IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Ai-Based Smart Shoes With Offline Voice Assistant And Wireless Process For Visually Impaired

Mrs.Manasa M, Amulya N, Isaac Yesuraj Y, M Mathew, Vasanth G

Assistant Professor, Student, Student, Student, Student
Department Of Artificial Intelligence and Machine Learning
Vijaya Vittala Institute of Technology, Bengaluru, India

Abstract:This project introduces a Smart Shoe system paired with a Pocket Device that enables offline voice-based interaction using the ESP32 Dev Module. It incorporates wake word recognition via TensorFlow Lite Micro, local command processing using ESP-SR, and voice feedback through ESP-TTS, with audio delivered over Bluetooth—completely independent of internet connectivity or cloud services. The Pocket Device serves as the processing hub, interpreting data from sensors embedded in the shoe and providing instant auditory responses through connected wireless earphones. Designed for energy efficiency and compactness, the system enhances mobility, safety, and accessibility—particularly for individuals with visual impairments.

Keywords: ESP32, Smart Shoe, Pocket Device, Wake Word Detection, Offline Voice Assistant, Bluetooth TTS, ESP-SR.

I. INTRODUCTION

In the modern era of fast-paced technological advancement, wearable gadgets are becoming smarter, more compact, and widely available. This project introduces a Smart Shoe equipped with a Voice Assistant, leveraging a pocket-sized device driven by the ESP32 Dev Module. It is specifically designed to enhance user mobility—especially for the visually impaired—by facilitating offline voice interactions. This is achieved through integrated wake word detection and local command recognition, eliminating the dependency on internet connectivity.

II. OBJECTIVE

The goal of this project is to design an innovative Smart Shoe integrated with a fully offline voice assistant. The system incorporates a variety of sensors and a compact, ESP32-powered pocket module to offer real-time, hands-free support to users. Key aims include:

1. Improving safety and independent mobility for individuals, particularly the visually impaired, by delivering real-time environmental data through sensor-based alerts.
2. Utilizing efficient, offline speech recognition technologies like TensorFlow Lite Micro and ESP-SR for wake word detection and command execution.
3. Enabling local text-to-speech conversion using ESP-TTS, with audio output transmitted wirelessly via Bluetooth-enabled headphones or speakers.
4. Achieving complete offline functionality by embedding all processing capabilities within the ESP32 chip, removing the need for internet or external hardware.
5. Providing a lightweight, detachable processing unit that interprets sensor data, handles voice commands, and issues spoken feedback.

6. Ensuring energy-efficient operation and easy portability through optimized resource management—making the system suitable for everyday use.

III. LITERATURE SURVEY

The fusion of wearable devices and smart voice-enabled systems has seen remarkable progress, particularly in enhancing mobility and independence for elderly individuals and those with visual impairments. Prior studies on smart footwear have largely centered around obstacle avoidance using sensors such as infrared or ultrasonic modules. For instance, research by S. Kumar et al. (2018) introduced a navigation system based on vibrational cues triggered by detected obstacles. On the other hand, mainstream voice assistants like Google Assistant and Alexa depend on constant internet connectivity, rendering them impractical for standalone or offline embedded platforms.

Recent developments from Espressif, such as the ESP-SR framework and TensorFlow Lite Micro, have made it feasible to implement offline speech recognition on resource-constrained devices like the ESP32. Their demonstrations highlight the effectiveness of wake word activation and command recognition without needing cloud-based processing. Similarly, the ESP-TTS engine supports localized text-to-speech functionality. This project leverages these capabilities within a compact wearable module, enabling efficient, internet-free interaction for smart shoe systems—paving the way for truly independent, real-world assistive solutions.

IV. PROPOSED SYSTEM

1. The user initiates interaction by saying the wake phrase (e.g., “Hi Smart Shoe”).
2. TensorFlow Lite Micro running on the ESP32 detects the wake word locally.
3. The system switches to listening mode, where ESP-SR identifies predefined voice commands such as “start navigation” or “check battery.”
4. The ESP32 accesses sensor data from both the smart shoe and the pocket module.
5. Based on the voice command and the input data, ESP-TTS converts the response into speech.
6. The synthesized audio is wirelessly transmitted via Bluetooth A2DP to a paired neckband or earbud.
7. All processes are handled offline, using internal flash or an SD card for storing audio models and voice data.

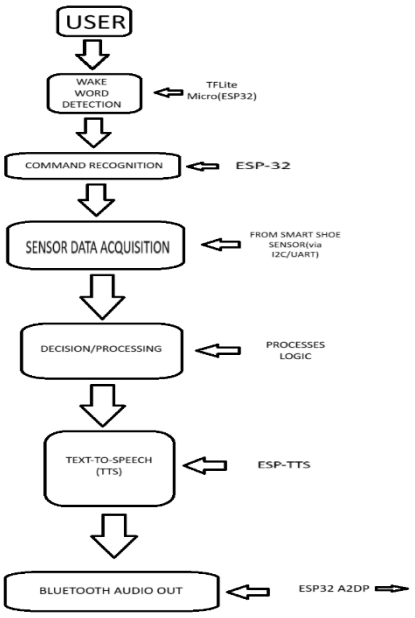

Fig. 1

Components & Block Description:

- Shoe-Mounted Sensors:
  - Ultrasonic Sensor: Detects obstacles in front.
  - IR/Tracking Sensor: Helps with surface recognition or line following.
  - MPU6050 IMU: Monitors posture, falls, or sudden motion.
  - These sensors connect to the ESP32 in the pocket device via I2C/digital lines through a detachable connector.
- Pocket Device (ESP32 Core Unit):
  - Wake Word Detection: TFLite Micro model embedded.
  - Voice Commands Recognition: Using ESP-SR.
  - Speech Output: ESP-TTS module for generating voice replies.
  - Bluetooth A2DP Sink: Sends responses to paired Bluetooth audio devices.
  - Power Supply: Powered by a 2000mAh Li-ion battery with TP4056 charging module.
  - Storage: Optional SD card for storing speech/audio models.
  - Status LEDs: Optional indicators for system status or responses.
- Bluetooth Audio System:
  - Communicates using A2DP Sink profile.
  - Sends real-time speech to wireless audio output (e.g., earbuds or neckband).
  - Outputs examples: “Obstacle ahead,” “Starting Navigation,” “Low battery.”

System Loop Summary:

1. ESP32 listens for the wake phrase (“Hi Smart Shoe”).
2. Once triggered, it activates voice command mode.
3. Reads relevant data from connected sensors.
4. Generates a response and converts it into speech.
5. Delivers it via Bluetooth earphones in real time.

V. SOFTWARE AND HARDWARE REQUIREMENTS

- Hardware Requirements
  - Main Controller:
    ESP32-S3 Module – Responsible for handling voice interactions and processing sensor data locally.
  - Sensors Integrated in System:
    - Ultrasonic Sensor – Detects frontal obstacles to avoid collisions.
    - Infrared (IR) Sensor – Identifies potholes and staircases for safe mobility.
    - Time-of-Flight (ToF) Sensor – Measures distances or elevation (height) accurately.
    - MPU6050 (IMU Sensor) – Tracks motion, posture, and detects sudden falls or imbalance.
    - GPS Module (NEO-6M) – Offers offline location tracking without internet access.
  - Wireless Communication Modules:
    ESP-NOW or NRF24L01 – Enables short-range, efficient communication between the shoe sensors and the pocket controller.
  - User Feedback Interfaces:
    - Bluetooth Neckband/Earbuds – Delivers verbal responses via A2DP Bluetooth connection.
    - Vibration Motor – Provides tactile (haptic) alerts for immediate awareness.
  - Power System:
    Rechargeable Lithium-Ion Battery (2500mAh – 5000mAh capacity) – Ensures portable, long-lasting operation.
  - Enclosure Design:
    Lightweight, shock-resistant housing to protect the ESP32 and components in both shoe and pocket device.

- Software Requirements
  - Development Environment:
    Programming in Python or MicroPython for prototyping and logic implementation.
  - Target Platform:
    ESP32-S3 microcontroller, running applications directly using ESP-IDF or Arduino Core.
  - Key Libraries and Frameworks:
    - Offline Wake Word & Command Recognition:
      TensorFlow Lite Micro, ESP-SR, or Picovoice (offline)
    - Speech-to-Text (Optional):
      Vosk API for local speech decoding
    - GPS Handling:
      TinyGPS++ library for parsing GPS data
      Kalman Filter to smooth noisy location inputs
    - Sensor Data Processing:
      Madgwick or Mahony Filters for accurate orientation from IMU
    - Wireless Communication:
      ESP-NOW for ESP-to-ESP networking
      Bluetooth BLE / A2DP for audio output to wireless earpieces

VI. FLOW OF DATA

1. Sensor Integration on Shoe-side ESP32
   - The ESP32 embedded in the smart shoe collects real-time data from multiple sensors:
     - Ultrasonic Module – Detects obstacles directly ahead.
     - IR or Line Tracking Sensor – Monitors surface conditions or follows lines.
     - MPU6050 IMU – Measures movement, posture changes, and detects falls.
   - These sensors are interfaced with the ESP32 using GPIO, I2C, or analog pins depending on their requirements.
   - The microcontroller processes the raw sensor data and formats it for transmission.
2. Power Supply Management
   - Both ESP32 units (in the shoe and in the pocket module) are powered using a 2000mAh Lithium-Ion battery.
   - The power setup ensures sufficient runtime for real-world applications without frequent charging.
3. Offline Communication via ESP-NOW
   - The smart shoe ESP32 sends sensor readings to the pocket ESP32 using ESP-NOW, a low-latency, energy-efficient, peer-to-peer wireless protocol.
   - This setup allows seamless offline communication, eliminating the need for Wi-Fi or internet connectivity.
4. Pocket Device Processing and GPS Integration
   - The Pocket Device ESP32 receives transmitted data and optionally processes input from a NEO-6M GPS module.
   - It intelligently interprets the user's context and command logic by combining GPS and sensor data to support mobility or safety instructions.
5. Voice Feedback via ESP-TTS
   - Based on the processed information, suitable voice responses (like *"Obstacle ahead"* or *"Route recalculating"*) are generated.
   - These textual messages are then converted to audible speech using the ESP-TTS engine.
6. Wireless Audio Output
   - Using the Bluetooth A2DP profile, the ESP32 streams the synthesized audio directly to:
     - A wireless neckband, or
     - A pair of Bluetooth earbuds for real-time user interaction.

System Loop Overview:

[Shoe Sensors] → [ESP32 in Shoe] → (ESP-NOW) → [Pocket ESP32 + GPS]

→ [Text Message] → [ESP-TTS Conversion] → [Bluetooth Earbuds Output]

Key Highlights:

- 100% Offline Operation – No dependence on the cloud or internet.
- Battery Efficient – Optimized for long usage with a single 2000mAh Li-ion cell.
- Real-time Feedback – Immediate voice alerts ensure safety and awareness.
- Modular Design – Both ESP32 units operate independently and can be replaced or reprogrammed as needed.

VII. RESULTS

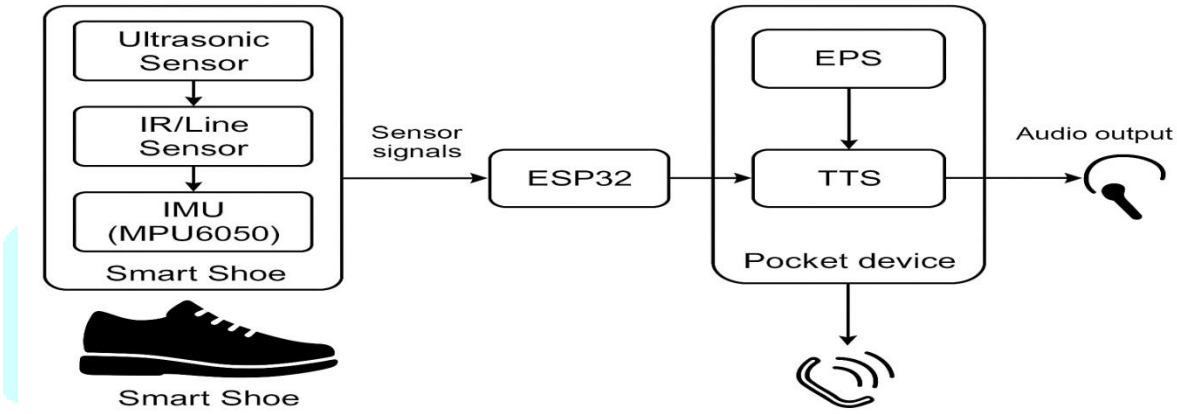

Fig. 2

1. Sensor Integration (Smart Shoe Layer)
   - The smart shoe is equipped with multiple sensors:
     - Infrared Sensor, MPU6050 IMU, Ultrasonic Module, and a Time-of-Flight Sensor
   - These components:
     - Monitor gait and motion patterns.
     - Identify obstacles.
     - Count steps.
     - Gauge proximity.
   - The collected sensor information is transmitted to the pocket module via ESP-NOW, a wireless protocol that works without Wi-Fi.
2. Microcontroller in the Shoe
   - The ESP32 microcontroller embedded in the shoe:
     - Reads data from all connected sensors.
     - Applies minimal local processing (like filtering or checking thresholds).
     - Sends the formatted data to the pocket-side ESP32 using ESP-NOW, allowing low-power, direct device-to-device communication.
3. Pocket Device Functions
   - The second ESP32 in the pocket acts as the central controller:
     - Receives real-time sensor data from the shoe via ESP-NOW.
   - Interfaces with:
     - An offline GPS module (e.g., NEO-6M) to provide geographic positioning.
     - TensorFlow Lite Micro for wake word recognition (e.g., “Hi Smart Shoe”).
     - ESP-SR to understand predefined voice commands without an internet connection.
     - ESP-TTS to synthesize voice responses from text.

4. Command Processing and Interpretation
    - When the wake word is heard:
        - The system activates listening mode.
        - It decodes user voice instructions like “Where am I?” or “Obstacle ahead?”.
    - At the same time:
        - Incoming sensor data is processed to offer real-time situational feedback (like object alerts or step count).
5. Audio Feedback System
    - The ESP32 sends synthesized speech (from ESP-TTS) via Bluetooth A2DP protocol to:
        - A neckband headset, or
        - Bluetooth-enabled earbuds.
    - This delivers clear audio alerts wirelessly, eliminating the need for wired audio output.
6. Power Management
    - Both ESP32 units (in the shoe and the pocket module) are powered by a 2000mAh lithium-ion battery.
    - Charging is handled using a TP4056 module, allowing USB-based safe and efficient battery recharging.
7. Completely Offline Functionality

- The entire system operates independently of the internet:
    - No Wi-Fi required.
    - No dependency on cloud servers.
    - Communication and processing happen locally within the two ESP32 devices.

VIII. CONCLUSION AND FUTURE SCOPE

The developed smart shoe architecture utilizes a combination of sensors and a compact ESP32-based pocket controller to deliver voice-guided support for individuals, particularly those with mobility or visual limitations. Through the integration of TensorFlow Lite Micro for detecting wake words, ESP-SR for offline command recognition, and ESP-TTS for audio response generation, the system offers completely offline, fast, and reliable performance. Data is transmitted from the shoe to the pocket module via ESP-NOW, while Bluetooth A2DP enables seamless audio delivery to a wireless headset or earbud, ensuring clear feedback in real-time. This modular, energy-efficient solution operates on a rechargeable Li-ion battery, promoting portability and low maintenance.

- Multilingual Interaction
    - Incorporate support for multiple languages to cater to a wider audience.
- AI-Based Navigation
    - Utilize machine learning algorithms for dynamic route planning and adaptive obstacle avoidance.
- Optional Cloud Backup
    - Provide optional cloud synchronization for data access by caregivers or healthcare providers.
- Mobile App Compatibility
    - Enable smartphone pairing to track user activity, health parameters, and location information.
- Customized Wake Word Models
    - Personalize wake word models to match the user's voice, improving detection reliability.
- Advanced Power Management
    - Introduce features like smart power control, extended runtime, and optional solar charging integration.
- Emergency Notification System
    - Add offline GSM or LoRa modules to trigger automatic alerts during emergencies like falls or sudden hazards.

IX. REFERENCES

1. Mischie, Matiu-Iovan, and Gasparesc (2018) discussed deploying Google Assistant functionalities on a Raspberry Pi platform, demonstrating how cost-effective hardware can enable voice-based interactions without traditional smartphones or smart speakers.
2. Dojchinovski, Ilievski, and Gusev (2019) developed an interactive healthcare system that incorporates voice assistant functionalities to support elderly or disabled users, enhancing home healthcare monitoring through voice automation

3. Vashistha and team (2019) proposed a personal voice-operated assistant, termed "Neo-bot," leveraging Raspberry Pi to execute user commands, showcasing its capabilities in embedded and aerospace technology applications.
4. Subhash et al. (2020) explored the integration of Artificial Intelligence in the development of voice assistants, focusing on secure and sustainable smart system trends, and highlighted AI's growing role in voice-based automation.
5. Klein et al. (2020) evaluated the quality of user experiences with voice assistants, emphasizing usability metrics and user satisfaction to improve interaction design in future assistant technologies