Assistant for people with visual disabilities through the use of object detection, speech to text and ESP32-CAM

Asistente para personas con discapacidad visual mediante el uso de object detection, speech to text y ESP32-CAM

Domínguez-Nava, José Enrique*[a], Reyes-Nava, Adriana [b] and Gil-Antonio, Leopoldo[c]

[a] ROR TecNM: Tecnológico de Estudios Superiores de Jocotitlán • LMO-8422-2024• 0009-0005-5017-955X• 2073847
[b] ROR TecNM: Tecnológico de Estudios Superiores de Jocotitlán • LNP-9046-2024 • 0000-0002-4440-909X • 786639
[c] ROR TecNM: Tecnológico de Estudios Superiores de Jocotitlán • AEX-8979-2022• 0009-0002-1490-9592• 289388

CONAHCYT classification:

DOI: https://doi.org/10.35429/H.2024.12.28.40

Area: Engineering
Field: Technological sciences
Discipline: Computer technology
Subdiscipline: Real-time systems

Key Handbooks

This project contributes to the Science and Technology generation by combining real-time object detection using advanced models such as YOLOv8 and voice interaction, which improves accessibility for people with visual disabilities. Two key aspects are highlighted to apply this knowledge at a universal level: the use of artificial intelligence technologies for accessibility and the integration of voice interaction systems with portable devices. The main conclusions are the viability of the system to identify objects in real time and the possibility of evolving towards a more precise detection of specific objects requested by users. The author of the work comes mainly from a public institution, the most used keywords are: object detection, accessibility, visual disability, voice interaction, and wearable technologies.

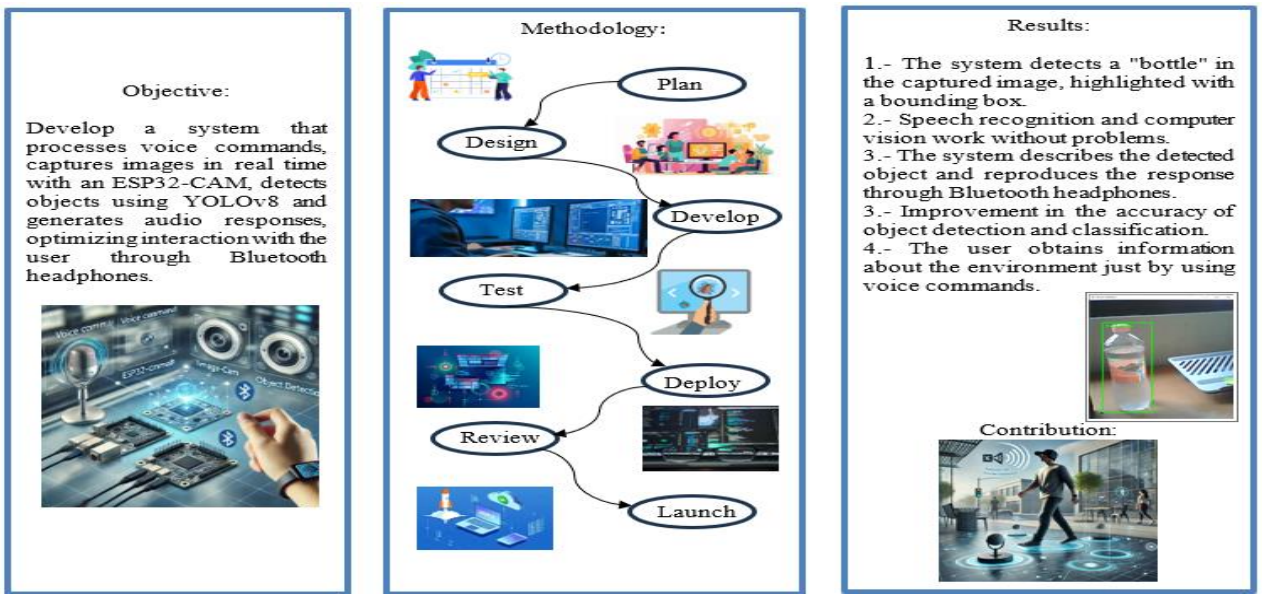
Citation: Domínguez-Nava, José Enrique, Reyes-Nava, Adriana and Gil-Antonio, Leopoldo. 2024. Assistant for people with visual disabilities through the use of object detection, speech to text and ESP32-CAM. 28-40. ECORFAN.

* 🖂 [2020150481131@tesjo.edu.mx]

Handbook shelf URL: https://www.ecorfan.org/handbooks.php

Abstract

This work presents the design of an assistant for people with visual disabilities, which combines real-time object detection and the use of voice recognition to facilitate verbal interactions. The project is divided into two phases: the first develops a server in Python that receives user requests using a keyword. Upon detection, the server sends an HTTP request to the ESP32-CAM, which captures images of the environment. These are processed with YOLOv8 to identify the objects present. The server then generates a response in audio format, played through Bluetooth headphones, describing the detected objects. The second phase seeks to improve object detection and help the user find a particular one, indicating its location or distance. The work focuses on the first phase, which covers the design of the communication between the user, the server and the ESP32-CAM.



Objective:

Develop a system that processes voice commands, captures images in real time with an ESP32-CAM, detects objects using YOLOv8 and generates audio responses, optimizing interaction with the user through Bluetooth headphones.

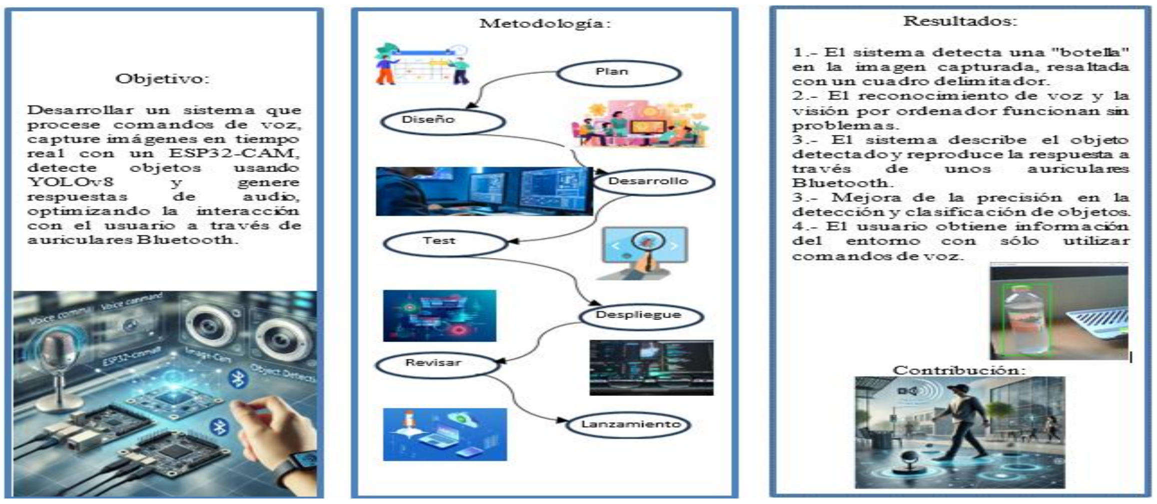
Results:

1.- The system detects a "bottle" in the captured image, highlighted with a bounding box.
2.- Speech recognition and computer vision work without problems.
3.- The system describes the detected object and reproduces the response through Bluetooth headphones.
3.- Improvement in the accuracy of object detection and classification.
4.- The user obtains information about the environment just by using voice commands.

Object Detection, Speech to Text, ESP32 CAM, People with Visual Disabilities

Resumen

Este trabajo presenta el diseño de un asistente para personas con discapacidad visual, que combina la detección de objetos en tiempo real y el uso de reconocimiento de voz para facilitar interacciones verbales. El proyecto se divide en dos fases: la primera desarrolla un servidor en Python que recibe solicitudes del usuario mediante una palabra clave. Al detectarla, el servidor envía una petición HTTP al ESP32-CAM, que captura imágenes del entorno. Estas se procesan con YOLOv8 para identificar los objetos presentes. Luego, el servidor genera una respuesta en formato de audio, que se reproduce a través de audífonos Bluetooth, describiendo los objetos detectados. La segunda fase busca mejorar la detección de objetos y ayudar al usuario a encontrar uno en particular, indicando su ubicación o distancia. El trabajo se enfoca en la primera fase, que cubre el diseño de la comunicación entre el usuario, el servidor y el ESP32-CAM.



Objetivo:

Desarrollar un sistema que procese comandos de voz, capture imágenes en tiempo real con un ESP32-CAM, detecte objetos usando YOLOv8 y genere respuestas de audio, optimizando la interacción con el usuario a través de auriculares Bluetooth.

Resultados:

1.- El sistema detecta una "botella" en la imagen capturada, resaltada con un cuadro delimitador.
2.- El reconocimiento de voz y la visión por ordenador funcionan sin problemas.
3.- El sistema describe el objeto detectado y reproduce la respuesta a través de unos auriculares Bluetooth.
3.- Mejora de la precisión en la detección y clasificación de objetos.
4.- El usuario obtiene información del entorno con sólo utilizar comandos de voz.

Detección de objetos, Speech to Text, ESP32 CAM, Personas con Discapacidades Visuales

1. Introduction

Accessibility for people with visual impairments remains a critical challenge in modern society, where technology has the potential to offer innovative solutions to improve their quality of life. According to the World Health Organization (WHO), approximately 285 million people in the world have visual impairments, of which 39 million are completely blind. The lack of autonomy in the navigation of public and private spaces is one of the most significant problems faced by these people, currently, tools such as canes or guide dogs are the main means of assistance, but they have limitations in terms of scope and functionality, it is in this context that the implementation of technologies based on artificial intelligence (AI) and voice recognition has shown enormous potential to improve the mobility and autonomy of people with visual impairments.

Recent advances in object detection and speech-to-text techniques have allowed for the development of virtual assistants that offer a more comprehensive solution, for example, projects such as Microsoft's Seeing AI use computer vision to detect objects and provide verbal descriptions. However, while these systems offer considerable improvement, they have limitations in terms of accuracy in dynamic environments and the ability to interact seamlessly with the user, another example is OrCam MyEye, which offers a combination of object detection and text recognition, but its high cost and reliance on specialized hardware restrict its accessibility.

This project proposes an innovative and accessible solution, leveraging low-cost hardware such as the ESP32 CAM and object detection models such as YOLOv8, along with a centralized server that processes images in real time and translates visual information into verbal responses through bluetooth headphones. Not only does this approach cost significantly less than other commercial products, but it also incorporates two-way interaction through the use of speech to text, allowing the user to not only receive information, but also to be able to make queries and give commands to the system.

The added value of this solution lies in its ability to detect objects in real time and offer fluid verbal interaction, differentiating itself from other systems by providing a more interactive and portable experience, previous studies have shown that the combination of computer vision and voice recognition can significantly improve the mobility of people with visual impairments. For example, a study published by the American Foundation for the Blind showed that 76% of participants in AI-assisted navigation tests experienced an improvement in their daily independence (American Foundation for the Blind, 2024).

The main objective of this project is to validate the feasibility of a visual assistance system that, in addition to offering object detection, can provide an interactive experience in which the user can request additional information about the environment. The central hypothesis is that by integrating object detection with voice recognition, it is possible to create a portable, efficient and economical system that allows visually impaired people to move more safely and autonomously in their everyday environment.

This project is divided into two phases: the first focuses on developing communication between the user and the server, which receives voice commands, detects objects through the ESP32 CAM camera and provides auditory responses. The second phase seeks to improve the detection of objects and allows the user to know the direction and distance of these, increasing the accuracy and usefulness of the system.

2. Theoretical background

2.1. Visual impairment

Sight is the most dominant of our senses, as it plays a fundamental role in all facets and stages of our lives, we take it for granted that we have sight, but without it it is difficult for us to learn, walk, read, participate in school and work. According to the WHO, globally at least 2,200 million people have near or distance vision problems, in at least 1,000 million (or almost half) of these cases, the disability could have been avoided or has not yet been addressed. Among these 1 billion people, the top diseases that cause distance vision loss or blindness are cataracts (94 million), refractive errors (88.4 million), age-related macular degeneration (8 million), glaucoma (7.7 million) and diabetic retinopathy (826 million). Because visual impairment severely affects the quality of life of adult populations, visually impaired adults may experience lower employment rates and higher rates of depression and anxiety (World Health Organization, 2023).

2.2. Advances in intelligence for people with visual impairments

AI has enabled the development of visual assistance systems based on computer vision, such as object detection, which provides real-time information about the environment, these systems allow users to know their immediate surroundings, recognizing objects and obstacles through cameras and deep learning models.

2.2.1. OrCam MyEye

This innovative product for the recognition of text, objects and faces, which consists of an accessory for any type of glasses that by taking images with a small camera is capable of analyzing them through artificial intelligence and converting what it identifies into voice format.

The device works autonomously without the need to connect to any phone or other type of equipment thanks to a battery, it is specially optimized for the detection of printed writing, but they have another series of interesting functionalities, it is configured to be used in the following languages: English, Hebrew, German, Spanish, French, Italian, Danish, Chinese, among others. The final retail price in Spain is 4500 euros plus VAT (Orientatech, 2019).

2.2.2. Seeing AI

Microsoft has launched the Seeing AI application for all Android devices, the app is free and is based on artificial intelligence (AI) technologies and cognitive services, which describe the world of blind or low-vision people directly from the mobile device, so it helps them in everyday tasks such as describing their environment, read email or listen to the characteristics of objects, photos or people, improving their autonomy. The application also allows you to change the audio channel to listen to specific information (Microsoft, 2023) such as:

- Short text: Reads the text as soon as it appears in front of the camera.
- Documents: Provides an audio guide for capturing a printed page and reads the content aloud, along with its original format.
- Products: Scans barcodes, using audio beeps as a guide and makes it easier to locate barcodes.
- Environment: Describes the environment and tapping "more info" generates a complete description.
- Currencies: Recognize banknotes and currencies, as well as their value.
- Colors: identifies and describes the perceived color.
- Handwriting: reads handwritten text.
- Light: generates an audible environment corresponding to the brightness of the environment.

2.3. Object Detection

Object detection is a computer vision task that aims to locate objects in digital images, as such, it is an instance of artificial intelligence that consists of training computers to see as humans do, specifically recognizing and classifying objects according to semantic categories. Object localization is a technique for determining the specific location of objects in an image by demarcating the object through a bounding box, object classification is another technique that determines which category a detected object belongs to, the object detection task combines object localization and classification subtasks to simultaneously estimate the location and type of object instances in one or more images (IBM, 2024)

2.3.1. YOLOv8

YOLOv8 is the latest iteration of the YOLO series of real-time object detectors, offering cutting-edge performance in terms of accuracy and speed, building on the advancements of previous versions of YOLO, YOLOv8 introduces new features and optimizations that make it the ideal choice for various object detection tasks in a wide range of applications (Ultralytics, 2024).

2.4 Speech to Text

Speech to Text uses model adaptation to improve the accuracy of frequently used words, expand the vocabulary available for transcribing, and improve the transcription of noisy audio, model adaptation allows users to customize Speech to Text to recognize specific words or phrases more frequently than other options that, otherwise, they would have been suggested, for example, you can adjust Speech to Text to transcribe "when" instead of "how much" more frequently. Speech to Text has three main methods for performing speech recognition: synchronous, asynchronous, and streaming. Each method returns text results based on whether it needs to be transcribed after processing, periodically or in real time, basically when you enter audio data, you receive a text response (Google Cloud, 2024).

2.5 ESP32 CAM

The ESP32 CAM module is a low-cost, full-featured ESP32-based microcontroller, an integrated small-sized OV2640 camera module, and a microSD card connector. This module integrates Bluetooth, WiFi, and BLE Beacon with two 32-bit high-performance LX6 CPUs, the frequency adjustment range of this module ranges from 80MHz to 240MHz, adopts stage channeling architecture, Hall sensor, on-chip sensor, temperature sensor, etc. This type of module is appropriate for industrial wireless control, smart home devices, wireless monitoring, and IoT applications that require a camera with superior functions such as image recognition and tracking (EL-PRO-CUS, 2024).

2.6 Agile Methodology

Agile is much more than a methodology for the development of projects that require speed and flexibility, it is a philosophy that involves a different way of working and organizing, in such a way that each project is broken into small parts that have to be completed and delivered in a few weeks. The objective is to develop quality products and services that respond to the needs of customers whose priorities are changing at an ever-increasing rate (BBVA, 2024).

The stages of the agile methodology (cognodata, 2024) are:

- Plan: the requirements and objectives of the sprint or cycle are identified, this stage defines what will be developed and the short-term goals.
- Design: a design or solution proposal is prepared for the functionalities to be developed, based on the requirements of the sprint.
- Develop: the development team implements the designed functionalities, codes and works on the prioritized tasks.
- Test: The team tests new functionalities to detect errors, ensuring that the software works correctly and meets the requirements.
- Deploy: Software or updates are released for customer or end-user use.
- Review: an evaluation of what has been completed is carried out, obtaining feedback from users or the team.
- Lauch: The functional version of the product is deployed for use, often with a version delivered to users

3. Development

This section will detail the development of the agile methodology, focused on the design of communication between the user and the server. Subsequently, the connection between the server and the ESP32 CAM will be implemented to carry out image capture and object detection in the environment. Finally, an auditory response based on the detected objects will be integrated.

3.1 Plan

In this first stage of the methodology, the plan contemplates the creation of a Python server capable of receiving requests from a local user. To simulate this environment, a client will be implemented that will activate the computer's microphone, allowing commands to be sent to the server. When the user speaks a keyword, the server will detect it and send an HTTP request to the ESP32 CAM in response.

This device will capture an image of the user's environment, which will be returned to the server for processing. The server will analyze the image, identify the objects present and, finally, generate an auditory response for the user, reproduced through headphones connected by Bluetooth.

3.2 Design

To know the communication flow, a diagram was made (see figure 1), the diagram describes an integrated system that facilitates the interaction between a client, a server and an ESP32 CAM device, designed to improve the autonomy of the user by detecting objects in their environment and providing auditory responses. Initially, the client, which might represent a local application or device, sends a request to the server to trigger object discovery. This request activates the client device's microphone, allowing the user to send voice commands.

If the request is successfully recognized by the server, the server proceeds to make an HTTP request to the ESP32 CAM, which is configured to take a picture of the user's environment. This image is then sent back to the server, where it is analyzed to identify objects present, this detection process is crucial, as it allows visual information to be transformed into an audible comment that is essential for users with visual limitations or for contexts where direct viewing is not possible.

Once the image has been processed and the objects detected, the server generates an auditory response, this response is transmitted through headphones connected to the system by Bluetooth, providing the user with information about their surroundings in an accessible way. This flow not only improves the user's interaction with their environment, but also enhances autonomy and safety, minimizing the need for direct assistance in navigating physical spaces.

Box 1

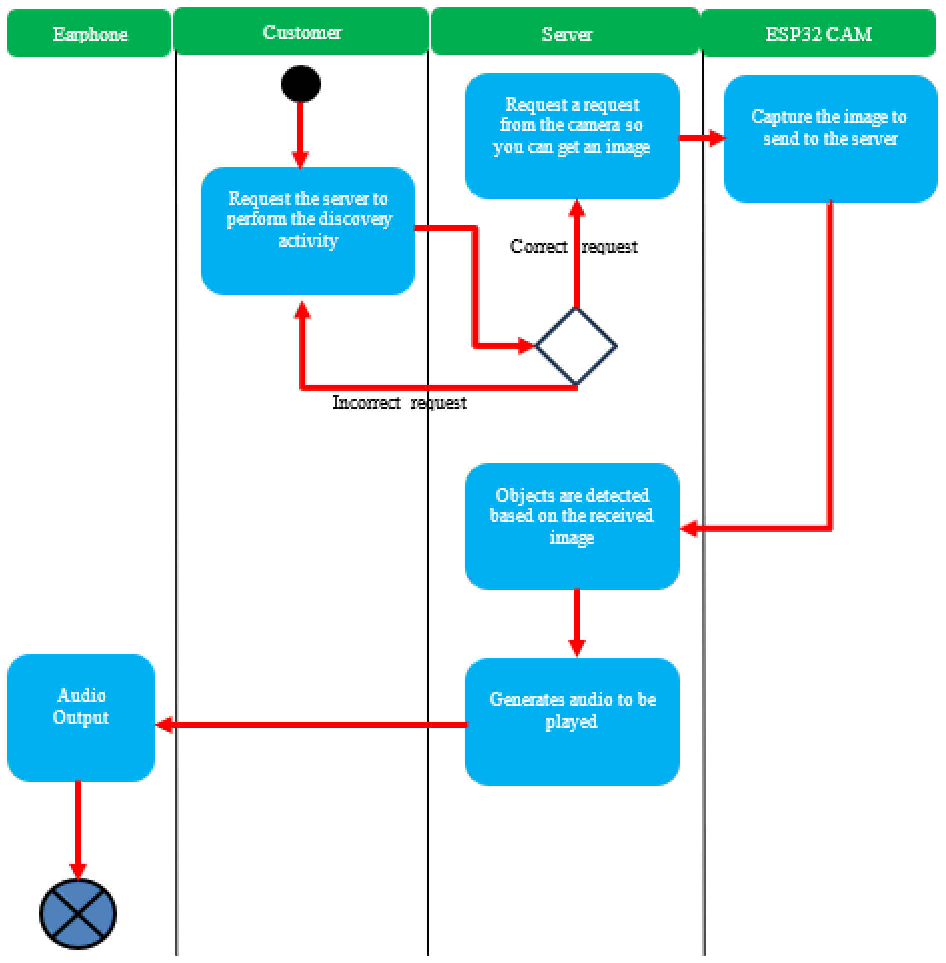

Earphone
Customer
Server
ESP32 CAM
Request a request from the camera so you can get an image
Capture the image to send to the server
Request the server to perform the discovery activity
Correct request
Incorrect request
Objects are detected based on the received image
Audio Output
Generates audio to be played

Figure 1

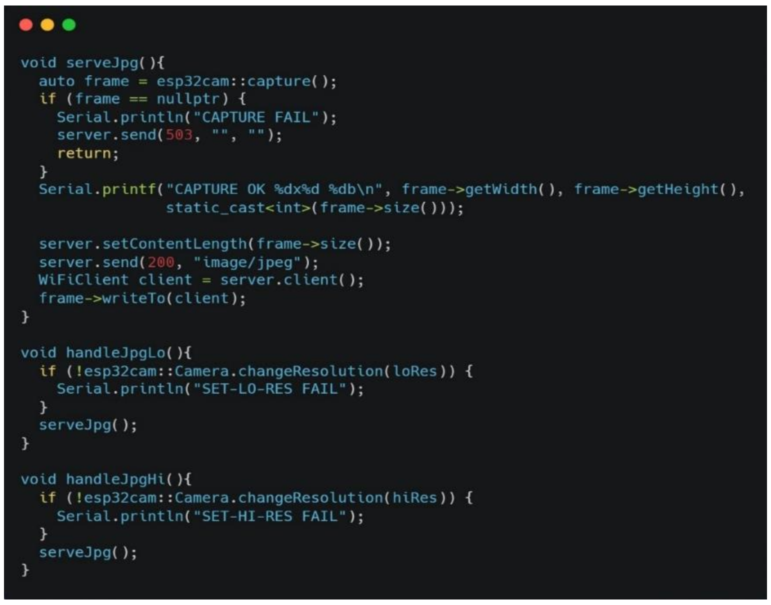
Flowchart of communication between client-server and server-ESP32-ACM

Source: Own work

3.3 Develop

As a first part an Arduino code was developed which will be loaded into the ESP32 CAM, this code configures an ESP32-CAM to capture and serve images through a web server, allowing users to access images in two different resolutions. First, it includes the necessary libraries: WebServer.h to manage the HTTP server, WiFi.h to connect to a Wi-Fi network, and esp32cam.h to handle the camera's functions. Two constants are defined that store the SSID and password of the Wi-Fi network to which the ESP32 will connect. A web server is then initialized on port 80, and two camera resolutions are configured: loRes (320x240) and hiRes (800x600). The serveJpg function (see Figure 2) captures an image using the ESP32's camera and sends it in JPEG format to the client that made the HTTP request. If the capture fails, a 503 (Service Unavailable) error is returned. In addition, two functions are defined, handleJpgLo and handleJpgHi, which change the camera resolution to low or high respectively, and then call serveJpg to capture and serve the image.

Box 2



```
void serveJpg(){
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
                static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleJpgLo(){
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handleJpgHi(){
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}
```

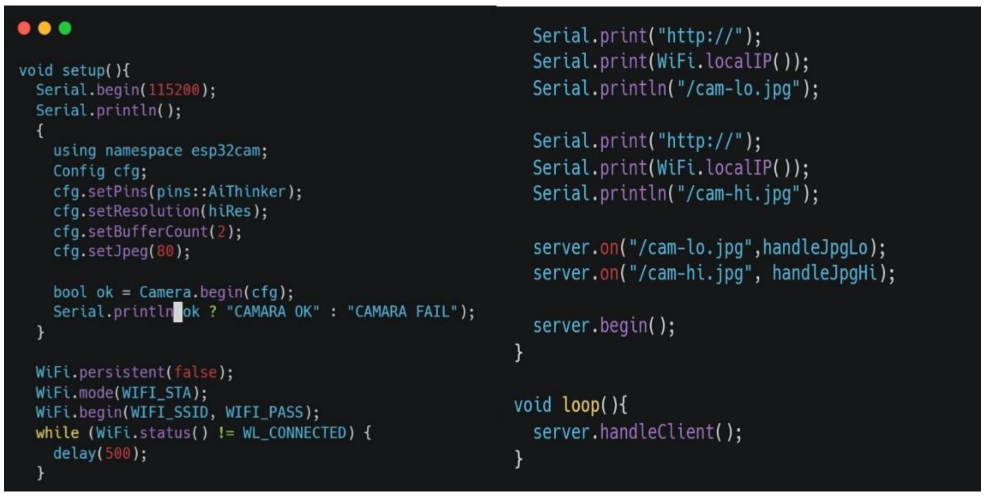
Figure 2

ServeJpg function

Source: Own work

In the setup function (see Figure 3) the ESP32 is initialized and connected to the Wi-Fi network using the credentials provided. Once connected, HTTP requests to the /cam-lo.jpg and /cam-hi.jpg paths are defined to execute functions to serve images in low and high resolution, respectively. Finally, in the loop function, the server continuously handles incoming requests. Thus, the ESP32-CAM can capture and transmit images over the network, accessible from a browser using the generated URLs.

Box 2



```
void setup(){
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMARA OK" : "CAMARA FAIL");
  }

  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
```

```
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/cam-lo.jpg");

  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/cam-hi.jpg");

  server.on("/cam-lo.jpg",handleJpgLo);
  server.on("/cam-hi.jpg", handleJpgHi);

  server.begin();
}

void loop(){
  server.handleClient();
}
```

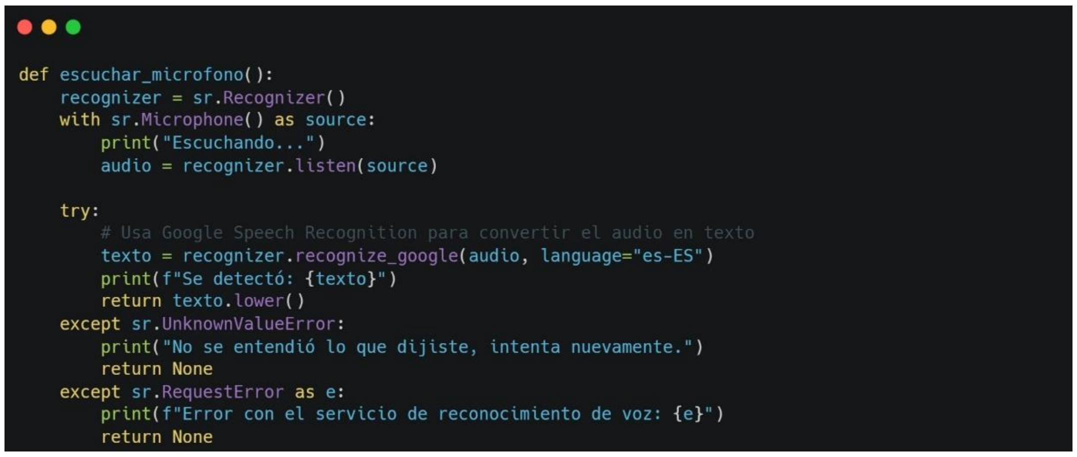
Figure 3

Setup and loop function

As a second phase, the development of the client was carried out using Python, so this code implements a system that listens for voice commands, sends requests to a server, and plays responses in audio format using the pyttsx3 library. First, several essential libraries are imported: speech_recognition for recognizing voice commands, requests for sending HTTP requests to the server, pygame for handling audio, and pyttsx3 for speech synthesis. The URL_SERVIDOR constant stores the address of the server to which the messages will be sent.

The escuchar_microfono function (see Figure 4) uses speech_recognition to activate the microphone and listen to the user's command. If speech recognition is successful, it converts audio to text using Google's API and returns it in lowercase. In case of an error, it shows messages indicating if it was not understood or if there was a problem with the service.

Box 4



```python
def escuchar_microfono():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Escuchando...")
        audio = recognizer.listen(source)

    try:
        # Usa Google Speech Recognition para convertir el audio en texto
        texto = recognizer.recognize_google(audio, language="es-ES")
        print(f"Se detectó: {texto}")
        return texto.lower()
    except sr.UnknownValueError:
        print("No se entendió lo que dijiste, intenta nuevamente.")
        return None
    except sr.RequestError as e:
        print(f"Error con el servicio de reconocimiento de voz: {e}")
        return None
```

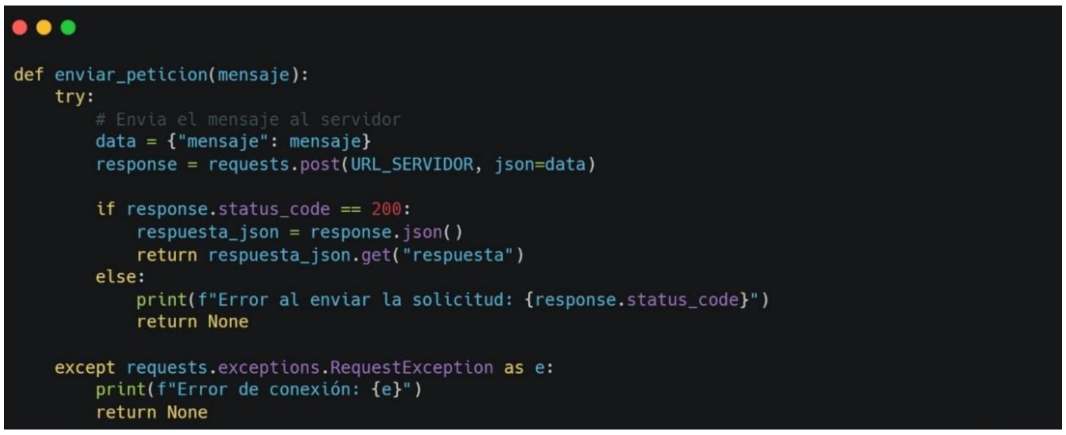
Figure 4

Function escuchar_microfono

Source: Own work

The enviar_peticion function (see Figure 5) takes the detected message and sends it to the server using an HTTP POST request. If the server's response is successful (code 200), it extracts and returns the "response" field from the received JSON. If the connection or request fails, it displays appropriate error messages.

Box 5



```python
def enviar_peticion(mensaje):
    try:
        # Envia el mensaje al servidor
        data = {"mensaje": mensaje}
        response = requests.post(URL_SERVIDOR, json=data)

        if response.status_code == 200:
            respuesta_json = response.json()
            return respuesta_json.get("respuesta")
        else:
            print(f"Error al enviar la solicitud: {response.status_code}")
            return None

    except requests.exceptions.RequestException as e:
        print(f"Error de conexión: {e}")
        return None
```

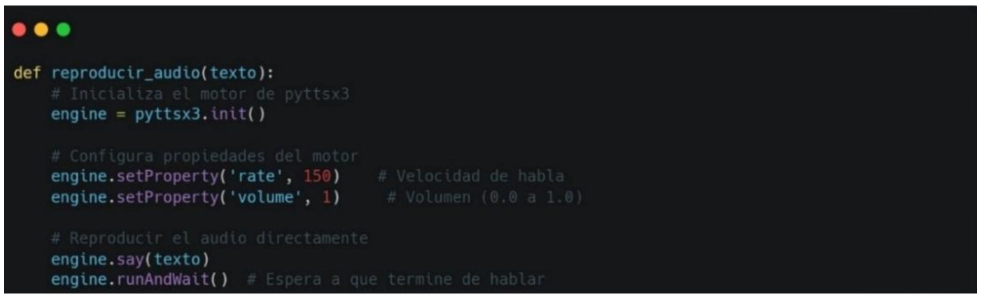
Figure 5

Function enviar_peticion

Source: Own work

The reproducir_audio function (see Figure 6) uses the pyttsx3 library to convert the received text into audio and play it back. You configure the speech rate and engine volume before generating the corresponding audio using the engine.say() function.

Box 6



```python
def reproducir_audio(texto):
    # Inicializa el motor de pyttsx3
    engine = pyttsx3.init()

    # Configura propiedades del motor
    engine.setProperty('rate', 150)    # Velocidad de habla
    engine.setProperty('volume', 1)     # Volumen (0.0 a 1.0)

    # Reproducir el audio directamente
    engine.say(texto)
    engine.runAndWait()  # Espera a que termine de hablar
```

Figure 6

Function reproducir_audio

Source: Own work

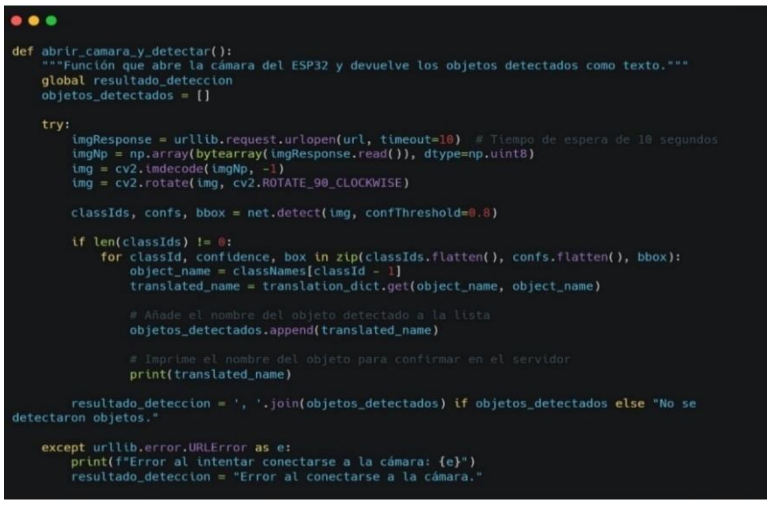
The main flow of the program is in the main function which constantly listens for voice commands. When the keyword "activate" is detected, it sends this message to the server, and if it receives a response, it plays it back in the form of audio. Subsequently, it waits for the user to say "what's there", sends this second request to the server, and also reproduces the response obtained.

Finally, for the server side, the code configures an object detection system using a pre-trained model of the **MobileNet SSD** neural network, trained on the COCO dataset. The model conFiguretion includes the upload of two key files: the conFiguretion file (ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt), which defines the structure of the neural network, and the weights file (frozen_inference_graph.pb), which contains the model's trained parameters. These files allow the model to analyze images and detect objects. To prepare images, you adjust properties such as input size, color scales, and color channel order using OpenCV's setInputSize, setInputScale, setInputMean, and setInputSwapRB functions. This ensures that the images are matched to the model's requirements to optimize its performance and accuracy.

It also includes a translation dictionary that is used to convert the names of detected objects (which are in English) into Spanish. This is done to provide more understandable results for Spanish-speaking users. For example, when the model detects a "person" or "car", the system returns "person" or "car", respectively. This step is important to maintain a more natural and friendly experience for the end user, by generating responses in their preferred language.

The abrir_camara_y_detectar function (see Figure 7) captures an image from an IP camera using an HTTP request with the urllib library. The image is downloaded in JPEG format, decoded, and rotated 90 degrees so that it is in the correct orientation. The OpenCV object detection model then analyzes the image, and if objects with a confidence greater than 0.8 are detected, their names are translated into Spanish, and they are saved in a list, this list is converted into text that is sent in response. If the connection to the camera fails, the function returns an error message.

Box 2



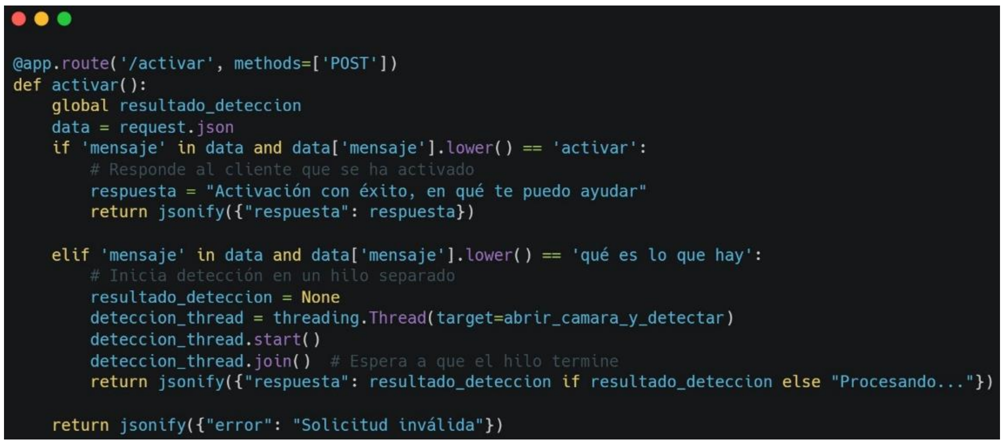
```python
def abrir_camara_y_detectar():
    """Función que abre la cámara del ESP32 y devuelve los objetos detectados como texto."""
    global resultado_deteccion
    objetos_detectados = []

    try:
        imgResponse = urllib.request.urlopen(url, timeout=10)  # Tiempo de espera de 10 segundos
        imgNp = np.array(bytearray(imgResponse.read()), dtype=np.uint8)
        img = cv2.imdecode(imgNp, -1)
        img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)

        classIds, confs, bbox = net.detect(img, confThreshold=0.8)

        if len(classIds) != 0:
            for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):
                object_name = classNames[classId - 1]
                translated_name = translation_dict.get(object_name, object_name)

                # Añade el nombre del objeto detectado a la lista
                objetos_detectados.append(translated_name)

                # Imprime el nombre del objeto para confirmar en el servidor
                print(translated_name)

        resultado_deteccion = ', '.join(objetos_detectados) if objetos_detectados else "No se
detectaron objetos."

    except urllib.error.URLError as e:
        print(f"Error al intentar conectarse a la cámara: {e}")
        resultado_deteccion = "Error al conectarse a la cámara."
```

Figure 7

Function abrir_camara_y_detectar

The Flask server defines the /activate endpoint (see Figure 8) to handle POST requests and process two messages. If you receive 'activate', it responds with a confirmation message indicating that the system is ready. If it receives 'what's there', it initiates object detection from the IP camera in a separate thread, allowing the server to continue responding to other requests. After the detection is complete, the server sends the list of detected objects. If you're not done yet, reply with "Processing..."

Box 8



```python
@app.route('/activar', methods=['POST'])
def activar():
    global resultado_deteccion
    data = request.json
    if 'mensaje' in data and data['mensaje'].lower() == 'activar':
        # Responde al cliente que se ha activado
        respuesta = "Activación con éxito, en qué te puedo ayudar"
        return jsonify({"respuesta": respuesta})

    elif 'mensaje' in data and data['mensaje'].lower() == 'qué es lo que hay':
        # Inicia detección en un hilo separado
        resultado_deteccion = None
        deteccion_thread = threading.Thread(target=abrir_camara_y_detectar)
        deteccion_thread.start()
        deteccion_thread.join()  # Espera a que el hilo termine
        return jsonify({"respuesta": resultado_deteccion if resultado_deteccion else "Procesando..."})

    return jsonify({"error": "Solicitud inválida"})
```

Figure 8
Activate endpoint

Source: Own work

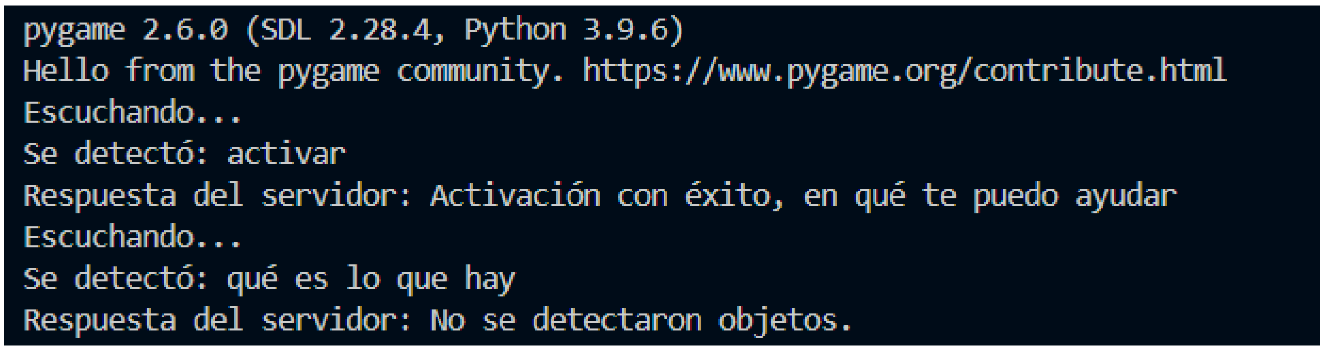
Finally, the Flask application runs in multithreaded mode on port 5000 allowing the server to process multiple requests simultaneously. This is crucial in applications where image processing can take time, as it allows multiple client requests to be handled without the server crashing or delaying its responses.

3.4 Test and review

The message indicates that the ESP32-CAM's camera has been successfully booted and provides two links to view images captured in low and high resolution. It also confirms that images with a resolution of 800x600 pixels have been captured and shows the size of each file in bytes. All of this suggests that the ESP32-CAM is performing well and capturing images smoothly.

The Flask server receives several HTTP POST requests from IP 192.168.167.157 to the /activate endpoint, successfully responding with code 200, since when the client detects the word "activate", it sends the request to the server (see figure 9) which responds with "Activation successfully, how can I help you". Subsequently, the client hears the phrase "what's there", sends it to the server, and it responds with "No objects detected", each response that the client obtains from the server is printed by console, but at the same time it is in the form of audio. This interaction flow shows how the client and server communicate efficiently, allowing for the processing of voice commands and the return of responses.

Box 9



```
pygame 2.6.0 (SDL 2.28.4, Python 3.9.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
Escuchando...
Se detectó: activar
Respuesta del servidor: Activación con éxito, en qué te puedo ayudar
Escuchando...
Se detectó: qué es lo que hay
Respuesta del servidor: No se detectaron objetos.
```

Figure 9
Server response

Source: Own work

3.5 Deploy and lauch

The initial deployment of the project has been carried out in a controlled environment, where the application, using a Flask server, allows the detection of objects through an IP camera and responds to voice commands. In this first phase, the system is able to identify common objects and return their names translated into Spanish. Although this working version is already underway, the project is not yet finished, as the current detection is limited to basic objects and does not provide additional details, such as their exact location or distance.

In the second phase, significant improvements will be implemented to increase the accuracy of the system and its usefulness. New functionalities will be added that will allow the user to know the **direction and distance** of the detected objects, making the system more detailed and accurate. In addition, detection algorithms will be optimized to improve the hit rate and reduce errors, allowing the system to better adapt to different conditions and scenarios. The final launch of the project is planned after the integration and testing of these improvements, guaranteeing a more complete and reliable experience for the user.

4. Results

When the client utters the phrase "what's up?", a communication flow is activated, initially the request is sent to the server, which is responsible for coordinating the interaction with an ESP32 CAM. This device, upon receiving the command, captures an image of the environment in real time. The image is then sent back to the server, where a computer vision model processes the visual information, analyzing the objects present in the scene. In this specific example, the system has detected an object, identified as a "bottle" (see Figure 10) highlighted with a bounding box and a corresponding label.

The success of this process depends on a series of seamless interactions between speech processing, image capture, and object detection algorithms. By combining voice recognition and machine vision, the system offers an integrated solution that allows users to obtain visual information of the environment simply through voice commands, this approach not only optimizes the capture and classification of objects.

Box 10

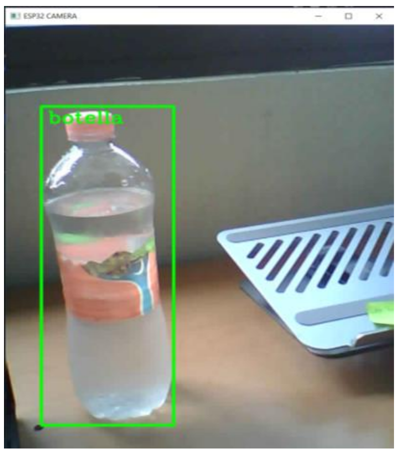


Figure 10
Image capture

Source: Own work

5. Conclusions

In conclusion, the development of the object detection system has come a long way, reaching a significant value with the implementation of the first phase, where it has been possible to establish a functional communication that combines a Flask server with an IP camera. This initial approach has not only enabled the identification of common objects through voice commands, but has also laid the groundwork for more intuitive and accessible interaction for the user. However, this first achievement is only the starting point, as the true potential of the system will be deployed in the second phase, which is geared towards optimizing the accuracy and expanding the usefulness of the system.

In this phase of improvement, the incorporation of functionalities that will allow the user not only to detect objects, but also to know their direction and distance, which will transform the nature of the system by offering contextual information that is a little more detailed. These innovations will not only enrich the user experience, but also expand the reach of the system in complex environments, such as security applications, assisted navigation. By integrating more functional detection algorithms and advanced image processing techniques, the system will be able to better adapt to different environmental conditions and dynamic scenarios, ensuring a higher hit rate and reducing false positives.

6. Declarations

6.1. Conflict of interest

The authors declare that they have no conflicts of interest. They have no known competing financial interests or personal relationships that might have appeared to influence the article reported in this paper.

6.2 Authors' contribution

Domínguez-Nava, José Enrique: In the development of this project, I was in charge of all phases, from conceptualization to final implementation. Designed and developed the accessible system, including the integration of real-time object detection and voice interaction. Additionally, I implemented the use of a Bluetooth microphone and headphones to ensure a comfortable and portable experience for users. The entire process was carried out by me, with special attention to improving the autonomy of people with visual disabilities.

Reyes-Nava Adriana: Support in the idea of the project, Distribution of parts of the project, Review of the article.

6.3. Availability of data and materials

The data and materials used and generated during this project are available upon reasonable request to the author. Interested parties may contact to obtain access to data, source code or materials related to the development of the accessible system. It is guaranteed that all information provided complies with ethical and data protection regulations.

6.4. Funding

This project did not have any financing.

6.5. Acknowledgements

I want to express my most sincere gratitude to my project advisors for their valuable guidance, support and feedback throughout the development of this work. Their experience and dedication were fundamental to the completion of this project.

6.6. Abbreviations

HTTP	HyperText Transfer Protocol
WHO	World Health Organization
AI	Artificial Intelligence
VAT	Value-Added Tax
Bit	Binary Digit
MHz	Megahercios
SSID	Service Set Identifier
JPEG	Joint Photographic Experts Group
URL	Uniform Resource Locator
API	Application Programming Interface
JSON	JavaScript Object Notation
IP	Internet Protocol

7. References

Backgroud

The American Foundation For The Blind. (2024). Blindness and Low Vision. Consultado 13 de septiembre, 2024

World Health Organization: WHO. (2023). Blindness and vision impairment. Consultado 13 de septiembre, 2024,

Orientatech. (2023). Orcam MyEye 2.0 - Orientatech | Tu asesor de tecnologías sociales. Consultado 13 de septiembre, 2024,

Basic

Microsoft. (2023). Microsoft lanza Seeing AI en Android, la app gratuita que facilita el reconocimiento y descripción del entorno a personas ciegas con nuevos idiomas y funciones actualizadas. Consultado 13 de septiembre, 2024

Murel, J. (2024). What is object detection?. Consultado 13 de septiembre, 2024, desde

Ultralytics. (2024). YOLOV8. Ultralytics YOLO Documentos. Consultado 13 de septiembre, 2024,

Support

Google Cloud. (2024). IA de Speech-to-Text: reconocimiento y transcripción de voz. Consultado 13 de septiembre, 2024

Agarwal, T. (2024). ESP32 Cam : PinOut, Specifications, Types, Interfacing & its Uses. ElProCus - Electronic Projects For Engineering Students. Consultado 13 de septiembre, 2024

Differences

Canfranc, M. R. (2023). «Scrum», «agile» . . . así son las nuevas formas de trabajo para la transformación de BBVA. BBVA NOTICIAS. Consultado 13 de septiembre, 2024, desde

Discussions

Cognodata. (2023). 12 principios de la metodología agile en el desarrollo de proyectos - Cognodata. Consultado 13 de septiembre, 2024