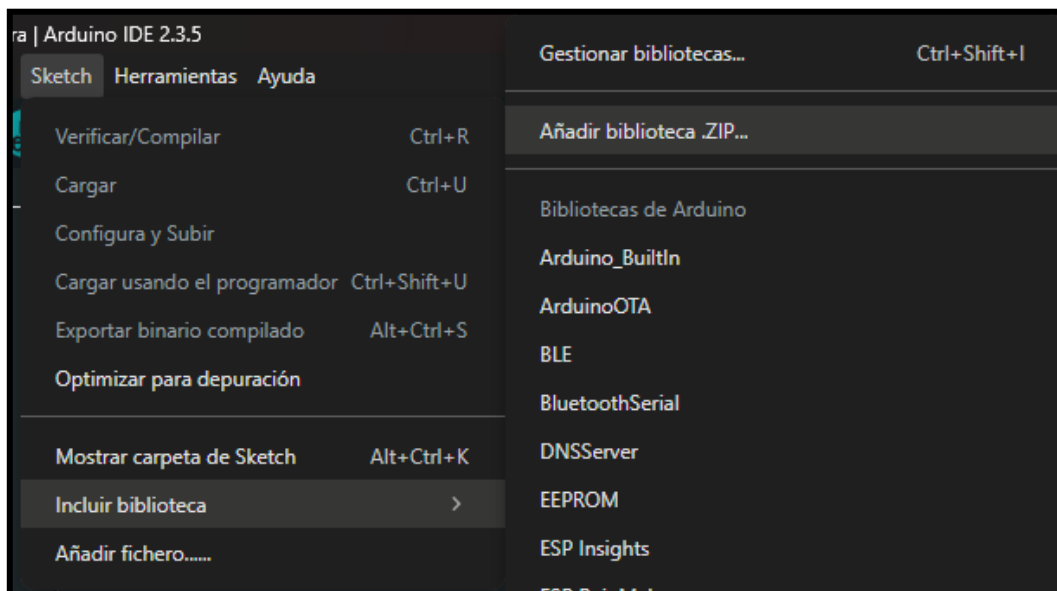
Y allí seleccionaremos la opción de Arduino y daremos a la opción de build.



Luego de esto aparecerá un mensaje de que se ha descargado satisfactoriamente tu librería.

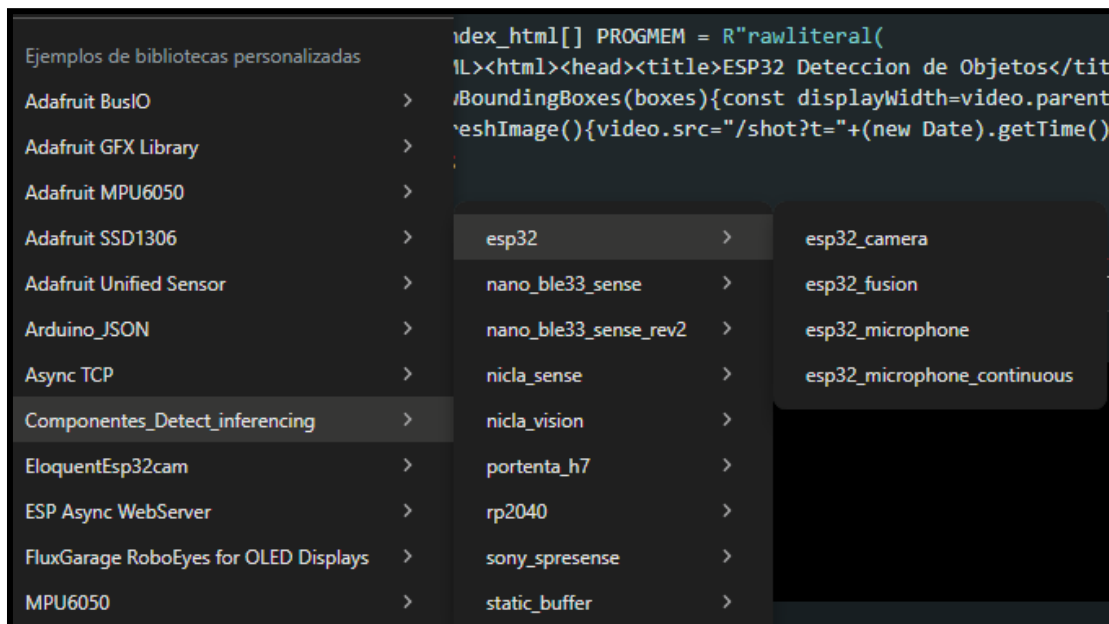


Una vez hecho esto subiremos la librería a nuestro entorno de Arduino por medio del gestor de librerías con la opción “agregar librería .ZIP”



Probar el resultado del entrenamiento.

Para comprobar el resultado del entrenamiento debemos subir el código que se encuentra como ejemplo en la librería que hemos subido que lo encontramos de la siguiente forma:




Escogemos ESP32_camera, y una vez abierto el código comentaremos la línea 35 y quitamos la línea 36 como comentario de la siguiente manera:

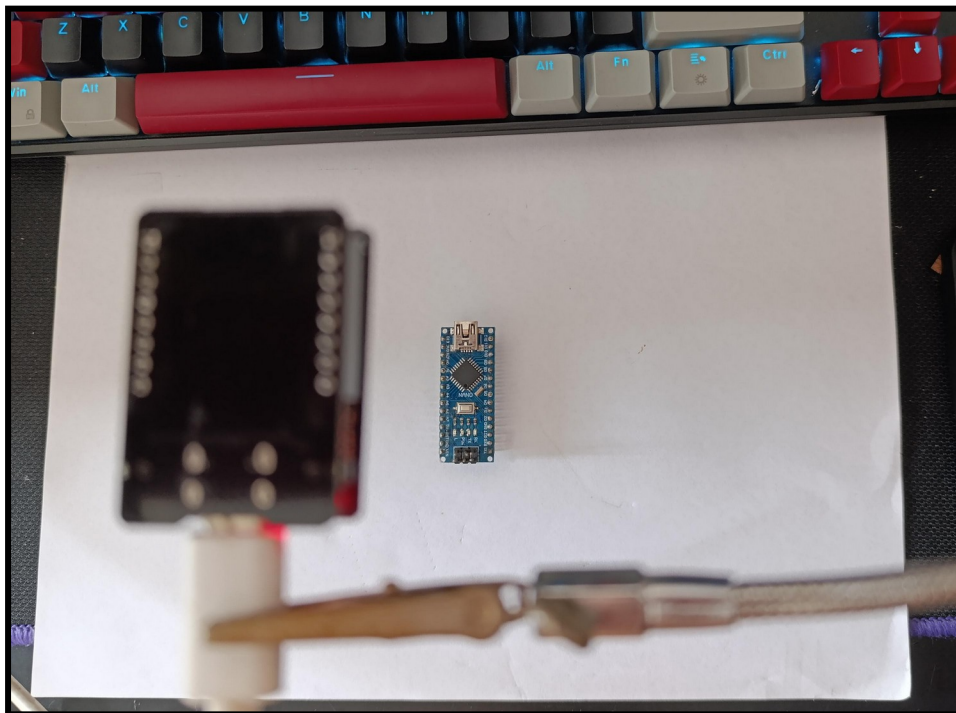
```
32 // Select camera model - find more camera models i
33 // https://github.com/espressif/arduino-esp32/blob
34
35 #define CAMERA_MODEL_ESP_EYE // Has PSRAM
36 // #define CAMERA_MODEL_AI_THINKER // Has PSRAM
37
```



```
32 // Select camera model - find more camera model
33 // https://github.com/espressif/arduino-esp32/blob
34
35 // #define CAMERA_MODEL_ESP_EYE // Has PSRAM
36 #define CAMERA_MODEL_AI_THINKER // Has PSRAM
37
```

Una vez realizado este cambio ya podemos subir el código a nuestro ESP32-CAM y posicionarla para realizar la detección, abriremos el monitor serial para ver los resultados.

 Carga completada.



Abrimos el monitor serial para ver el resultado

```
Object detection bounding boxes:
  NANO (0.746094) [ x: 40, y: 40, width: 8, height: 16 ]
Predictions (DSP: 16 ms., Classification: 720 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
  NANO (0.792969) [ x: 40, y: 40, width: 8, height: 16 ]
Predictions (DSP: 16 ms., Classification: 720 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
  NANO (0.773438) [ x: 40, y: 40, width: 16, height: 8 ]
```

Como podemos observar hemos realizar la detección de forma correcta.

¡EXTRA!

En **CODELAB** hemos realizado un código que puedes subir a tu ESP32-CAM una vez ya está instalada tu librería, este te permitirá conectarte a tu red wifi y ver la cámara y los resultados en tiempo real

```
10 // --- Configuración WiFi ---  
11 const char* ssid = " ";  
12 const char* password = " ";  
13
```

Debes modificar los datos de tu red WIFI en la línea 11 y 12, y proceder a subir tu código, ya debe estar instalada tu librería y verificar que el nombre de esta sea correcto.

```
2 #include <Componentes_Detect_inferencing.h> // Asegúrate que este nombre de archivo sea correcto
```

Una vez hecho esto abrirás el monitor serial y copiaras la dirección, para abrirla en tu navegador web y ya solo queda probar!

¡EXTRA!

En **CODELAB** hemos realizado un código que puedes subir a tu ESP32-CAM una vez ya está instalada tu librería, este te permitirá conectarte a tu red wifi y ver la cámara y los resultados en tiempo real

```
10 // --- Configuración WiFi ---
11 const char* ssid = " ";
12 const char* password = " ";
13
```

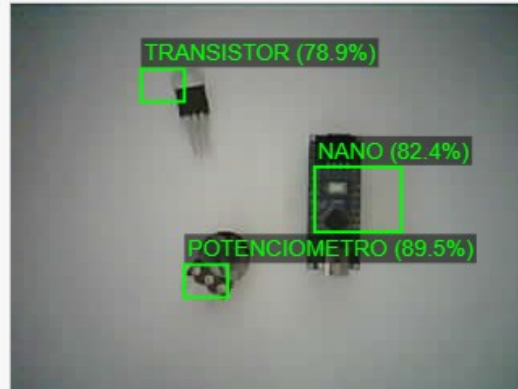
Debes modificar los datos de tu red WIFI en la línea 11 y 12, y proceder a subir tu código, ya debe estar instalada tu librería y verificar que el nombre de esta sea correcto.

```
2 #include <Componentes_Detect_inferencing.h> // Asegúrate que este nombre de archivo sea correcto
```

Una vez hecho esto abrirás el monitor serial y copiaras la dirección, para abrirla en tu navegador web y ya solo queda probar!

CODELAB

ESP32 Deteccion de Objetos



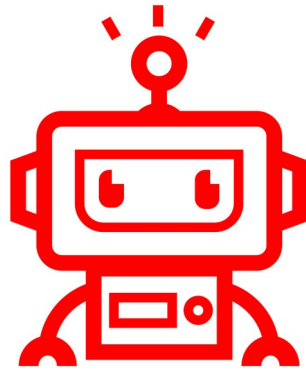
La imagen se refresca. Las detecciones se actualizan en tiempo real.

Te invitamos a seguirnos y apoyarnos para poder seguir desarrollando estos proyectos!

Visitanos o contactanos:

- roboticacodelab.com - Sitio Web Principal
- roboticacodelab.online - Sitio Web Educativo
- www.instagram.com/robotica_codelab - Instagram
- wa.me/573153338069 - WhatsApp
- RoboticaCodelab@gmail.com - Correo Electronico

Con esta programación podemos observar en tiempo real la detección que realiza el ESP32-CAM.



Esta es una guía desarrollada por **CODELAB** dedicada a estudiantes y maestros entusiastas por la electrónica y robótica, somos un club enfocado en el aprendizaje de nuevas tecnologías y programación para preparar los profesionales del futuro! Puedes contactarnos por los siguientes medios:



315 333 8069



RoboticaCodelab@gmail.com



Robotica Codelab

Agradecer a las Unidades Tecnológicas de Santander por el apoyo brindado y por dar la oportunidad de fomentar estas nuevas tecnologías a mas estudiantes y maestros.



Unidades
Tecnológicas
de Santander