INSTITUCIÒN: Instituto Tecnológico Superior Del Occidente Del Estado De Hidalgo

tecnologías De La Información Y Comunicaciones

REPORTE TECNICO: Prototipo ESPCAM

Garcia Azpeitia Gael

Salas Rodríguez Luis Francisco

Introducción a las Tics

Siclo:2025-2029

DOCENTE: Saul Isaí Soto Ortiz

LUGAR: Laboratorio de cisco FECHA: 28/10/2025

INDICE

RESUMEN

En resumen este proyecto tiene como objetivo desarrollar y demostrar la viabilidad de un sistema de detección de objetos compacto y económico utilizando la placa ESP-CAM **(**ESP32**-**CAM**)**. El trabajo pondrá a prueba los conocimientos de la materia mediante la implementación de este sistema.

Entrenamiento del Modelo**:** Se utilizará la plataforma Edge Impulse para entrenar la ESP-CAM, permitiéndole detectar y clasificar objetos específicos.

Integración Eléctrica y Funcional**:** El sistema integra componentes de **eléctrica/circuitos** para reaccionar ante el reconocimiento. Al detectar un objeto, la ESP-CAM debe encenderse

INTRODUCCION

En este trabajo pondremos a prueba todo lo aprendido en todos los módulos anteriores de la materia, por medio de una ESPCAM y el software llamado Arduino nos dimos la tarea de entrenar la ESPCAM para detectar ciertos objetos los cuales entrenamos en una página llamada Edge impulse, explicaremos paso a paso como realizamos eso y además implementaremos temas de eléctrica o circuitos ya que además de detectar objetos la ESPCAM también debe de encender, mostrar el mensaje de reconocimiento en la pantalla oled que vamos a conectar a la ESPCAM y opcionalmente agregar un buser.

A continuación, también explicaremos de manera elabora el proceso de cargar las librerías ya que este trabajo no puede ser tan fácil ya que en algunos casos el programa Arduino no funciona como debe y por ello debemos de hacer cambios o incluso cambiar la versión descargada según sea el tipo.

El presente trabajo aborda el desarrollo de un sistema de detección de objetos de bajo costo y fácil implementación, utilizando la placa ESP-CAM (ESP32-CAM) en conjunto con la plataforma de desarrollo Arduino. En el panorama actual, la visión artificial y la detección de objetos se han convertido en pilares fundamentales para una amplia gama de aplicaciones, que van desde la seguridad y la vigilancia hasta la automatización industrial y los sistemas domésticos inteligentes.

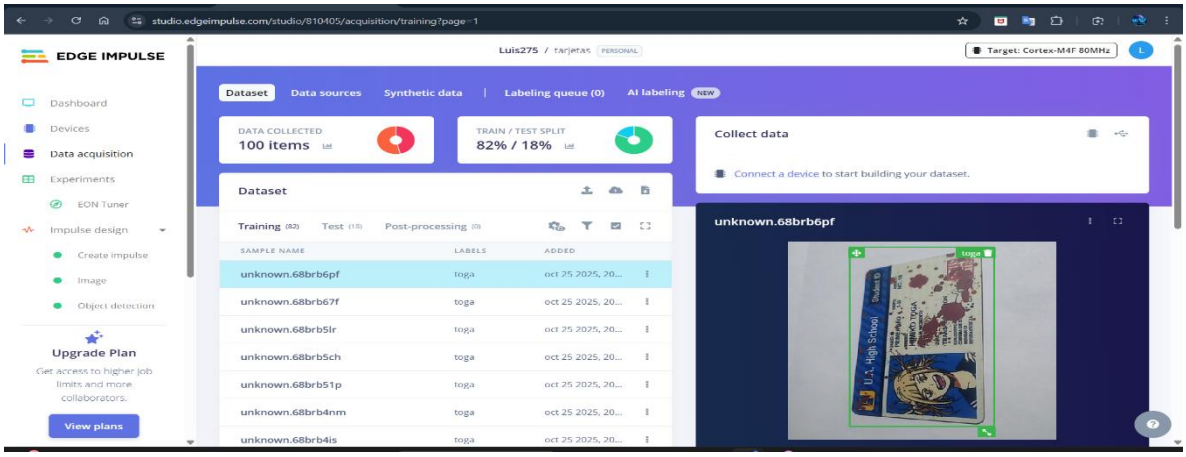
El objetivo principal de este proyecto es explorar y demostrar la viabilidad de utilizar esta combinación de hardware para identificar y clasificar objetos en tiempo real o casi real. Esto se logrará mediante la implementación de algoritmos de visión sencillos o modelos de aprendizaje automático optimizados, aprovechando las capacidades de conectividad Wi-Fi de la ESP32 para la transmisión de datos o notificaciones. Se espera que los resultados obtenidos sirvan como base para futuros proyectos de automatización y monitoreo que requieran una solución compacta y económica.

INSTALACION Y VERIFICACION DE ESPCAM

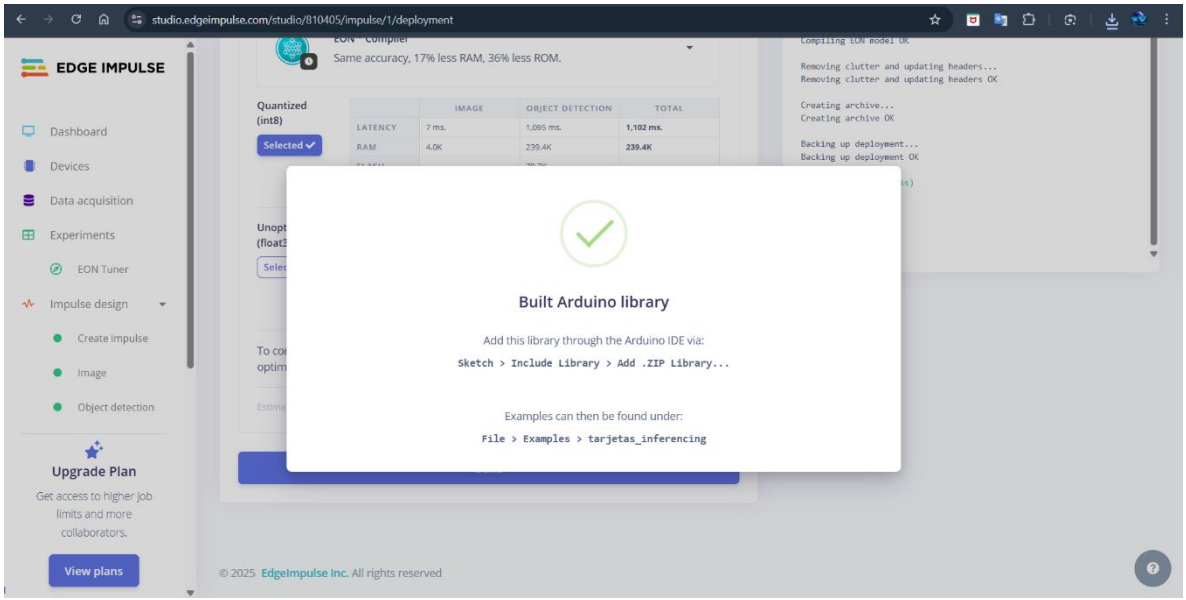
No fue una instalación como tal de la ESPCAM sino una instalación de Arduino, para instalarlo que hicimos fue ingresar al link que nos proporcionó el profesor en sesiones anteriores, aunque también podíamos buscarlo directamente en algún navegador, la instalamos y para verificar el buen funcionamiento con la ESPCAM, lo que hicimos fue crear un ejemplo de cámara web en la aplicación Arduino, conectando la ESPCAM y entrenando un objeto. Una manera rápida de verificar que si nos funciones es conectando la ESPCAM y buscar si detecta el puerto a el que esta conectado, ejemplo, tools – port – com 5.

GENERACION DE LALIBRERIA EN EDGE IMPULSE

Generar una librería es un paso muy importante ya que contiene los objetos que se entrenaron para detectar y para ello lo que hicimos fue acceder a la pagina Edge impulse y registrarnos, creamos un nuevo proyecto pero antes tomamos varias fotografías a un objeto, la fotografías deben de ser al menos 50 y tomadas de diferentes ángulos del objeto para que la ESPCAM lo pueda reconocer fácilmente, después las guardamos todas juntas en una carpeta, una ves echo eso regresamos a Edge impulse donde subimos la carpeta con las fotos y las recortamos para que se enfocara mas en el objeto, después creamos un un impulso para un objeto y nos dirigimos a la parte donde se entrena, dimos clic en sabe y train, ya que ahí nos debe de marcar un porcentaje importante del reconocimiento del objeto.



Recolección de datos y etiquetado por objeto diferente



Creación de la librería por medio de Edge impulse

INSTALACION DE LA LIBRERÍA PARA LA ESPCAM

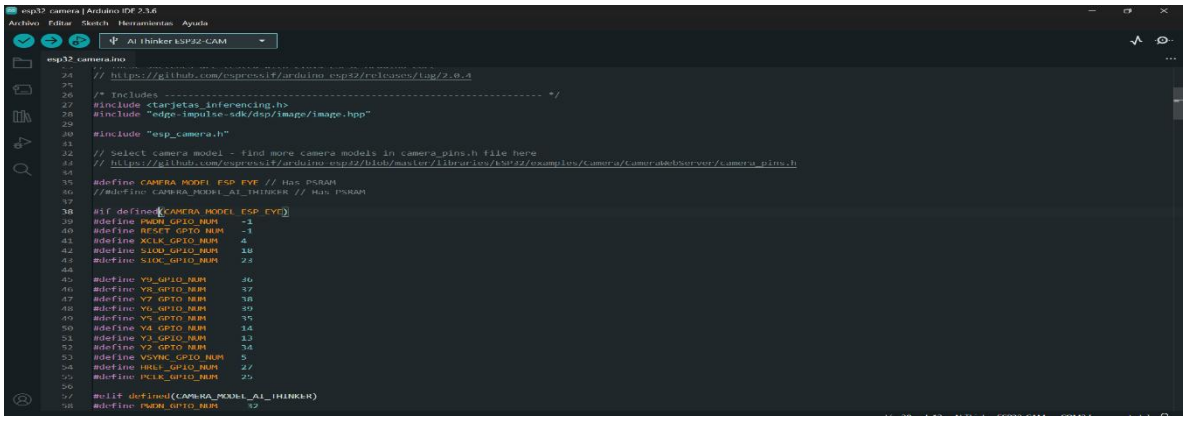
Para instalar la librería abrimos el Arduino en donde ya conectada la ESPCAM a la laptop y que haya reconocido el puerto, lo siguiente es ir al apartado de sketch después a incluir librería y en add zip library seleccionamos el archivo zip del objeto que entrenamos en Edge impulse, y es necesario que el archivo si este guardado en archivo zip ya que si no, no podremos incluirla en el ejemplo Arduino para la ESPCAM

Después de que realizamos lo anterior pudimos observar el apartado inferior de sketch que al añadir una librería efectivamente nos aparecía el nombre de nuestra librería con los objetos que entrenamos, ya solo fue necesario agregarla para que en nuestro código especificara que de ahí se tomarían las imágenes ya entrenadas, nosotros observamos que al principio de nuestro código ya estaba incluyendo la librería.

DEMOSTRACION DEL FUNCIONAMIENTOCON LA LIBRERÍA

Para verificar el funcionamiento lo que realizamos fue una vez subida la librería, creamos un ejemplo de cara web con esta misma para que incluya el objeto entrenado

y la subimos a la ESPCAM, este proceso puede tardar un poco ya que pasa todo el código de ejemplo a un lenguaje que la ESPCAM pueda leer, y ya que se logro eso nos aseguramos que estuviera enfocando al objeto.



Código de la librería punto zip

VISUALIZACION EN EL DISPLAY Y LEDS

Posteriormente para poder visualizar en el desplaye el mensaje de reconocimiento para cada objeto, utilizamos un código de ejemplo distinto en el cual agregábamos el mensaje de reconocimiento y librerías nuevas como la de ADAFRUIT GFX, sin embargo también fue muy importante la parte física ya que agregamos una desplaye oled de 3.3v para visualizar dichos mensajes, lo que realizamos fue alimentar el desplaye con ayuda de dos pines de la ESPCAM, el pin de GND y el de alimentación (3,3v) también hicimos la conexión de los pines de comunicación los cuales enviarían la información de la ESPCAM, todas estas conexiones la hicimos soldando los cables con ayuda de un cautín y estaño para soldar.

Visualización de prototipo final los resultados de la detección de objetos son mandados a la pantalla oled

instalación y verificación de ESPCAM en Windows.

Como resultado de este apartado, obtuvimos como respuesta que la ESPCAM con la que vamos a trabajar esta funcionando correctamente, pudimos visualizar en pantalla que efectivamente introduciendo la IP de la ESPCAM que conectamos anteriormente a una zona de cobertura wi-fi, en un navegador nos pudo mostrar la imagen en pantalla, esto significaba que nuestra ESPCAM no presento fallas y nos funcionaria correctamente para la detección de objetos, esto fue necesario ya que la cámara que incluye juega un rol muy importante en esta practica.

Generación de librería de Edge Impulse

ya que se realizó correctamente todo lo anterior, obtuvimos como resultado por parte de Edge impulse un archivo de librería zip, para saber que el resultado de esto sea efectivo en Edge impulse después de entrenar todas nuestra imágenes pudimos visualizar un porcentaje de entrenamiento o reconocimiento, mientras mas fotos de buena calidad y de varias posiciones hallamos agregado mejor seria el porcentaje resultante , nosotros obtuvimos un resultado de porcentaje del 96 y el98 porciento de reconocimiento lo cual resulto bueno ya que mientras mayor sea el porcentaje, mas fácil y seguro será el reconocimiento por parte de la ESPCAM.

Instalación de librería en ESPCAM

Después de hacer todo el proceso para la demostración nosotros le dimos en la opción de validar el código para asegurarnos de que no tenga errores antes de subirlo a la ESPCAM, ya que lo hicimos subimos el código a la ESP y en la parte de abajo nos empezó a mostrar varios mensajes simultáneamente indicándonos de que no reconocía nada hasta que acercáramos el objeto de dicha librería lo cual probaba que la librería no solo se había echo correctamente sino que en el código la librería se había ejecutado correctamente.

Visualización de la etiqueta de identificación de display OLED

Después de que hicimos correctamente el procedimiento de instalación del desplaye en la ESPCAM, a la hora de subir el código, no solo nos mostraba los mensajes de reconocimiento en la pantalla de la computadora si no que ahora también podíamos visualizar dichos mensajes de reconocimiento en la pantalla oled, ahí nos mostró un mensaje de reconocimiento según fuera el objeto entrenado que le mostráramos y si no le mostrábamos ningún objeto de la librería entonces ahí mismo nos avisaba sobre dicho caso.

Una vez haciendo la conexión de los leds a GND por parte de la patita pequeña conectamos los anodos a los GPIO 12,13, y 2 soldándolo a la placa donde agarrarían corriente y al reconocer nuestro objeto numero 1 lo marco bien sin problema así mismo el dos y el led cuando el objeto no está viendo nada siendo este de color azul, el objeto 2 color verde y el objeto 2 color rojo. La prueba fue echa y todo fue echo correctamente el resultado se mostró en la pantalla y los led cambiaron conforme a su objeto.

Código en Arduino IDE.

```
// Conexión del módulo OLED
//Pin GND-GND
//5V-VCC
//14-SCL
//15-SDA

#include <tarjetas_inferencing.h>      // Librería del modelo de Edge Impulse en nuestro caso se llama tarjetas
#include "edge-impulse-sdk/dsp/image/image.hpp"
#include "esp_camera.h"          // Librería para la cámara ESP32-CAM

#define CAMERA_MODEL_AI_THINKER      // Modelo de cámara (versión AI Thinker)

// --- Configuración de pines del ESP32-CAM (modelo AI Thinker) ---
#if 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
#define HREF_GPIO_NUM 23
#define PCLK_GPIO_NUM 22
#else
#error "Camera model not selected"
#endif


// Tamaño original del frame de la cámara
#define EI_CAMERA_RAW_FRAME_BUFFER_COLS 320
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS 240
#define EI_CAMERA_FRAME_BYTE_SIZE 3


// Librerías para el display OLED
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

```
// Pines personalizados para la pantalla oled SDA Y SCL
#define I2C_SDA 15
#define I2C_SCL 14
TwoWire I2Cbus = TwoWire(0);

// Tamaño de pantalla OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &I2Cbus, OLED_RESET);

// Pines de los LEDs indicador
#define LED_AZUL 12
#define LED_VERDE 13
#define LED_ROJO 2

static bool debug_nn = false;
static bool is_initialised = false;
uint8_t *snapshot_buf;  // lugar donde se guardará la imagen capturada

// Configuración de la cámara
static camera_config_t camera_config = {
 .pin_pwdn = PWDN_GPIO_NUM,
```

```
.pin_reset = RESET_GPIO_NUM,
.pin_xclk = XCLK_GPIO_NUM,
.pin_sscb_sda = SIOD_GPIO_NUM,
.pin_sscb_scl = SIOC_GPIO_NUM,

// Pines de la señal de imagen
.pin_d7 = Y9_GPIO_NUM,
.pin_d6 = Y8_GPIO_NUM,
.pin_d5 = Y7_GPIO_NUM,
.pin_d4 = Y6_GPIO_NUM,
.pin_d3 = Y5_GPIO_NUM,
.pin_d2 = Y4_GPIO_NUM,
.pin_d1 = Y3_GPIO_NUM,
.pin_d0 = Y2_GPIO_NUM,

.pin_vsync = VSYNC_GPIO_NUM,
.pin_href = HREF_GPIO_NUM,
.pin_pclk = PCLK_GPIO_NUM,

.xclk_freq_hz = 20000000,      // Reloj de la cámara
.ledc_timer = LEDC_TIMER_0,
.ledc_channel = LEDC_CHANNEL_0,

.pixel_format = PIXFORMAT_JPEG, // Formato JPEG
.frame_size = FRAMESIZE_QVGA,   // 320x240
.jpeg_quality = 12,
```

```
  .fb_count = 1,
  .fb_location = CAMERA_FB_IN_PSRAM,
  .grab_mode = CAMERA_GRAB_WHEN_EMPTY,
};

// Declaración de funciones
bool ei_camera_init(void);
void ei_camera_deinit(void);
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf);

void setup() {

  Serial.begin(115200);

  // Inicializar bus I2C para el OLED
  I2Cbus.begin(I2C_SDA, I2C_SCL, 100000);

  // Inicializar pantalla OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.printf("SSD1306 OLED display failed to initialize.\n");
    while (true);
  }

  // Configuración de LEDs
  pinMode(LED_AZUL, OUTPUT);
```

```
pinMode(LED_VERDE, OUTPUT);
pinMode(LED_ROJO, OUTPUT);

// LED azul encendido = sistema listo pero sin detección
digitalWrite(LED_AZUL, HIGH);
digitalWrite(LED_VERDE, LOW);
digitalWrite(LED_ROJO, LOW);

Serial.println("Edge Impulse Inferencing Demo");

// Inicializar cámara
if (ei_camera_init() == false) {
  ei_printf("Failed to initialize Camera!\r\n");
} else {
  ei_printf("Camera initialized\r\n");
}

// Mensaje de inicio en pantalla
display.clearDisplay();
display.setCursor(0, 0);
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.print("Starting continious\n inference in\n 2 seconds...");
display.display();
ei_sleep(2000);
display.clearDisplay();
```

```
}

void loop() {

 display.clearDisplay();

 // Pequeña pausa
 if (ei_sleep(5) != EI_IMPULSE_OK) return;

 // Reservar memoria para la imagen capturada
 snapshot_buf = (uint8_t *)malloc(EI_CAMERA_RAW_FRAME_BUFFER_COLS *
                EI_CAMERA_RAW_FRAME_BUFFER_ROWS *
                EI_CAMERA_FRAME_BYTE_SIZE);

 if (snapshot_buf == nullptr) {
  ei_printf("ERR: Failed to allocate snapshot buffer!\n");
  return;
 }

 // Estructura requerida por Edge Impulse
 ei::signal_t signal;
 signal.total_length = EI_CLASSIFIER_INPUT_WIDTH *
EI_CLASSIFIER_INPUT_HEIGHT;
 signal.get_data = &ei_camera_get_data;

 // Capturar imagen
```

```
 if (ei_camera_capture(EI_CLASSIFIER_INPUT_WIDTH,
                       EI_CLASSIFIER_INPUT_HEIGHT,
                       snapshot_buf) == false) {
  ei_printf("Failed to capture image\r\n");
  free(snapshot_buf);
  return;
 }

 ei_impulse_result_t result = { 0 };

 // Ejecutar el modelo
 EI_IMPULSE_ERROR err = run_classifier(&signal, &result, debug_nn);

 if (err != EI_IMPULSE_OK) {
  ei_printf("ERR: Failed to run classifier (%d)\n", err);
  return;
 }

 // Mostrar tiempos de procesamiento
 ei_printf("Predictions (DSP: %d ms., Classification: %d ms., Anomaly: %d ms.):
\n",
        result.timing.dsp, result.timing.classification, result.timing.anomaly);

#if EI_CLASSIFIER_OBJECT_DETECTION == 1
 bool bb_found = result.bounding_boxes[0].value > 0;
 bool detecto_lufy = false;
```

```
bool detecto_toga = false;


// Recorrer todos los objetos detectados
for (size_t ix = 0; ix < result.bounding_boxes_count; ix++) {
 auto bb = result.bounding_boxes[ix];
 if (bb.value == 0) continue;


 // Mostrar etiqueta y porcentaje
 display.setCursor(0, 20 * ix);
 display.setTextSize(2);
 display.setTextColor(SSD1306_WHITE);
 display.print(bb.label);
 display.print("-");
 display.print(int(bb.value * 100));
 display.print("%");
 display.display();


 // Bandera para LEDs
 if (strcmp(bb.label, "lufy") == 0) detecto_lufy = true;
 if (strcmp(bb.label, "toga") == 0) detecto_toga = true;
}


// Control de LEDs según el objeto detectado
if (bb_found) {
 if (detecto_lufy) {
  digitalWrite(LED_AZUL, LOW);
```

```
    digitalWrite(LED_VERDE, HIGH);
    digitalWrite(LED_ROJO, LOW);
   }
   else if (detecto_toga) {
    digitalWrite(LED_AZUL, LOW);
    digitalWrite(LED_VERDE, LOW);
    digitalWrite(LED_ROJO, HIGH);
   }
   else {
    digitalWrite(LED_AZUL, HIGH); // Otro objeto
    digitalWrite(LED_VERDE, LOW);
    digitalWrite(LED_ROJO, LOW);
   }
  } else {
   // Nada detectado
   display.setCursor(0, 20);
   display.setTextSize(2);
   display.print("No detecto");
   display.display();

   digitalWrite(LED_AZUL, HIGH);
   digitalWrite(LED_VERDE, LOW);
   digitalWrite(LED_ROJO, LOW);
  }
#endif
```

```
 free(snapshot_buf);  // Liberar memoria
}

// --- Inicialización de la cámara ---
bool ei_camera_init(void) {
 if (is_initialised) return true;

 esp_err_t err = esp_camera_init(&camera_config);

 if (err != ESP_OK) {
  Serial.printf("Camera init failed with error 0x%x\n", err);
  return false;
 }

 // Configuración extra para el sensor OV3660
 sensor_t *s = esp_camera_sensor_get();
 if (s->id.PID == OV3660_PID) {
  s->set_vflip(s, 1);     // Voltear vertical
  s->set_brightness(s, 1); // Aumentar brillo
  s->set_saturation(s, 0);
 }

 is_initialised = true;
 return true;
}
```

```
// --- Desinicializar cámara ---
void ei_camera_deinit(void) {
 esp_err_t err = esp_camera_deinit();
 if (err != ESP_OK) {
  ei_printf("Camera deinit failed\n");
  return;
 }
 is_initialised = false;
}

// --- Captura de imagen ---
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) {

 if (!is_initialised) {
  ei_printf("ERR: Camera is not initialized\r\n");
  return false;
 }

 // Tomar foto
 camera_fb_t *fb = esp_camera_fb_get();
 if (!fb) {
  ei_printf("Camera capture failed\n");
  return false;
 }
```

```
  // Convertir JPEG a RGB888
  bool converted = fmt2rgb888(fb->buf, fb->len, PIXFORMAT_JPEG, snapshot_buf);
  esp_camera_fb_return(fb);

  if (!converted) {
   ei_printf("Conversion failed\n");
   return false;
  }

  // Cambiar tamaño si es necesario
  if ((img_width != EI_CAMERA_RAW_FRAME_BUFFER_COLS) ||
     (img_height != EI_CAMERA_RAW_FRAME_BUFFER_ROWS)) {

   ei::image::processing::crop_and_interpolate_rgb888(
     out_buf,
     EI_CAMERA_RAW_FRAME_BUFFER_COLS,
EI_CAMERA_RAW_FRAME_BUFFER_ROWS,
     out_buf,
     img_width, img_height
   );
  }

  return true;
}

// --- Conversión de datos RGB a formato para Edge Impulse ---
```

```
static int ei_camera_get_data(size_t offset, size_t length, float *out_ptr) {
 size_t pixel_ix = offset * 3;
 size_t out_ptr_ix = 0;

 // Convertir cada píxel a un entero R+G+B
 while (length--) {
  out_ptr[out_ptr_ix++] = (snapshot_buf[pixel_ix + 2] << 16) +
                (snapshot_buf[pixel_ix + 1] << 8) +
                snapshot_buf[pixel_ix];
  pixel_ix += 3;
 }

 return 0;
}

// Validación del modelo
#if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR != EI_CLASSIFIER_SENSOR_CAMERA
#error "Invalid model for current sensor"
#endif
```

Explicación

Este programa implementa un sistema de detección de objetos utilizando un ESP32-CAM, un modelo de Edge Impulse, una pantalla OLED SSD1306 y tres LEDs de estado. El objetivo principal es capturar imágenes con la cámara, procesarlas mediante un

modelo de inteligencia artificial y mostrar los resultados tanto en pantalla como a través de los LEDs.

1. Configuración del Hardware

Cámara ESP32-CAM AI Thinker

Se define el modelo de cámara y los pines utilizados por el módulo:

Pines de datos: Y2 – Y9

Formato de la imagen: JPEG

Esta configuración permite capturar imágenes adecuadas para el modelo de Edge Impulse.

2. Pantalla OLED SSD1306

La pantalla se comunica por I2C, con los pines:

SDA → GPIO 15

SCL → GPIO 14

Se usa la librería Adafruit_SSD1306, con tamaño:

128×64 píxeles

La pantalla se utiliza para mostrar mensajes del sistema y resultados de detección.

3. LEDs de Estado

Se conectan tres LEDs para indicar el estado del sistema:

LED	Pin	Significado
Azul	12	Sin detección / estado inicial
Verde	13	Objeto "lufy" detectado
Rojo	2	Objeto "toga" detectado

Estos LEDs permiten visualizar la clasificación sin depender de la pantalla.

4. Inicialización del Sistema (setup)

En la función setup() se realizan los siguientes pasos:

Inicialización del puerto serial (115200 baudios).

Inicio del bus I2C y de la pantalla OLED.

Configuración de los LEDs como salidas.

Encendido inicial del LED azul.

Inicialización de la cámara mediante ei_camera_init().

Mensaje de espera de 2 segundos antes de iniciar la inferencia continua.

5. Ciclo Principal (loop)

Dentro de loop() se ejecuta continuamente el proceso de:

Captura de imagen

Se reserva memoria para el búfer de imagen.

Se captura una foto con ei_camera_capture().

Se convierte la imagen JPEG a formato RGB888.

Se ajusta al tamaño requerido por el modelo.

Preparación del modelo

Se genera un objeto signal que contiene cada píxel convertido para utilizarlo en el algoritmo de Edge Impulse.

Inferencia

Se ejecuta el modelo usando:

run_classifier(&signal, &result, debug_nn);

El modelo devuelve:

Coordenadas de los objetos detectados (si es detección por bounding boxes).

Nombres de las clases detectadas.

Porcentajes de confianza.

Tiempos de procesamiento (DSP, clasificación y anomalía).

6. Procesamiento de Resultados

Detección de Objetos

Si el modelo está configurado como detector, se revisan las cajas detectadas:

Se muestra en pantalla la etiqueta y el porcentaje.

Se identifica si se detectó:

"lufy"

"toga"

Control de LEDs según la detección

Detección	LED azul		LED verde	LED rojo
Nada detectado	ON	OFF	OFF	
lufy	OFF	ON	OFF	
toga	OFF	OFF	ON	
Otro objeto	ON	OFF	OFF	

Mensaje en OLED

Si hay detección: muestra la etiqueta y el porcentaje.

Si no hay detección: imprime “No detecto”.

7. Funciones Internas Importantes

ei_camera_init()

Inicializa la cámara con los parámetros establecidos.

Ajusta brillo, saturación y volteo de imagen si el sensor es OV3660.

ei_camera_capture()

Captura la imagen.

Convierte JPEG → RGB888.

Reduce o recorta la imagen al tamaño del modelo.

ei_camera_get_data()

Convierte valores RGB a formato entero para los algoritmos internos de Edge Impulse.

CONCLUSION

El proyecto de detección de objetos compacto y económico mediante la placa ESP-CAM (ESP32-CAM) no solo cumplió su objetivo principal, sino que se erigió como una demostración práctica y exitosa de la integración de conocimientos multidisciplinarios en el campo de la electrónica, la programación de sistemas embebidos y el aprendizaje automático *Edge* (en el dispositivo). La finalidad de desarrollar y probar la viabilidad de este sistema se alcanzó plenamente, validando la ESP32-CAM como una plataforma robusta y accesible para la visión artificial de bajo costo.

El trabajo sirvió como una prueba de fuego para los conocimientos adquiridos en la materia, obligando a la implementación práctica de conceptos que abarcan desde la programación en el entorno Arduino hasta la integración eléctrica y el entrenamiento de modelos de Machine Learning. La estrategia de utilizar la plataforma Edge Impulse se reveló como un acierto, facilitando el proceso de entrenamiento y optimización de un modelo de clasificación para el *hardware* limitado de la ESP-CAM. La consecución de porcentajes de reconocimiento entre el 96% y el 98% tras el entrenamiento es un indicador clave de la calidad del modelo generado y la efectividad de la recolección de datos (al menos 50 fotografías por objeto desde múltiples ángulos).

Uno de los logros más significativos fue la integración eléctrica y funcional del sistema. No se trató simplemente de un ejercicio de programación de cámara, sino de la creación de un sistema reactivo. La capacidad de la ESP-CAM para encender un LED **o** mostrar un mensaje en una pantalla OLED (Display**)** como respuesta directa a la detección de un objeto prueba el dominio de los principios de interfaz de *hardware* (interfaz I2C para el display) y el manejo de periféricos (GPIO). La implementación de librerías específicas como ADAFRUIT GFX fue fundamental para la correcta visualización de los mensajes de reconocimiento, demostrando la necesidad de gestionar dependencias y librerías externas.

El proceso de instalación y configuración del entorno de desarrollo (Arduino IDE) expuso un aprendizaje crucial sobre la realidad de trabajar con *hardware* y *software* de código abierto. La necesidad de solucionar problemas de comunicación (puertos COM), ajustar versiones de software y gestionar correctamente las librerías (*zip library*) subraya que la programación de sistemas embebidos requiere no solo conocimiento de código, sino también una habilidad avanzada en la depuración de la cadena de herramientas (*toolchain*). Los desafíos con la carga de librerías y la verificación inicial del funcionamiento de la cámara (mediante la visualización de la IP en el navegador)

no fueron meros pasos, sino hitos de validación que aseguraron que la base del proyecto fuera sólida.

El éxito de la generación e instalación de la librería de Edge Impulse y su posterior demostración de funcionamiento con la ESP-CAM, donde el *Serial Monitor* mostraba el estado de no-reconocimiento hasta que el objeto entrenado era presentado, valida la cadena completa del proceso: desde la adquisición de datos y el entrenamiento en la nube hasta el despliegue del modelo binario en el dispositivo final.

En conclusión, este proyecto ha trascendido la mera teoría para establecer un sistema funcional, costo-efectivo y eficiente de detección de objetos. Se ha comprobado la capacidad del equipo para integrar el Machine Learning con la ingeniería de hardware, resultando en una solución práctica que pone a prueba la habilidad de los desarrolladores para superar retos técnicos específicos del *hardware* de bajo consumo. El resultado es un sistema que no solo "ve" y clasifica, sino que reacciona de forma tangible a su entorno, sentando una base sólida para futuros proyectos de Internet de las Cosas (IoT**)** con capacidades de visión artificial integrada

Fuentes

“Login - Edge Impulse.” https://studio.edgeimpulse.com/studio/810405/acquisition/training?page=1

No title. (s/f). Arduino.cc. Recuperado el 30 de octubre de 2025, de https://www.arduino.cc/en/software/

boton_led - wokwi ESP32, STM32, Arduino simulator. (s/f). Wokwi.com. Recuperado el 30 de octubre de 2025, de https://wokwi.com/projects/443390746506023937

(S/f-c). Robu.in. Recuperado el 30 de octubre de 2025, de https://robu.in/orange-esp-32-cam-iot-kit/

(S/f-d). Circuitdigest.com. Recuperado el 30 de octubre de 2025, de https://circuitdigest.com/sites/default/files/circuitdiagram_mic/circuit-object-detector.jpg