INFORME PROCESO CON ESP32-CAM Y EDGE IMPULSE

1. Instalación del Entorno de Desarrollo ESP32 en Arduino IDE

Para programar la **ESP32-CAM**, primero se configuró el **Arduino IDE** para soportar las placas ESP32. Se siguieron los siguientes pasos:

Agregar la URL JSON de las tarjetas ESP32

1. Se abrió el **Arduino IDE** y se accedió a **Preferencias**.
2. En el campo **"Gestor de URLs Adicionales de Tarjetas"**, se agregó la siguiente URL: https://dl.espressif.com/dl/package_esp32_index.json

3. Se guardaron los cambios y se cerró la ventana.

Instalación de las Tarjetas ESP32

1. Se abrió el **Gestor de Tarjetas** en **Herramientas → Placa → Gestor de Tarjetas**.
2. Se buscó **ESP32 by Espressif Systems** y se instaló la última versión disponible.

2. Conexión de la ESP32-CAM al Módulo FTDI para la Programación

Como la **ESP32-CAM** no tiene un puerto USB incorporado, se utilizó un **módulo FTDI** para conectarlo a la computadora.

Conexión de Pines

ESP32-CAM	Módulo FTDI
5V	**5V**
GND	**GND**
U0R (GPIO3)	**TX**
U0T (GPIO1)	**RX**
IO0	**GND** *(Para entrar en modo programación)*

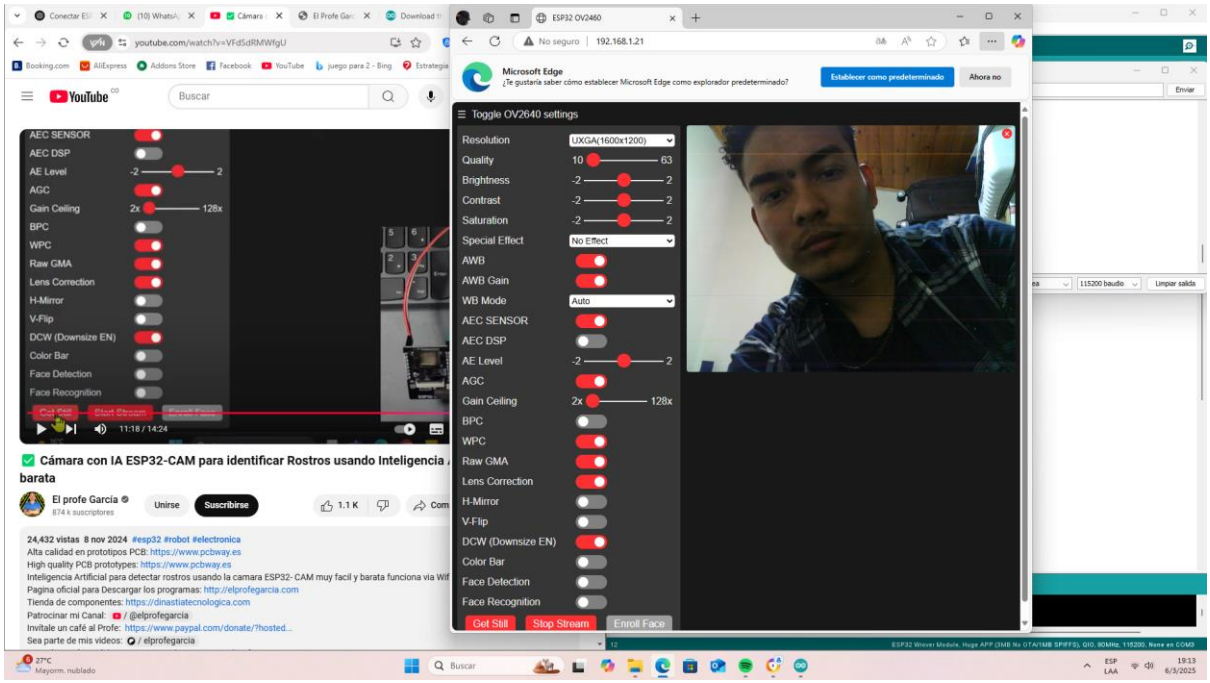
- Se aseguró que el módulo FTDI estuviera configurado en **5V** y no en 3.3V para un mejor funcionamiento de la ESP32-CAM.

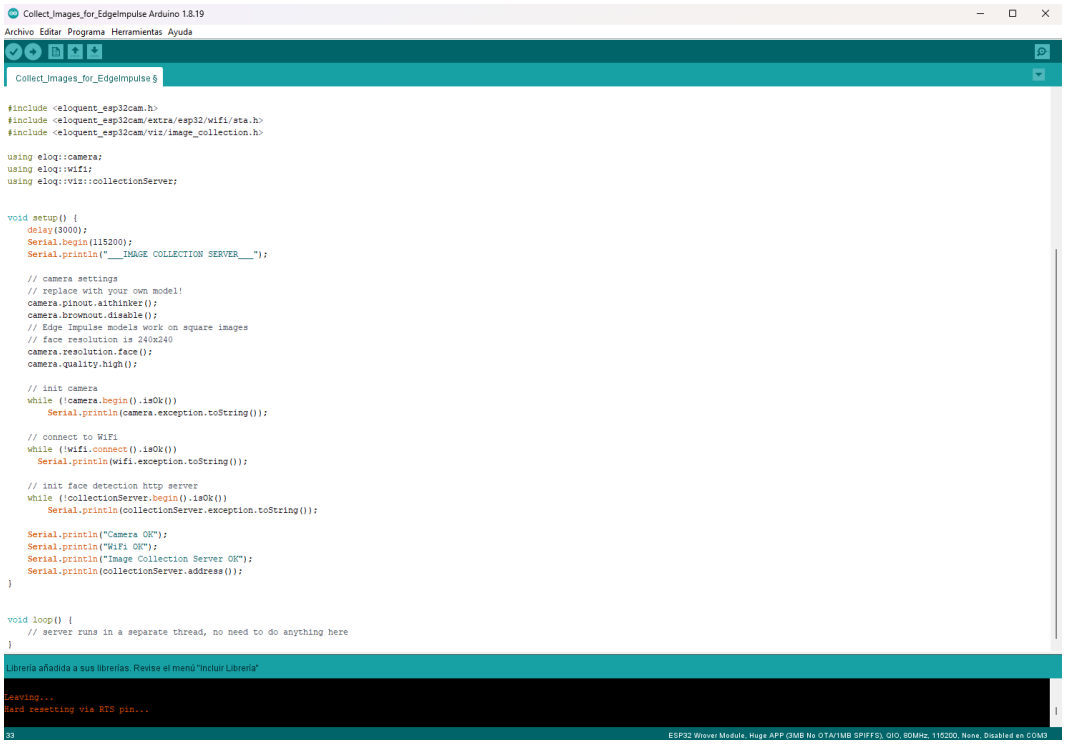
3. Prueba Inicial de la Cámara ESP32-CAM

1. Se utilizó un código de prueba en **Arduino IDE** para verificar que la cámara OV2640 estuviera funcionando correctamente.
2. Se cargó el código y se abrió el **Monitor Serie**.

3. Se verificó que la ESP32-CAM se conectara al Wi-Fi y mostrara la URL de acceso.

4. Se intentó acceder a la transmisión de video mediante: http://192.168.1.21
   Si la imagen se mostraba correctamente, se validó que la cámara estaba funcionando.



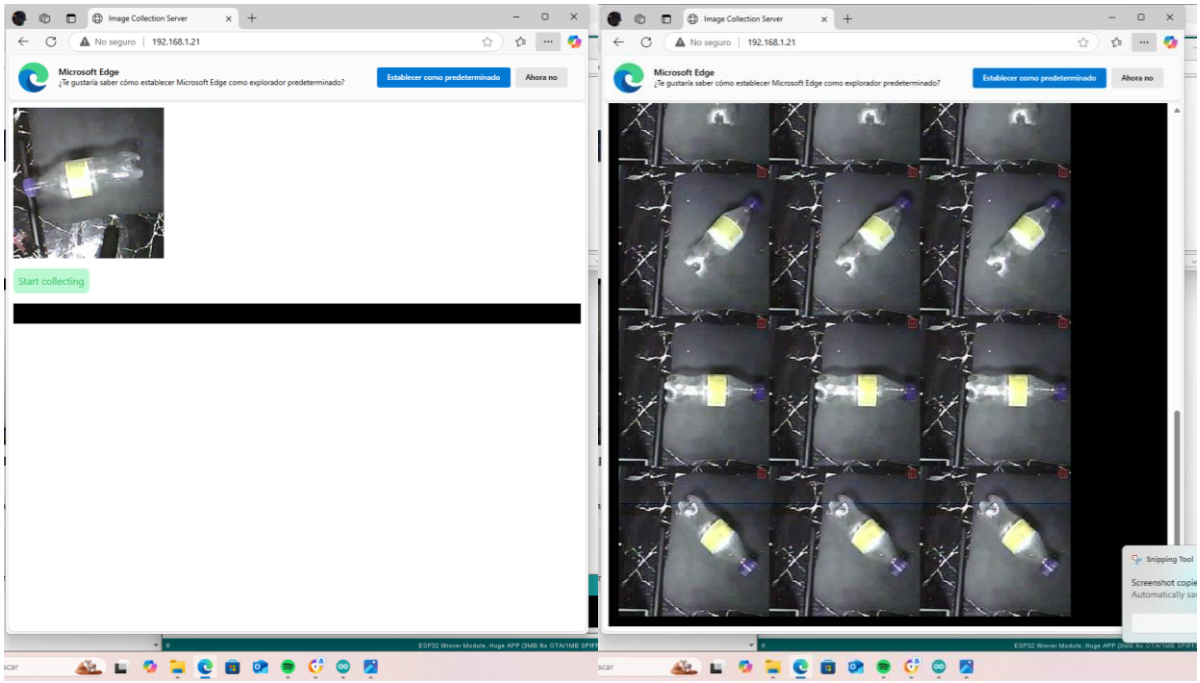


4. Instalación de EloquentEsp32 para Captura de Imágenes

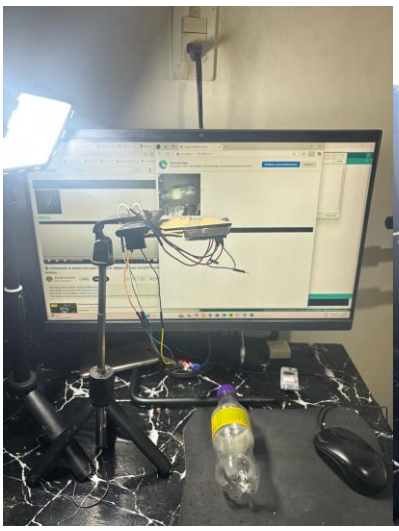
Para capturar imágenes con la **ESP32-CAM**, se utilizó la librería **EloquentEsp32**.

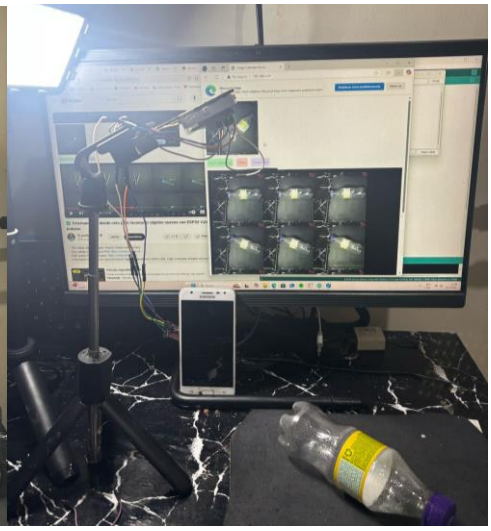
1. Se instaló la librería desde **Arduino IDE** → **Gestor de Bibliotecas** buscando: EloquentEsp32

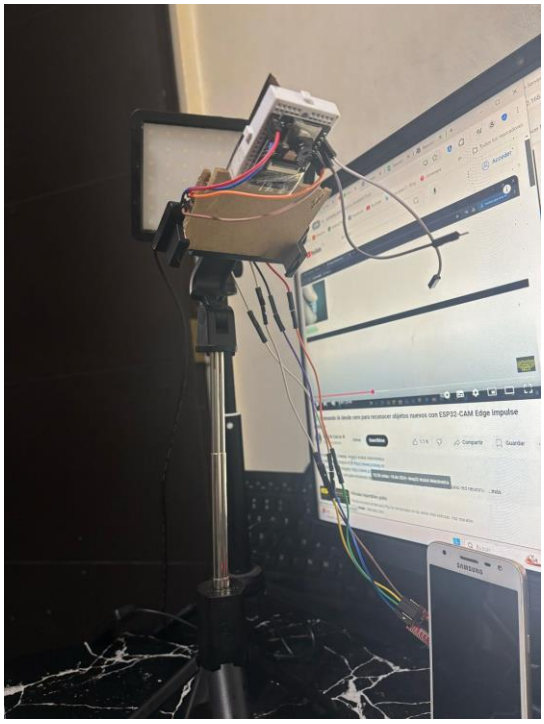
2. Se cargó un código que permitía capturar imágenes y guardarlas en la memoria para su posterior uso en **Edge Impulse**.
3. Se usó el comando **collect_image_for_edge_impulse()** para tomar múltiples fotografías de la botella desde distintos ángulos y condiciones de luz.

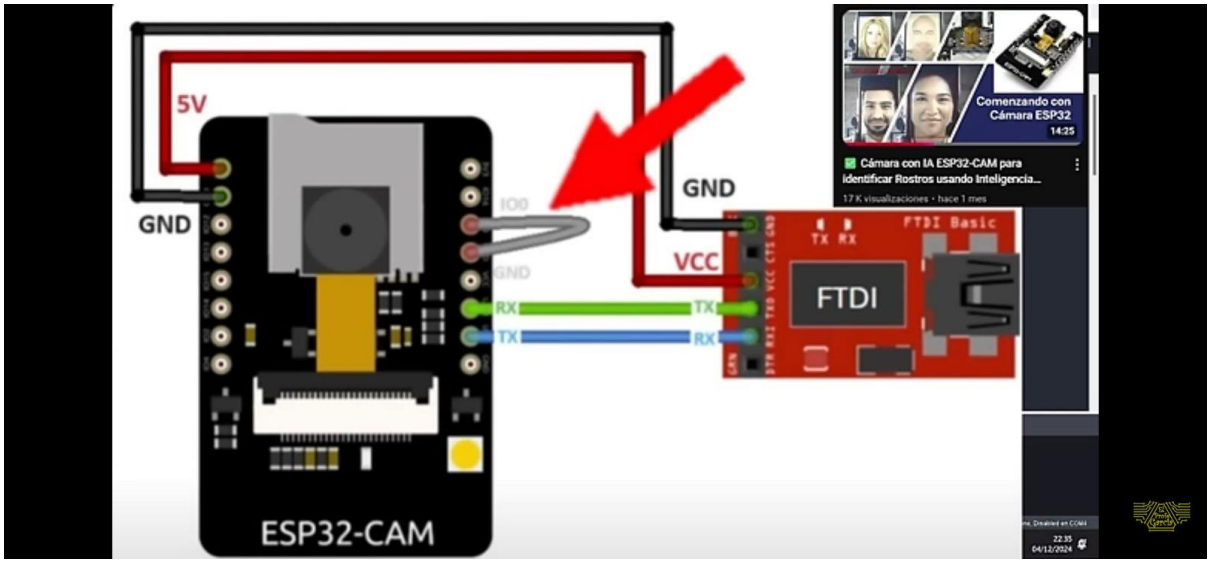


Montaje








5V
GND
IO0
GND
VCC
RX
TX
TX
RX
FTDI
FTDI Basic
ESP32-CAM
Comenzando con Cámara ESP32
14:25
Cámara con IA ESP32-CAM para identificar Rostros usando Inteligencia...
17 K visualizaciones • hace 1 mes

5. Carga de Imágenes en Edge Impulse y Entrenamiento del Modelo

Subida de Imágenes a Edge Impulse

1. Se accedió a la plataforma **Edge Impulse** (https://www.edgeimpulse.com).
2. Se creó un nuevo proyecto para la clasificación de imágenes.
3. Se subieron las imágenes capturadas con la ESP32-CAM.
4. Se etiquetaron correctamente las imágenes como **"Botella"** o **"No Botella"**.

Entrenamiento del Modelo

1. Se seleccionó la opción de **Clasificación de imágenes** dentro del Dashboard.
2. Se entrenó un modelo de inteligencia artificial basado en **redes neuronales convolucionales (CNN)**.
3. Se ajustaron parámetros como la cantidad de épocas y el tamaño del lote para mejorar la precisión del modelo.

Prueba del Modelo

1. Se validó el modelo con imágenes nuevas para comprobar su precisión.
2. Se revisaron las métricas de precisión y recall para asegurar un buen rendimiento.

6. Exportación del Modelo e Implementación en la ESP32-CAM

Exportación de la Librería para Arduino

1. Se seleccionó la opción **"Deployment"** en Edge Impulse.
2. Se eligió la opción **Arduino Library** para exportar el modelo en un formato compatible con la ESP32-CAM.
3. Se descargó la librería en formato ZIP y se agregó a **Arduino IDE**.

Carga del Código Final en la ESP32-CAM

1. Se cargó un código en la ESP32-CAM que utilizaba la librería exportada para procesar imágenes en tiempo real.
2. Se utilizó el modelo entrenado para reconocer si en la imagen capturada había una **botella** o no.

3. Se verificó el resultado desde el Monitor Serie o enviando los datos a una plataforma web.

Conclusión

A través de este proceso, se logró configurar y programar la **ESP32-CAM** para capturar imágenes, procesarlas con **Edge Impulse**, y entrenar un modelo de IA que permitió reconocer imágenes de una botella. Finalmente, se implementó el modelo en la ESP32-CAM, logrando el reconocimiento en tiempo real.