**Vision-Based Assistive Device for the Visually Impaired using Xiao ESP32-S3 Sense and YOLOv3**

**Project Overview:**

This project focuses on the development of a cost-effective and portable assistive system for blind and visually impaired individuals. The device uses real-time object detection and audio feedback to inform the user about nearby objects, enhancing navigation and situational awareness.

**Key Components:**

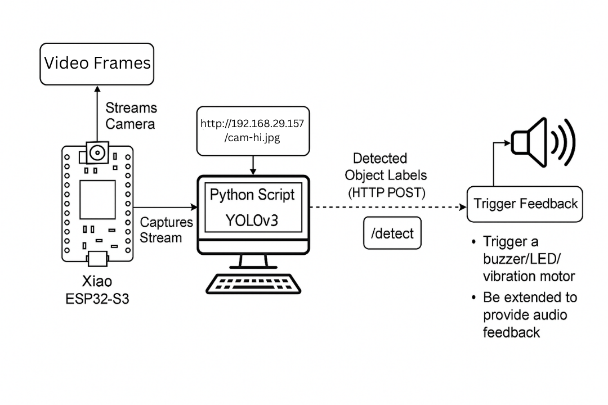
**1. Hardware**

* **Microcontroller:** Xiao ESP32-S3 Sense
* **Camera Module:** Built-in with the ESP32-S3 Sense
* **Other Peripherals:** GPIO-based outputs for buzzer/vibrator or LED
* **Power Supply:** USB-C or LiPo battery (for future portability)

**2. Software Stack**

* **Microcontroller Environment:** Arduino IDE with ESP32 libraries
* **Object Detection:** YOLOv3 (Darknet model) using Python + OpenCV
* **Communication Protocol:** HTTP (via POST requests from PC to ESP32)
* **Camera Streaming:** Captures live stream from the Xiao ESP32-S3 via /camera route

**System Architecture:**

1. The Xiao ESP32-S3 captures video frames through its onboard camera.
2. The ESP32 streams the video locally (e.g., http://192.168.x.x/camera).
3. A **Python script** running on a computer captures this stream and performs object detection using YOLOv3.
4. Detected object labels are extracted from the model output and sent to the ESP32-S3 as HTTP POST requests (/detect route).
5. Upon receiving the label, the ESP32 can:
   * Trigger a **buzzer/LED/vibration** motor for feedback.
   * Be extended to provide **audio feedback** using external TTS modules or via PC speakers.

**Implementation Details:**

**ESP32 Arduino Sketch**

* Configured camera settings using esp\_camera.h.
* Launched a simple web server with two endpoints:
  + /camera: Streams camera feed.
  + /detect: Accepts detected object labels via POST and triggers GPIO pin for feedback.
* Used Serial.println() for real-time debugging of received object labels.

**Python + YOLOv3 Script**

* Loaded the YOLOv3 model and configuration files (.cfg and .weights).
* Read class labels from coco.names.
* Captured image frames from the camera stream using OpenCV.
* Processed each frame to:
  + Detect objects with bounding boxes.
  + Filter results based on confidence and NMS thresholds.
  + Send detected labels to the ESP32 via POST request.
* Provided live feedback on console for detection results.

**Feedback Mechanism:**

Currently, the system uses a **GPIO trigger** to turn on an output pin upon receiving any label. This is intended to be connected to:

* **LED**: For basic visual cue
* **Buzzer**: For short beeps upon detection
* **Vibration Motor**: For tactile feedback (recommended for visually impaired users)

**Planned Upgrade:**

* **Audio Feedback using ESP Sense Board and Amplifier:**
  + Using prerecorded audio stored in a SD Card to be played using an amplifier board and speaker.
* **Audio Feedback via Text-to-Speech (TTS):**
  + Using Python's pyttsx3 or gTTS to read out the detected object labels.
  + Can be played through a speaker for direct user feedback.

**Sample Output:**

Detected Objects:

→ PERSON (98%)

→ CHAIR (89%)

→ BOTTLE (90%)

**Testing Status:**

* Camera stream working via ESP32
* Object detection accurate and real-time
* Object label transmission via HTTP POST
* GPIO output control verified
* Audio feedback integration (next step)