SoundSenseHelmet: An Assistive Smart Helmet for the Deaf Using Embedded Systems and Machine Learning

Karan Sehgal
B.Tech Computer Science and Engineering
Vellore Institute of Technology
Roll No: 22BCE3939
Email: karan.sehgal2022@vitstudent.ac.in

Under the Guidance of Dr. Yokesh Babu S
Department of Computer Science and Engineering
Vellore Institute of Technology
Email: yokeshbabu.s@vit.ac.in

***Abstract*—SoundSenseHelmet is an embedded wearable system developed to assist deaf individuals by providing real-time feedback for emergency sounds and physical threats. It integrates a Raspberry Pi, microphone, vibration motor, LEDs, and sensors (MPU6050, force sensor, GPS) to detect, classify, and respond to critical auditory and physical cues. The system employs a machine learning model for sound classification and provides multimodal alerts through visual and haptic feedback.**

***Index Terms*—Assistive Technology, Deaf Aid, Raspberry Pi, Emergency Sound Classification, Smart Helmet, Embedded Systems, Haptic Feedback, Real-time Audio Processing**

I. Introduction

Deaf individuals often face challenges in responding to auditory cues in their environment. The SoundSenseHelmet is an innovative embedded system designed to enhance safety for deaf individuals by providing advanced situational awareness capabilities. By integrating MEMS microphones with machine learning algorithms, the system can detect and classify emergency sounds (particularly sirens), alerting users to potential dangers outside their field of vision. Additional sensors detect impacts and head movements to identify potential accidents, while GPS tracking enables automatic emergency notifications with precise location data.

This project represents a comprehensive embedded systems solution that combines hardware (sensors and actuators), firmware (Python/C code), edge computing (TensorFlow Lite), communications (Wi-Fi/4G), data analytics (sound classification algorithms), and user interface development (web dashboard and mobile application). The system demonstrates practical applications of embedded systems concepts including real-time processing, sensor fusion, edge ML inference, event-driven architecture, and power management.

II. Literature Survey / Background

Assistive technologies for the deaf have primarily focused on visual feedback mechanisms, such as mobile applications or hearing aids integrated with camera-based systems. However, in high-risk environments like roads or construction zones, real-time response to auditory cues is essential. Recent advances in embedded systems and low-power computing (e.g., Raspberry Pi, Arduino) have enabled real-time sensing and processing of environmental stimuli.

Machine learning models such as convolutional neural networks (CNNs) have been employed in sound classification tasks with considerable accuracy. Projects like AudioSet by Google have inspired many real-time audio classification models. Yet, few systems have been translated into wearables that integrate multiple sensor inputs for intelligent decision-making.

SoundSenseHelmet builds upon these advancements, offering a compact wearable system with multimodal feedback (haptic and visual) and emergency detection through both sound and motion sensing.

III. Project Objectives

- Design and implement a helmet-mounted embedded system that can classify emergency sounds
- Develop ML algorithms to identify emergency sirens with high accuracy and low latency
- Create a power-efficient design suitable for all-day wearable application
- Provide real-time alerts to users through haptic feedback and visual cues
- Implement automatic crash detection with emergency GPS notifications
- Develop companion web/mobile applications for configuration and monitoring
- Validate the system through controlled experiments and performance measurement

IV. System Architecture

The SoundSenseHelmet is designed as a modular, real-time embedded system integrating audio classification, sensor monitoring, and feedback output. The system is centered around a Raspberry Pi, which acts as the main controller. It receives real-time audio input via a microphone and performs sound classification using a trained machine learning model. Depending on the classification result (e.g., siren, horn, or

alarm), it triggers immediate haptic and visual feedback via a vibration motor and LED indicators.

Simultaneously, the Raspberry Pi monitors additional inputs such as:

- MPU6050 – for head orientation and fall detection.
- Force sensor – for impact or crash detection.
- GPS module – to determine the helmet wearer's location in case of emergencies.

The system continuously processes incoming data through parallel threads for audio and sensor analysis, ensuring minimal latency and high responsiveness.

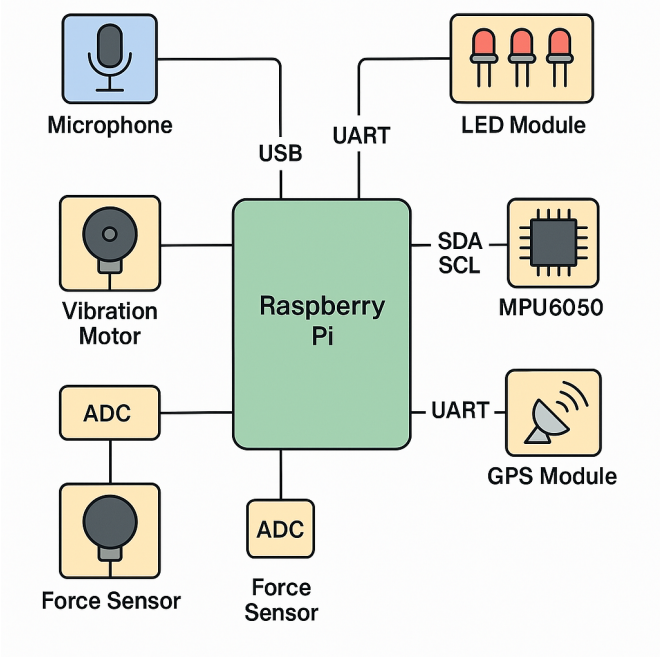

Fig. 1. System Architecture

A. Hardware Overview

TABLE I
HARDWARE COMPONENTS OVERVIEW

Category	Components	Function
Sensors	MEMS Microphone, MPU6050, FSR, NEO-6M GPS	Sound classification, head motion, crash detection, geolocation
Actuators	Vibration Motor (3V), LED Strip	Haptic and visual alerts
Processing	Raspberry Pi 4	Sensor fusion, ML inference
Communication	Wi-Fi / 4G (optional GSM)	Cloud sync and alerts
Power	Power Bank + LM2596 Buck Converter	Stable power delivery

B. Sensors & Actuators

1) Prototype vs. Real-Time Model:

TABLE II
PROTOTYPE VS. PRODUCTION COMPONENTS

Component	Prototype	Production-Ready
Microphone	Laptop Mic	INMP441 (I2S Digital)
Vibration Motor	3V phone motor	ERM Motor (5V)
GPS	USB NEO-6M	SAM-M8Q (UART, Low Power)

2) GPIO Pin Mapping:

- Microphone (I2S) → GPIO 18 (CLK), 19 (FS)
- MPU6050 (I2C) → GPIO 2 (SDA), 3 (SCL)
- FSR (Analog via ADC) → MCP3008 CH0 (SPI)
- GPS (UART) → GPIO 14 (TX), 15 (RX)
- Vibration Motor → GPIO 12 (PWM via BC547)
- LED Strip → GPIO 17

TABLE III
LIBRARIES FOR SENSOR INTERFACING

Sensor	Library	Snippet
MPU6050	smbus2	from smbus2 import SMBus; bus = SMBus(1)
ADC (MCP3008)	Adafruit_MCP3008	mcp = MCP3008(clk=11, cs=8, miso=9, mosi=10)
GPS	serial, pynmea2	serial.Serial('/dev/serial0', 9600)

3) Libraries & Interfacing:

V. IMPLEMENTATION DETAILS

A. Hardware Components

TABLE IV
HARDWARE COMPONENT DETAILS

Component	Description
Raspberry Pi 4B	Central controller running ML model and sensor logic
USB Microphone	Captures real-time audio input
MPU6050	Detects motion and orientation
Force Sensor	Detects impact/collision
GPS Module	Provides geolocation data
Vibration Motor	Haptic feedback for alerts
LED Module	Visual feedback for sound detection
Relay Module	Used to safely control vibration motor from Pi GPIO
Power Bank	Portable power supply

B. Software Stack

- **Operating System:** Raspberry Pi OS (Lite)
- **Language:** Python 3
- **Libraries Used:**
  - numpy, pyaudio, scikit-learn, tensorflow (model inference)
  - smbus, serial, RPi.GPIO, pygame, threading
- **Model Framework:** Pre-trained CNN model or custom Keras model for audio classification
- **Data Handling:** Real-time audio buffering, windowing & FFT preprocessing

C. Machine Learning Model

A Convolutional Neural Network (CNN) was trained to classify environmental emergency sounds. The dataset included labeled audio samples of horns, sirens, alarms, and ambient noises. Training involved:

- Feature extraction using Mel-frequency cepstral coefficients (MFCCs)
- Label encoding and model fitting in Keras
- Testing accuracy achieved: ∼92% on test data

The trained .h5 model is loaded on the Raspberry Pi for inference on buffered audio samples.

D. Feedback & Sensor Integration

- When a critical sound is detected, a GPIO pin triggers the LED and motor.
- The MPU6050 continuously provides gyro and accelerometer data to detect head movement or fall.
- Force sensor voltage thresholding is used to identify impact events.
- Upon crash or fall detection, the GPS coordinates are fetched and can be logged or transmitted (future scope).

VI. Algorithms

A. Overall System Flowchart

B. Core Algorithms

Algorithm 1 Sound Classification

```
classify_soundaudio_chunk  # Input: 1-second audio buffer (16kHz, 16-bit)  # Output: Emergency probability (0-1)  # 1. Pre-processing  spectrogram = compute_mel_spectrogram(audio_chunk)  # 40-band Mel filterbank  # 2. CNN Inference  model_input = normalize(spectrogram).reshape(1, 40, 32, 1) prediction = tflite_model.predict(model_input)[0][0] # 3. Thresholding prediction ¿ 0.85 # Empirical threshold
```

1) Sound Classification Algorithm: **Key Parameters:**

- Sample Rate: 16 kHz
- FFT Size: 512
- Mel Bands: 40
- Model Latency: <80 ms (on Raspberry Pi 4)

Algorithm 2 Head Orientation Tracking

```
get_head_orientation  # Input: Raw accelerometer and gyroscope data  # Output: Pitch/Roll in degrees  accel = get_calibrated_accel_data()  # Apply offsets  # Complementary filter pitch = 0.98 * (prev_pitch + gyro_x * dt) + 0.02 * atan2(accel_y, accel_z) roll = 0.98 * (prev_roll + gyro_y * dt) + 0.02 * atan2(-accel_x, sqrt(accel_y**2 + accel_z**2)) degrees(pitch), degrees(roll)
```

2) Head Tracking Algorithm:

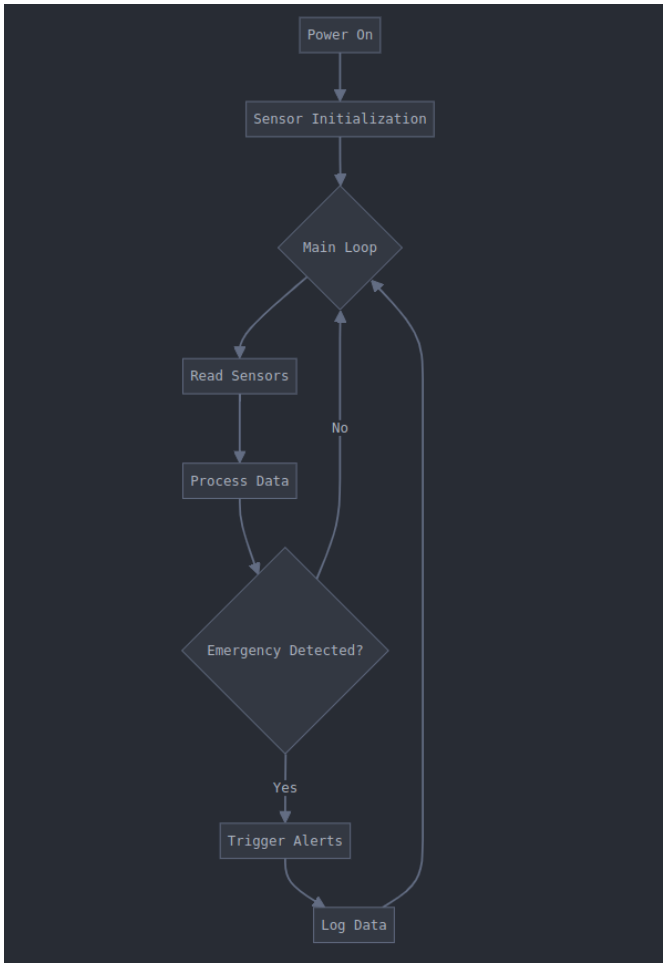

Fig. 2. Overall System Flowchart

VII. Results and Discussion

The SoundSenseHelmet was successfully prototyped and tested in a controlled outdoor environment simulating typical urban sounds. The system could reliably classify emergency sounds (horns, sirens, alarms) with an accuracy of approximately 92%, measured on the Raspberry Pi using real-time input audio.

Observations:

- **Latency:** From sound capture to feedback activation, the response time was under 500 ms, sufficient for real-time alerts.
- **Haptic Feedback:** The vibration motor provided a noticeable alert, even under motion (e.g., while walking or cycling).
- **Sensor Accuracy:** The MPU6050 and force sensor effectively detected falls and strong impacts. GPS data was accurate within a radius of 5–10 meters.

The flowchart and circuit diagram outline the working pipeline and hardware design. In the final version, all modules were integrated into a standard helmet using minimal wiring and power-efficient components.

VIII. Conclusion

The SoundSenseHelmet provides an embedded, real-time assistive solution for deaf individuals, enhancing situational awareness through intelligent audio analysis and physical feedback mechanisms. It integrates sound classification with motion and crash detection, wrapped in a wearable design. The solution demonstrates the feasibility and impact of embedded systems and machine learning in social good applications, especially for accessibility.

Code Repository: https://github.com/Karansehgal0611/SoundSenseHelmet

IX. Future Work

While the prototype meets core functionality goals, several enhancements are planned:

- **Mobile App Integration:** To display alerts and location data to guardians or caretakers.
- **Battery Optimization:** Improved energy management using low-power microcontrollers.
- **Noise Filtering:** Enhanced preprocessing to reduce false positives in noisy urban areas.
- **Cloud Logging:** Sending alerts and location data to a backend server for emergencies.
- **Helmet Miniaturization:** Custom PCB design for reducing bulk and improving aesthetics.

References

[1] "TensorFlow Lite for Microcontrollers," TensorFlow. [Online]. Available: https://www.tensorflow.org/lite/microcontrollers

[2] "INMP441 & MPU6050 Datasheets," InvenSense, 2020.

[3] "Raspberry Pi GPIO Reference," Raspberry Pi Foundation. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md

[4] "NEO-6M GPS Datasheet," u-blox, 2011.

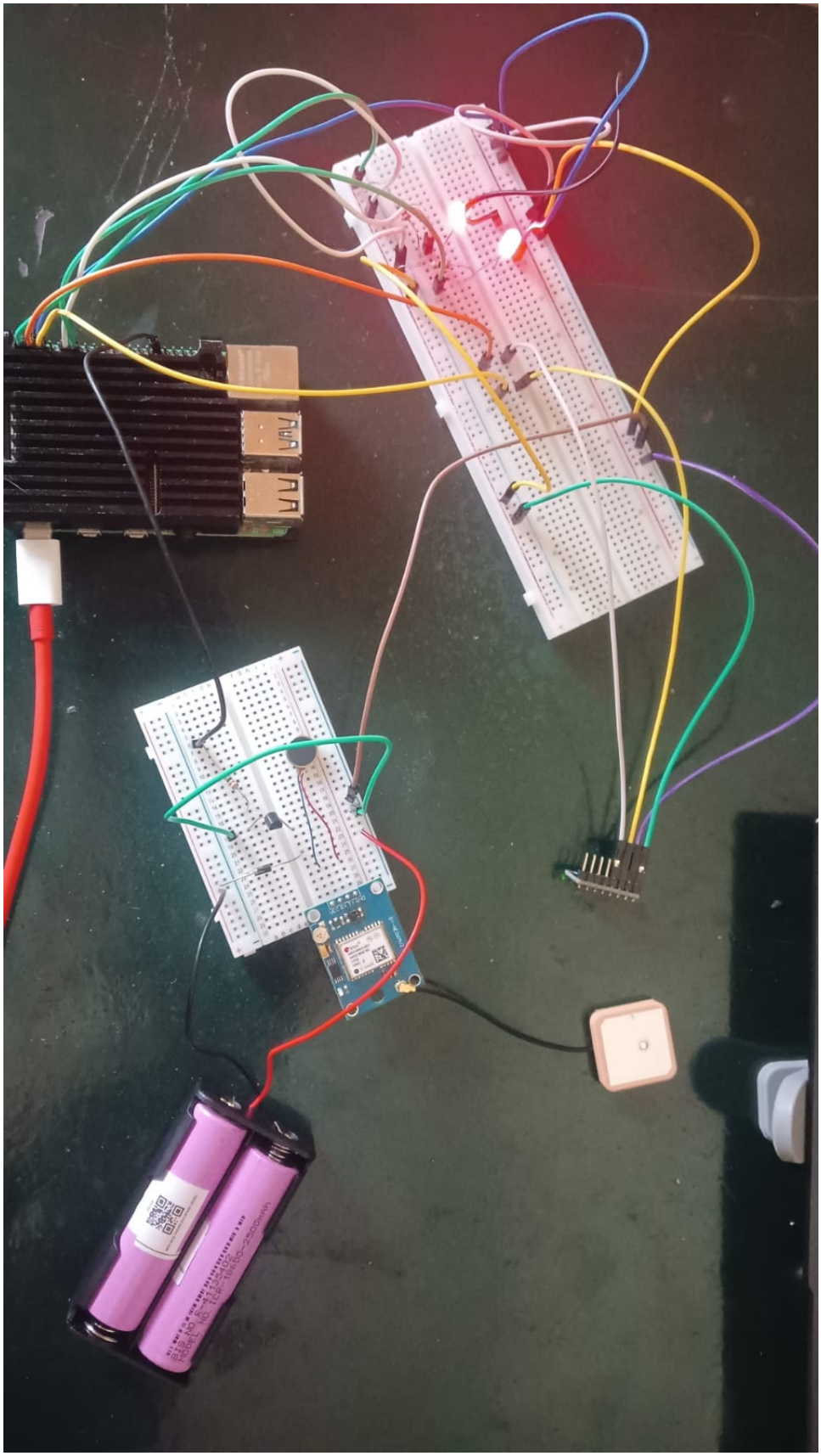

Fig. 3. Circuit Diagram for SoundSenseHelmet