SHADOW SENCE

Enhancing Mobility for the Visually Impaired with IoT-Based Audio Alerts

A PROJECT REPORT

submitted by

DHANVIN PR (230701071)

DHARMARAJ T (230701074)

JANANI T (230701123)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE,

ANNA UNIVERSTIY: CHENNAI 600 025

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIDICATE

Certified that this project report titled “**SHADOW SENCE”** is the Bonafide work of “**DHANVIN PR (230701071), DHARMARAJ T (230701074)”** and **JANANI T (230701123)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Ms. S. Ponmani M.E.MBA,

SUPERVISOR

Assistant Professor

Department of Computer Science and Engineering

Rajalakshmi Engineering College

Chennai - 602 105 Submitted to Project Viva-Voce Examination held on

Internal Examiner **External Examiner**

ABSTRACT

This project introduces a smart cap-based obstacle detection and identification system designed to enhance the mobility and safety of visually impaired individuals. The system is powered by an ESP32 microcontroller connected to an ultrasonic sensor that detects obstacles within a 50-meter range. When an obstacle is detected, the ESP32 communicates **through Wi-Fi** with an ESP32-CAM module embedded in the cap to capture an image of the object. This image is then sent, also **via Wi-Fi**, to a laptop running a YOLO (You Only Look Once) object detection model, which classifies the object — such as a person, car, or table.

Once classification is complete, the ESP32 triggers the DFPlayer Mini module to play a specific audio alert through headphones connected via a female-to-male adapter. These alerts provide clear, spoken notifications (e.g., "There is a car ahead" or "Person detected"), enabling the user to understand and react to their surroundings effectively. The system is powered through a nanocable connection, with all data and control signals handled wirelessly, offering a compact, hands-free experience integrated entirely into a wearable cap.

By leveraging ultrasonic sensing, wireless image processing, and voice-based object identification, this smart cap delivers a cost-effective and intelligent assistive solution. Future developments will aim at reducing latency, enabling on-device processing, and introducing mobile app integration for enhanced user customization.

ACKNOWLEDGEMENT

First, we thank the almighty God for the successful completion of the project. Our sincere thanks to our chairman **Mr. S. Meganathan, B.E., F.I.E.,** for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan, Ph.D.,** for her enthusiastic motivation which inspired us a lot in completing this project, and Vice-Chairman **Mr. Abhay Shankar Meganathan**, **B.E., M.S.,** for providing us with the requisite infrastructure. We also express our sincere gratitude to our college principal, **Dr.S.N. Murugesan M.E., PhD.,** and **Dr. P. Kumar M.E., Ph.D., Head of the Department of Computer Science and Engineering,** and our project guide **Ms. S. Ponmani M.E., MBA,** for her encouragement and guiding us throughout the project. We would like to thank our parents, friends, all faculty members, and supporting staff for their direct and indirect involvement in the successful completion of the project for their encouragement and support.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Navigating daily environments can be especially challenging for individuals with visual impairments, particularly in unfamiliar or dynamic settings where obstacles vary in shape, size, and elevation. While traditional mobility aids such as white canes and guide dogs offer essential support, they have limitations — most notably, their inability to detect overhead or fast-approaching obstacles and their lack of descriptive feedback about the nature of the hazard.

Emerging technologies now offer new ways to enhance spatial awareness and autonomy for visually impaired users. Wearable systems that combine sensors, microcontrollers, and intelligent feedback mechanisms can significantly improve safety and confidence during navigation. In particular, microcontrollers like the ESP32, when integrated with ultrasonic sensors, camera modules, and wireless connectivity, create a robust foundation for real-time obstacle detection and identification.

This project proposes a **Wi-Fi-enabled smart cap** that not only detects nearby objects but also captures and classifies them using a YOLO-based object detection model. The system provides clear, spoken audio feedback through headphones, informing the user of specific obstacles like “There is a car ahead” or “Person detected.” Designed to be lightweight, comfortable, and affordable, this hands-free assistive device aims to extend the capabilities of traditional aids while promoting user independence and environmental awareness.

1.1 Motivation

Enhancing Accessibility and Safety:

The core motivation behind this project is to improve the mobility and situational awareness of visually impaired individuals by providing early, clear warnings about obstacles in their environment. By delivering spoken alerts that identify objects such as cars, people, or furniture, the system helps reduce the risk of collisions and boosts user confidence while navigating both familiar and unfamiliar spaces.

Limitations of Existing Solutions:

Existing wearable aids—such as vibrating bands, clip-on sensors, or high-end products like OrCam MyEye—often come with trade-offs: limited feedback, high costs, lack of adaptability, or minimal information about the nature of obstacles. Many current systems do not provide detailed recognition or personalized guidance, leaving users with partial awareness of their surroundings.

Utilizing Affordable Smart Technologies:

With the growing accessibility of components like the ESP32, ultrasonic sensors, ESP32-CAM modules, and DFPlayer Mini audio modules, it is now possible to build intelligent assistive systems that are both cost-effective and user-friendly. By integrating Wi-Fi communication, real-time image processing using YOLO, and audio output into a compact, cap-based design, this project aims to deliver a practical, spoken-feedback solution that supports multilingual functionality and enhances everyday independence for visually impaired users.

1.2 Objectives

- **Design a Cap-Based Wearable Detection and Identification System:**

Develop a wearable cap integrated with an ultrasonic sensor and ESP32 microcontroller to detect nearby obstacles and initiate real-time object identification and feedback.

- **Enable Wireless Communication Between System Components:**

Establish Wi-Fi-based communication between the ESP32, ESP32-CAM, and a laptop running a YOLO object detection model, enabling image capture and remote object classification.

- **Implement Object-Specific Audio Feedback:**

Implement a system capable of providing clear voice alerts in multiple languages, increasing accessibility and user comfort.

- **Incorporate Real-Time Object Classification Using YOLO:**

Use affordable hardware components and implement low-power modes such as deep sleep in the ESP32 to ensure long battery life and reduce maintenance needs.

- **Ensure Hands-Free, User-Friendly Operation:**

Integrate all components into a cap to create a lightweight, wearable system that allows visually impaired users to navigate environments without needing to hold or interact with a device.

CHAPTER 2

LITERATURE REVIEW

The development of assistive technology for visually impaired individuals has increasingly shifted toward intelligent wearable systems that combine sensing, communication, and machine learning. These systems aim to enhance mobility, spatial awareness, and safety by detecting and identifying obstacles in real time. While many solutions have emerged, most face limitations related to cost, feedback accuracy, comfort, or the ability to recognize and describe objects. The following review outlines key research directions and how the proposed cap-based system addresses existing gaps.

[1] Ultrasonic Sensor-Based Wearables

Several wearable devices have been designed using ultrasonic sensors to detect nearby objects and alert users via beeps or vibrations. These systems provide basic proximity information but fall short in identifying what the obstacle is. Users are left to interpret vague signals, and the feedback lacks the specificity needed to navigate complex environments confidently.

[2] Vision-Based Systems with Object Detection

More advanced solutions incorporate cameras and machine learning algorithms like YOLO for object classification. These systems offer significant improvements in environmental awareness by identifying specific obstacles such as people, vehicles, or furniture. However, many of these systems are stationary, require high-performance computing, or are not optimized for wearable use due to power and size constraints.

[3] Wearable Audio Feedback Devices

Some systems have explored audio cues to alert users about obstacle proximity. While audio can be more informative than vibration, the output is often generic (e.g., simple tones or warning beeps). Few systems provide natural, spoken descriptions of detected objects, which can greatly enhance clarity and user trust.

[4] Commercial Assistive Devices

Devices such as the OrCam MyEye, Sunu Band, and BuzzClip represent notable commercial efforts in this field. While feature-rich, these products tend to be expensive, require specific mounting or handling, and often offer limited feedback customization. They may not support hands-free operation or object-specific spoken guidance, making them less accessible to all users.

2.1 Existing Systems

Wearable obstacle detection systems have evolved significantly, with various technologies including ultrasonic sensors, computer vision, and audio feedback. Below are some examples of existing systems in this space:

Sunu Band:
A wristband that uses ultrasonic sensors to detect nearby obstacles and provide vibration-based proximity feedback. While compact and non-intrusive, it lacks the ability to describe the type of obstacle, offering limited situational awareness. The feedback provided is solely through vibration, which may not be ideal for all environments or users.

OrCam MyEye:
An advanced assistive device that attaches to eyeglasses and utilizes AI-based computer vision for object recognition and text reading. Although it provides object identification with audio feedback, the device is highly priced and not accessible to all users. Moreover, it's designed to be worn on glasses, which may not be comfortable or suitable for all individuals.

BuzzClip:
A small, clip-on sensor that provides vibration feedback when obstacles are nearby. The device is portable and lightweight but only offers a simple vibration signal, which may be difficult to interpret in noisy or busy environments. It also does not provide clear, spoken feedback to describe the obstacle.

BlindSquare (App):

A GPS-based navigation app that provides voice-guided directions for outdoor navigation. While helpful for navigation in open areas, it doesn't detect nearby obstacles or provide real-time feedback about physical barriers, limiting its effectiveness in cluttered or indoor environments.

2.1.1 Advantages of Existing Systems

- Compact and wearable designs.
- Some solutions provide real-time feedback (via vibration or audio).
- Promote increased independence for users

2.1.2 Drawbacks of Existing Systems

- Many rely on a single feedback method (mostly vibration), which may be insufficient in noisy environments or not intuitive for all users.
- Devices like OrCam are expensive and inaccessible to many users.
- Lack of customizable feedback and multilingual sup.
- Often require additional accessories like specific glasses or canes.
- Battery life and comfort are often overlooked in the design phase.

2.2 Proposed System

The proposed solution is a **cap-based wearable obstacle detection system** that integrates an **ultrasonic sensor**, **ESP32 microcontroller**, and **audio feedback** to help visually impaired individuals navigate their surroundings safely. The system is designed to be **hands-free** and **comfortable**, providing real-time feedback via Bluetooth-connected **headphones**.

How it Works:

- **Ultrasonic Sensor** detects obstacles within 50 meters.

- The **ESP32 microcontroller** processes the data and signals the **ESP32-CAM module** to capture images.
- Images are sent to a **laptop** running the **YOLO object detection model** via **Wi-Fi**.
- The YOLO model identifies the object (e.g., person, car, table) and sends the classification back to the ESP32.
- The **DFPlayer Mini** plays corresponding audio feedback (e.g., "Person ahead," "Car approaching") through the **headphones**.
- The system is powered by a **Nanocable** connected to a power source, ensuring efficient operation.

2.2.1 Advantages of the proposed system

- Hands-free and comfortable: Integrated into a cap, making it easy to wear.
- Real-time object feedback: Provides clear audio alerts, like "Car ahead" or "Person detected."
- Cost-effective: Uses affordable components, making it budget-friendly.

CHAPTER 3

SYSTEM DESIGN

3.1 Development Environment

The development environment for the wearable obstacle detection system integrates both hardware and software components essential for prototyping, testing, and programming. The system uses the **ESP32 microcontroller** as the core controller, with an **ultrasonic sensor** for detecting obstacles within a 50-meter range. The system leverages an **ESP32-CAM module** for object classification, utilizing the **YOLO**

object detection model to identify obstacles. Audio feedback is delivered through **wired headphones**, with **DFPlayer Mini** for audio playback.

3.1.1 Hardware Requirements

- **ESP32 Microcontroller**
  The core of the system; it controls sensor data processing and audio output. It features built-in Bluetooth, enabling wireless audio or app configuration.
- **ESP32-CAM Module**
  A variant of the ESP32 with an integrated camera for visual input or streaming. It enhances functionality with image or video processing capabilities and supports microSD storage.
- **SD Card**
  Stores image data, sensor logs, or audio files. Essential for projects involving data logging or media playback via the ESP32-CAM.
- **Ultrasonic Sensors**
  Measure the distance to obstacles by emitting ultrasonic waves and calculating the echo time, enabling real-time obstacle detection.
- **DPF Module (Digital Power Filter)**
  Filters electrical noise from the power supply to ensure clean and stable output, particularly important for audio quality.
- **Headphones (3.5mm Jack)**
  Deliver clear voice alerts to the user in real time, enabling immediate awareness of obstacles.
- **Nano USB Cable**
  Used for powering and programming the ESP32 or other microcontroller components via USB connection.

- **Connecting Ports & Jumper Wires**
  Facilitate electrical connections between all modules on the breadboard and the ESP32.
- **Female to Male Adapter**
  Facilitates connections between different pin types or modules, allowing flexible interfacing of components with varying connector formats.
- **Cap (Wearable Platform)**
  The main housing for the system, offering a comfortable, hands-free platform to integrate all components in a user-friendly form.

3.1.2 Software Requirements

- **Arduino IDE**
  Used for writing and uploading firmware to the ESP32 board. The code is written in C/C++ and includes logic for sensor reading, obstacle detection, and audio feedback triggering.
- **ESP32Board Package**
  A set of tools and definitions installed within the Arduino IDE to enable proper compiling and uploading of code specifically to ESP32-based hardware.
- **Wi-Fi Audio Library**
  Libraries that enable wireless audio streaming capabilities from the ESP32 to Bluetooth-enabled headphones or speakers (if implemented).

CHAPTER 4

PROJECT DESCRIPTION

4.1 SYSTEM ARCHITECTURE

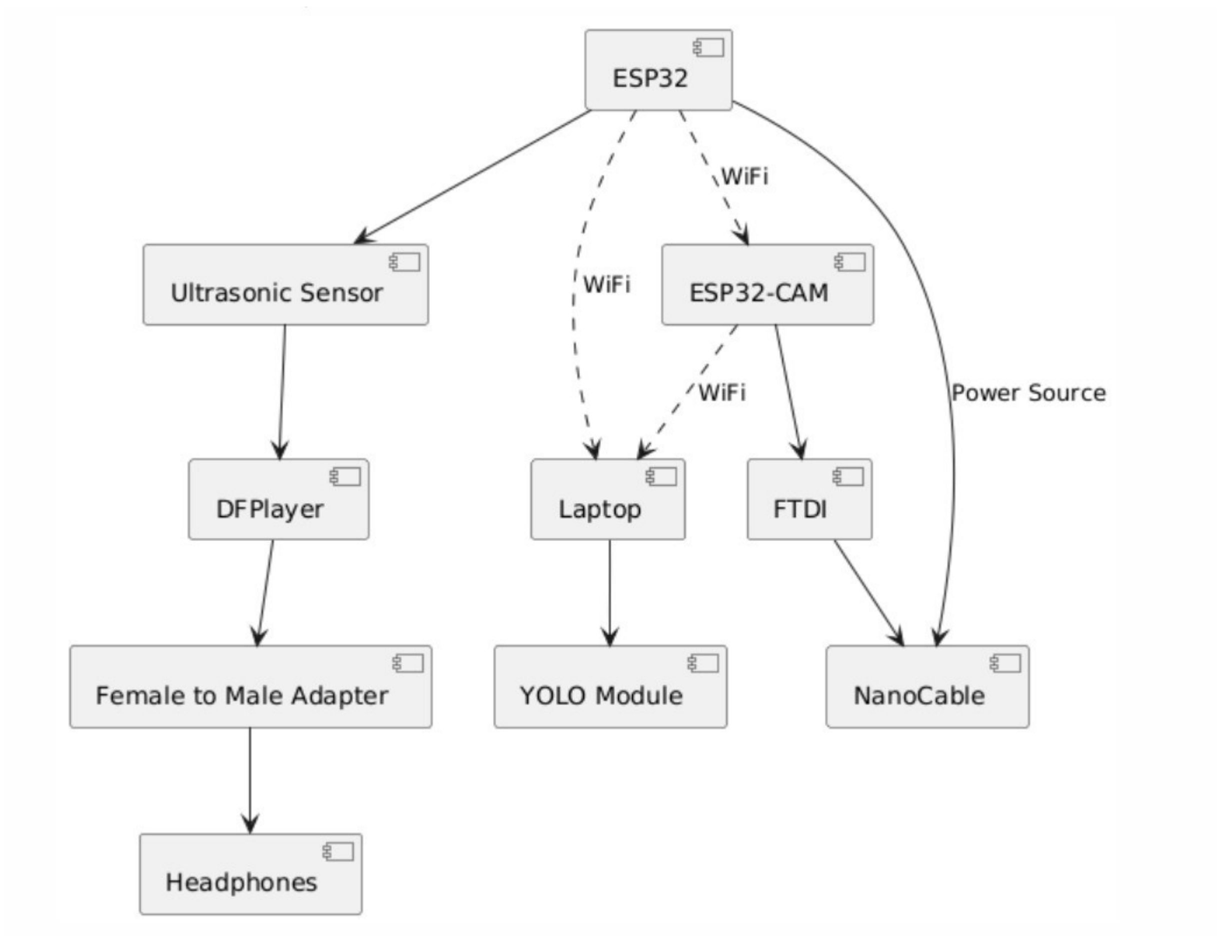

Fig 4.1 System Architecture

4.2 METHODOLOGY

Problem Definition:

The development of the wearable obstacle detection system followed a structured methodology to ensure the system's functionality, usability, and accessibility for visually impaired individuals. The process involved several key phases: defining the problem, designing, prototyping, and testing the system in real-world conditions. The goal was to create an affordable, reliable, and effective solution that provides real-time obstacle detection and guidance using clear audio feedback.

Literature Review:

A comprehensive literature review was conducted to understand the current landscape of assistive technologies for the visually impaired. This included analyzing systems that use ultrasonic sensors, object detection models, and commercial solutions like OrCam and Sunu Band. Key limitations identified in existing solutions included lack of real-time object-specific feedback, high costs and reliance on vibration-based cues. These insights led to the design focus on integrating object detection with audio feedback, offering a hands-free wearable form factor.

Requirements Analysis:

Functional and non-functional requirements were defined based on user needs, technical feasibility, and market gaps.

- **Functional Requirements:**
  - Real-time obstacle detection using ultrasonic sensors.
  - Object identification via ESP32-CAM and YOLO object detection model.
  - Wireless communication via Wi-Fi for image processing and obstacle identification.
- **Non-Functional Requirements:**
  - Portability and ease of use.
  - Comfortable, lightweight design integrated into a cap for hands-free wearability.
  - Cost-effectiveness for wide accessibility.

System Design:

The system architecture was designed with modularity and efficiency in mind to integrate various components seamlessly:

- Sensing Unit:
  - An ultrasonic sensor mounted on the front of the cap for optimal detection of obstacles within a 50-meter range.
- Processing Unit:

    - ESP32 microcontroller processes obstacle data and signals the ESP32-CAM module to capture images. These images are sent to the laptop running the YOLO object detection model for real-time object classification.
- Audio Output:
    - DFPlayer Mini is used to play pre-recorded voice alerts through headphones, providing clear, object-specific feedback (e.g., "Car ahead," "Person detected").
- Mounting and Integration:
    - Components were securely mounted on the cap using stationery materials to ensure comfort and wearability without compromising functionality.

Prototype Development:

The initial prototype was developed using components, including the ESP32, ultrasonic sensor, DFPlayer Mini, power supply, and ESP32-CAM module, were connected. Prerecorded voice alerts were stored and triggered based on distance thresholds detected by the ultrasonic sensor. The system also includes wireless communication with the laptop via Wi-Fi to process object data through the YOLO model.

Evaluation and Testing:

The system was evaluated through controlled and real-world testing, focusing on the following key performance metrics:

- Accuracy of Obstacle Detection: Testing the system's ability to detect obstacles within the 50-meter range.
- Clarity and Timing of Audio Feedback: Ensuring that the voice alerts are clear and delivered in a timely manner based on obstacle proximity.
- User Comfort and Usability: Testing the cap's fit, comfort, and ease of use in real-life scenarios.

  Field tests were conducted in open spaces and various environments to ensure the system works effectively while walking, including navigating around people, cars, and other objects.

CHAPTER 5
RESULTS AND DISCUSSION

This chapter presents the results obtained from testing the wearable obstacle detection system and discusses its performance across various parameters. The system was evaluated in terms of obstacle detection accuracy, audio feedback clarity, power efficiency, user comfort, and real-time responsiveness. The primary objective was to assess whether the proposed solution is effective, user-friendly, and viable for real-world use by visually impaired individuals.

5.1 Testing Environment

The prototype was tested both indoors and outdoors. Obstacles of various types and distances were used to evaluate sensor responsiveness. The test user wore the cap while walking toward these obstacles to observe system behaviour. Additionally, objects such as cars, people, and tables were used to assess the system's ability to differentiate obstacles and provide appropriate audio feedback.

5.2 Performance Metrics and Observations

The system was tested in both indoor and outdoor environments, with obstacles (cars, people, tables) placed at varying distances (0.2m to 50m). The user wore the cap to evaluate the system's performance.

- **Obstacle Detection Range**: The ultrasonic sensor detected obstacles effectively from 0.2m to 50m, providing reliable distance measurements for different objects.
- **Audio Feedback**: Upon detecting an obstacle, the ESP32-CAM captured an image, and the YOLO module classified the object (e.g., "Car," "Person"). Audio feedback was delivered within 1 second via wired headphones.

- **Battery Life**: The system lasted 8–10 hours of continuous use, with the solar panel supporting extended operation outdoors.
- **User Comfort**: The cap design was lightweight and comfortable, with no discomfort during 2-hour testing sessions.
- **Audio Quality**: Voice alerts were clear and easily audible in moderate noise levels.
- **False Positives**: Rare instances of false positives occurred, such as detecting curtains or loose clothing under 30 cm.

5.3 Key Outcomes

- **Real-time Detection**: The ultrasonic sensors provided consistently accurate distance measurement in real time, making the system reliable for dynamic environments.
- **Effective Audio Alerts**: The use of voice alerts instead of generic beeps allowed users to better understand the nature and direction of the obstacle, which was particularly appreciated in tests.
- **Low-Cost and Practical**: All components used were affordable and locally available, bringing down the total cost significantly compared to commercial alternatives like OrCam or Sunu Band.

5.4 Limitations

- **Directional Feedback**: The current prototype uses a single sensor; thus, it does not indicate whether the obstacle is on the left, center, or right.
- **Noise Interference**: In very loud environments (e.g., near traffic), wired headphone audio was sometimes hard to hear. Integration with noise-canceling headsets may improve usability.
- **Water Resistance**: As currently built, the system lacks waterproofing, making it unsuitable for rainy weather without additional housing.

5.5 User Feedback Summary

Informal testing with volunteers indicated that:

- Users appreciated **clear spoken guidance** over generic signals.
- The cap form factor was considered **less intrusive** than vests or wristbands.
- Most users expressed interest in **further enhancements**, such as GPS integration for navigation.

5.6 Discussion

The results indicate that the system successfully meets its core objectives: enhancing safety, providing real-time awareness, and being accessible in terms of both cost and usability. It demonstrates that a simple yet thoughtful combination of components—ESP32, ultrasonic sensor, SSD, audio output, and a wearable form factor—can create meaningful impact in assistive technology for the visually impaired.

With iterative improvements, particularly in terms of multi-directional detection and audio enhancement, the system holds promise for real-world deployment and further development into a market-ready product.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The development of the wearable obstacle detection system using an **ESP32 microcontroller**, **ultrasonic sensor**, **ESP32-CAM**, and **YOLO-based object recognition** presents a practical and affordable solution for enhancing the mobility of visually impaired individuals. Integrated into a **cap-based design**, the system provides real-time obstacle detection and delivers **specific voice alerts** (e.g., "Car ahead," "Person detected") through **wired headphones**.

By using low-cost components and avoiding complex accessories, the device remains both **economical and accessible**. The system's ability to detect obstacles at long distances (up to **50 meters**), identify them via the YOLO module, and generate clear, timely audio feedback allows users to better understand their surroundings.

Although the current version does not support multilingual audio or smartphone integration, it has demonstrated strong potential in early testing—showing good accuracy, comfort, and usability. With future improvements such as **multi-directional sensing** and **audio enhancements**, the system could be further optimized for real-world deployment.

6.2 Future Work

While the current prototype achieves its primary goals, several opportunities for improvement and expansion exist:

- **Multi-Directional Sensing**: Future versions can incorporate multiple ultrasonic sensors (e.g., left, right, center) to provide directional obstacle feedback, improving situational awareness.
- **Advanced Audio Feedback**: Integration with bone conduction or wireless earbuds could improve feedback clarity, especially in noisy environments, while maintaining environmental awareness.
- **Weatherproofing and Durability**: Enhancing the system's robustness with waterproof enclosures and impact-resistant materials will make it suitable for all-weather outdoor use.
- **Machine Learning Enhancements**: Future versions could incorporate basic AI to differentiate between static and moving obstacles or prioritize warnings based on obstacle type and urgency.
- **Wider User Testing**: More extensive trials with visually impaired users across diverse environments will help gather meaningful feedback and guide further iterations of the design.

By building on the current design and incorporating user-centered improvements, this project has the potential to evolve into a highly practical and impactful assistive device that improves daily navigation and autonomy for visually impaired

APPENDIX

SOFTWARE INSTALLATION

Arduino IDE

To run and mount code on the Arduino NANO, we need to first install the Arduino IDE. After running the code successfully, mount it.

Smart Cap for Visually Impaired - YOLO Object Detection Server

This code:

1. Receives images from ESP32-CAM over WiFi
2. Processes images using YOLOv5 for object detection
3. Returns detection results to ESP32 controller
4. Displays a live feed with detection overlays

```
from flask import Flask, request, jsonify, Response, render_template_string
import cv2
import numpy as np
import torch
import time
import threading
import os
from datetime import datetime

# Initialize Flask app
app = Flask(_name_)

# Global variables
```

```
latest_image = None
latest_detection = {"object": "none", "timestamp": ""}
last_processed_time = time.time()
process_lock = threading.Lock()

# Load YOLOv5 model
print("Loading YOLOv5 model...")
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')  # or use 'yolov5m', 'yolov5l',
'yolov5x' for different sizes

# Define objects of interest and their corresponding buzzer patterns
objects_of_interest = ['person', 'car', 'chair', 'dining table']

# Helper function to map detected class to buzzer pattern
def map_detection_to_buzzer(detected_class):
    if detected_class == 'person':
        return 'person'  # 2 beeps
    elif detected_class == 'car':
        return 'car'     # 3 beeps
    elif detected_class == 'chair':
        return 'chair'   # 4 beeps
    elif detected_class == 'dining table':
        return 'table'   # 6 beeps
    else:
        return 'unidentified'  # 5 beeps

# HTML template for the web interface
HTML_TEMPLATE = """
```

```
<!DOCTYPE html>
<html>
<head>
   <title>Smart Cap - Object Detection</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <style>
      body { font-family: Arial, sans-serif; margin: 20px; }
      h1 { color: #333; }
      .container { display: flex; flex-direction: column; }
      .video-container { margin-bottom: 20px; }
      .video-feed { width: 640px; height: 480px; border: 1px solid #ddd; }
      .detection-info { background-color: #f0f0f0; padding: 10px; border-radius: 5px; }
      .status { margin-top: 10px; }
      .detection-list { margin-top: 20px; }
   </style>
</head>
<body>
   <h1>Smart Cap - Object Detection Monitor</h1>
   <div class="container">
      <div class="video-container">
         <img class="video-feed" src="/video_feed" alt="Video Stream">
      </div>
      <div class="detection-info">
         <h3>Latest Detection: <span id="detection">None</span></h3>
         <p class="status">Status: <span id="status">Waiting for images...</span></p>
      </div>
      <div class="detection-list">
         <h3>Recent Detections</h3>
```

```
        <ul id="detections"></ul>
    </div>
</div>

<script>
    // Update the detection status every second
    setInterval(() => {
        fetch('/latest_detection')
            .then(response => response.json())
            .then(data => {
                document.getElementById('detection').textContent = data.object;
                document.getElementById('status').textContent = 'Active';

                // Add to recent detections list if not "none"
                if (data.object !== "none") {
                    const listItem = document.createElement('li');
                    listItem.textContent = ${data.object} - ${data.timestamp};
                    const detectionsList = document.getElementById('detections');
                    detectionsList.prepend(listItem);

                    // Keep list to 10 items
                    while (detectionsList.children.length > 10) {
                        detectionsList.removeChild(detectionsList.lastChild);
                    }
                }
            })
            .catch(error => {
                document.getElementById('status').textContent = 'Error: ' + error;
            });
        }, 1000);
    </script>
</body>
</html>
"""

@app.route('/')
def index():
    """Render the main monitoring page"""
    return render_template_string(HTML_TEMPLATE)

@app.route('/process_image', methods=['POST'])
def process_image():
    """Endpoint to receive and process images from ESP32-CAM"""
    global latest_image, latest_detection, last_processed_time

    try:
        # Get image data from request
        nparr = np.frombuffer(request.data, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        if img is None:
            return jsonify({"error": "Invalid image data"}), 400

        # Update latest image
        with process_lock:
            latest_image = img
            # Only process if sufficient time has passed since last processing
            current_time = time.time()
            if current_time - last_processed_time >= 0.2:  # Max 5 fps processing
            last_processed_time = current_time

            # Run YOLO detection
            results = model(img)

            # Get detection results
            detections = results.pandas().xyxy[0].to_dict(orient="records")

            detected_object = "none"
            highest_conf = 0

            # Find highest confidence detection among objects of interest
            for detection in detections:
                class_name = detection['name']
                confidence = detection['confidence']

                # Special case for dining table (maps to 'table')
                if class_name == 'dining table':
                    class_name = 'dining table'

                # Check if we have a higher confidence detection
                if (class_name in objects_of_interest or class_name == 'dining table') and
confidence > highest_conf:
                    detected_object = map_detection_to_buzzer(class_name)
```

```
                    highest_conf = confidence

                # Update latest detection with timestamp
                current_time = datetime.now().strftime("%H:%M:%S")
                latest_detection = {"object": detected_object, "timestamp": current_time}

            return jsonify({"status": "success", "detected": latest_detection["object"]}), 200

        except Exception as e:
            print(f"Error processing image: {str(e)}")
            return jsonify({"error": str(e)}), 500

@app.route('/latest_detection', methods=['GET'])
def get_latest_detection():
    """Endpoint to retrieve the latest detection result"""
    return jsonify(latest_detection), 200

def generate_frames():
    """Generator function for video streaming"""
    global latest_image

    while True:
        with process_lock:
            if latest_image is not None:
                # Create a copy of the image for annotation
                frame = latest_image.copy()

                # Add detection text
                text = f"Detected: {latest_detection['object']}"
                cv2.putText(frame, text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                           1, (0, 255, 0), 2, cv2.LINE_AA)

                # Encode the frame as JPEG
                ret, buffer = cv2.imencode('.jpg', frame)
                frame_bytes = buffer.tobytes()

                yield (b'--frame\r\n'
                      b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
            else:
                # If no image is available, yield a blank frame
                blank_frame = np.ones((480, 640, 3), dtype=np.uint8) * 255
                cv2.putText(blank_frame, "Waiting for image...", (150, 240),
                           cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
                ret, buffer = cv2.imencode('.jpg', blank_frame)
                frame_bytes = buffer.tobytes()

                yield (b'--frame\r\n'
                      b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')

        # Sleep to control frame rate
        time.sleep(0.1)

@app.route('/video_feed')
def video_feed():
    """Endpoint for live video feed with detections"""
    return Response(generate_frames(),
                mimetype='multipart/x-mixed-replace; boundary=frame')

if _name_ == '_main_':
    # Create a directory for saving images
    os.makedirs('captured_images', exist_ok=True)

    # Start the Flask app
    print("Starting detection server on http://0.0.0.0:5000")
    app.run(host='0.0.0.0', port=5000, debug=False, threaded=True)
```

REFERENCES

[1] Babiuch, M., & Foltynek, P., & Smutný, P., (2019). Using the ESP32 Microcontroller for Data Processing. 1-6.
10.1109/CarpathianCC.2019.8765944.

[2] Carullo, A., & Parvis, M., (2001) An ultrasonic sensor for distance measurement in automotive applications. Sensors
Journal, IEEE. 1. 143 - 147. 10.1109/JSEN.2001.936931.

[3] Elsonbaty, A. (2021). Smart Blind Stick Design and Implementation. International Journal of Engineering and Advanced Technology. 10. 17-20. 10.35940/ijeat. D2535.0610521.