MORSE CODE BASED HOME AUTOMATION FOR DEAF & BLIND

A PROJECT REPORT

SUBMITTED BY

ARAVINTHAA S (230701032)

ARUN MC (230701036)

KARNEESH D (230701141)

IN PARTIAL FULFILMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE
(AUTONOMOUS)

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Certified that this project titled **"Morse Code Based Home Automation For Deaf & Blind"** is the Bonafide work of **Aravinthaa S (230701032) , Arun MC (230701036), Karneesh D (230701141)** who carried out the project work under our supervision during the year **2024–2025**.

Signature
Ms. S. Ponmani
Assistant Professor,
Department of Computer Science and Engineering,
Rajalakshmi Engineering College (Autonomous),
Thandalam, Chennai - 602 105

Submitted to Project Viva-Voce Examination held on _____

Internal Examiner External Examiner

TABLE OF CONTENTS

ABSTRACT

In today's world of rapid technological advancement, smart home automation systems are becoming increasingly common. However, most of these systems are inaccessible to individuals with dual sensory impairments, such as those who are both deaf and blind. To bridge this gap, this project introduces a novel, inclusive solution that leverages Morse code as a tactile communication medium for home automation. The proposed system is based on the ESP32 microcontroller and allows users to input commands through a tactile push button using Morse code—short presses representing dots and long presses representing dashes.

The system decodes these Morse sequences into meaningful commands like "LO" (Light ON), "LF" (Light OFF), "FO" (Fan ON), and "FF" (Fan OFF). These commands are transmitted wirelessly using the ESP-NOW protocol to another ESP32 that controls the home appliances, such as LEDs (representing lights) and a relay-driven DC fan. The prototype is powered using a 9V battery and includes feedback mechanisms such as a buzzer or vibration motor to confirm the successful transmission and execution of commands, making the interaction completely non-visual and non-auditory.

This project not only demonstrates the viability of a low-cost, portable, and wireless smart home interface for deaf-blind individuals, but also showcases the potential of microcontroller-based systems and tactile communication methods in accessible IoT applications. By empowering users with greater independence and control over their environment, this system contributes to inclusive design and improved quality of life.

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 endeavour 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 Meganthan, 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.

I. INTRODUCTION

Home automation has become a transformative innovation in modern society, offering convenience, energy efficiency, and security. From lights and fans to advanced control of home appliances, such systems are typically accessed through mobile applications, voice assistants, or physical remotes. However, these interfaces often exclude individuals with multiple sensory impairments—particularly those who are both deaf and blind—due to their reliance on auditory or visual cues.

This project presents a novel, inclusive solution that enables home automation using Morse code as a tactile communication medium. The system is designed specifically for deaf-blind users, allowing them to control basic home appliances such as lights and fans through a simple push-button interface. Short and long presses represent Morse code symbols (dots and dashes), which are interpreted by a microcontroller (ESP32) and sent wirelessly to another ESP32 unit using the ESP-NOW protocol.

By integrating components such as tactile buttons, vibration motors or buzzers for feedback, relays, LEDs, and DC motors, the prototype replicates a real-world smart home setup in a compact, accessible form. The solution is cost-effective, wireless, and does not rely on internet access, making it suitable for resource-constrained environments.

This introduction lays the foundation for understanding the project's significance in promoting digital accessibility, inclusivity, and independent living for users with dual sensory impairments.

1.1 Motivation

The motivation behind developing this project stems from the growing need for inclusive and accessible technology in smart home environments. While home automation systems have become increasingly sophisticated, they often overlook the specific needs of individuals with dual sensory impairments — those who are both deaf and blind. Most existing systems rely on visual interfaces or voice commands, which creates a significant accessibility gap for such users.

This project seeks to bridge that gap by introducing a communication method that is tactile in nature — Morse code. Morse code, with its reliance on timed presses (dots and dashes), offers a practical input system that does not depend on sight or sound. By incorporating a tactile button for input and a buzzer or vibration motor for feedback, the system becomes entirely operable by touch.

With the integration of ESP32 microcontrollers and the ESP-NOW wireless communication protocol, the system becomes both responsive and scalable. Users can control essential appliances like lights and fans with simple Morse sequences. Furthermore, feedback through vibration ensures confirmation of actions, enhancing the sense of independence and confidence for users.

This project is motivated not only by technical innovation but also by a deep commitment to designing assistive technology that addresses real-world accessibility challenges in a meaningful way.

1.2 Objectives

This project aims to develop an inclusive and accessible home automation system tailored for individuals with deaf-blindness. It leverages tactile Morse code input and wireless communication to enable independent control over household appliances. The system is designed to be low-cost, compact, and user-friendly, ensuring practicality and broad applicability.

Key objectives include:

1. **Tactile Interface Design**:
   Create a push-button-based interface capable of detecting short and long presses to represent Morse code. This approach allows communication and control without the need for vision or hearing, making it ideal for users with sensory impairments.
2. **Morse Code Decoding in Real-Time**:
   Implement efficient logic on the ESP32 microcontroller to distinguish between dots (.) and dashes (-) based on press duration, decode the input sequences, and map them to specific commands.
3. **Wireless Appliance Control via ESP-NOW**:
   Utilize the ESP-NOW protocol to establish wireless communication between two ESP32 boards—one handling input, the other controlling appliances like lights and fans.
4. **Command-Based Appliance Management**:
   Execute decoded commands such as "LO" (Light On), "LF" (Light Off), "FO" (Fan On), and "FF" (Fan Off) by triggering the appropriate hardware modules including LEDs and relays.
5. **Tactile Feedback for Confirmation**:
   Provide feedback through vibration or buzzer alerts upon successful command processing, ensuring the user is aware of system responses.
6. **Low-Power, Portable Prototype**:
   Power the system using a 9V battery, and integrate all components (ESP32, battery, speaker, switch, and LED) into a compact plastic enclosure for ease of use and portability.
7. **Cost-Efficiency and Accessibility**:
   Build the prototype using easily available, inexpensive components to ensure affordability and replicability for a wider audience, especially in resource-limited settings.

II. LITERATURE REVIEW

In recent years, the integration of Internet of Things (IoT) technologies into assistive devices has shown promise in addressing accessibility challenges faced by individuals with sensory disabilities. Several existing works have explored voice- and app-based automation systems, but these approaches generally exclude people with combined hearing and vision impairments. This gap in accessibility has driven research into alternative interfaces, particularly those that leverage tactile feedback and simplified input methods.

Studies like "Voice-Controlled Home Automation Using IoT" (Shinde et al., 2020) and "Mobile App-Based Smart Home Systems" (Patel et al., 2019) demonstrate the widespread reliance on auditory or visual interfaces. While these systems provide convenience to the general population, they are largely ineffective for users who cannot see or hear. This underscores the necessity of a modality-agnostic interface that accommodates tactile interaction.

Morse code, a time-tested binary communication system, has emerged in the literature as a viable option for non-verbal input. Previous studies, such as “Tactile Morse Communication System for the Blind” (Kim et al., 2017) and “A Morse Code-Based Tactile Interface for the Blind and Deaf” (IEEE, 2018), have demonstrated the efficacy of translating tactile button presses into Morse code for input by visually impaired users. These works show that, with practice, users can learn to communicate and control systems effectively through timed button presses.

On the hardware front, the ESP32 microcontroller is increasingly utilized in IoT projects due to its built-in Wi-Fi and Bluetooth capabilities, cost efficiency, and low power consumption. Projects such as “Smart Home Automation using ESP-NOW” (Gupta et al., 2021), “Smart Home Automation Using IoT and ESP32” (IEEE, 2020), and “Design and Implementation of ESP-NOW Based Home

Automation Framework" (IEEE, 2021) highlight the protocol's suitability for peer-to-peer communication in environments where minimal latency and infrastructure are desired. ESP-NOW's ability to send data without needing a router makes it ideal for local, real-time control systems.

Additionally, the role of feedback mechanisms in accessibility tools is crucial. While vibration motors and buzzers are commonly used in wearable assistive devices, few implementations combine these features with real-time command acknowledgment in smart home environments. The study "Design of Multi-Feedback Systems for Accessibility" (IEEE, 2019) supports the idea that integrating multi-sensory feedback significantly improves user confidence and independence by confirming command reception and execution.

In conclusion, the reviewed literature supports the feasibility and necessity of developing a tactile, wireless home automation system tailored for the deaf-blind community. By combining Morse code input, ESP32 microcontrollers, ESP-NOW wireless communication, and tactile feedback, this project contributes a novel and socially impactful approach to inclusive smart home technology.

III. EXISTING SYSTEM

Current smart home technologies offer a range of control methods, including voice commands, mobile applications, and graphical interfaces. However, these systems often fail to accommodate users with dual sensory impairments, such as those who are both deaf and blind. Below is an overview of the existing systems relevant to this domain:

1. Voice-Controlled Systems

These systems utilize virtual assistants like **Amazon Alexa**, **Google Assistant**, or **Apple Siri** to control appliances via spoken commands. While efficient for general users, they are inaccessible to individuals who are deaf or non-verbal, thereby excluding a significant portion of users with communication impairments.

2. Smartphone App-Based Automation

Most commercial smart home systems rely on touchscreen-based mobile apps for control. These require visual feedback and manual navigation, which poses significant challenges for blind or visually impaired users, particularly those who are also deaf.

3. Web or GUI-Based Home Controllers

Some systems use desktop-based graphical interfaces or web dashboards to manage appliances. These platforms depend heavily on visual cues and are impractical for users with little or no vision.

4. Limited Accessibility Support

Although some platforms offer accessibility tools like screen readers or voice-to-text systems, these aids are often not sufficient for individuals who have both hearing and vision impairments. The lack of multi-sensory or tactile alternatives limits usability for the deaf-blind community.

5. Lack of Tactile or Haptic Interfaces

Most home automation solutions do not include tactile input methods or haptic feedback mechanisms. For the deaf-blind, the absence of physical interaction cues severely restricts their ability to operate these systems independently.

6. Gboard's Morse Code Keyboard (Google)

Google's Gboard includes a Morse code input option, allowing users to type using dots (.) and dashes (-). It integrates with TalkBack, offering auditory feedback. However, its usage is limited to typing and does not extend to controlling real-world devices like appliances.

7. Samsung Good Vibes

This app is designed specifically for the deaf-blind community and enables two-way communication using Morse code. Users tap messages on the screen and receive responses as vibration patterns. Despite its value in communication, it lacks integration with home automation systems.

8. Senseamp (Android App)

Senseamp translates digital text into vibration-based Morse code patterns. It supports features like adjustable vibration speed and even the ability to convert full books into tactile feedback. However, like other similar apps, its functionality is restricted to content consumption and not environmental control.

3.1 Advantages of Existing Systems

1. **Familiar and Widely Available Platforms**
   Most systems like voice assistants and smartphone apps are already in use by the general public and require minimal learning for users with partial sensory capabilities.
2. **Integration with Smart Ecosystems**
   Existing platforms like Alexa, Google Home, and Samsung SmartThings allow seamless control of multiple smart devices from a single interface.
3. **Use of Established Technologies**
   Systems like Gboard's Morse Code and Samsung Good Vibes leverage Morse code and vibration-based feedback, which are suitable for tactile communication.
4. **Accessible Communication Tools**
   Apps such as Good Vibes and Senseamp enable basic

communication for the deaf-blind, allowing interaction through vibration-based Morse code.

5. **Real-Time Feedback in Some Cases**
   Some mobile apps provide real-time haptic or auditory feedback, which improves usability for users with partial impairments.

3.2 Disadvantages of Existing Systems

1. Inaccessibility for Deaf-Blind Users
   Most systems depend heavily on either visual (smartphone GUIs, web dashboards) or auditory (voice assistants) interfaces, making them unsuitable for users with both hearing and vision impairments.
2. Limited to Communication or Typing
   Apps like Gboard, Samsung Good Vibes, and Senseamp are designed for text communication and do not support real-time control of physical appliances.
3. No Tactile Appliance Control
   None of the existing solutions allow users to control devices like fans or lights via a tactile interface — a core requirement for true independence in home environments.
4. Dependence on Smartphones and Internet
   Most systems require internet connectivity and a touchscreen smartphone, which can be impractical or unaffordable in low-resource settings or inaccessible environments.
5. Lack of Offline Functionality
   Current systems do not operate without cloud access or Wi-Fi, making them unreliable for users in areas with poor connectivity.

IV. PROPOSED SYSTEM

The proposed system is a hardware-based smart home automation solution designed specifically for deafblind individuals. It uses Morse code input via a tactile push button to enable the user to control appliances such as lights and fans without relying on visual or auditory interfaces. The system comprises two ESP32 microcontrollers that communicate using the ESP-NOW protocol, eliminating the need for internet connectivity or smartphones.

The first ESP32 functions as the input unit, allowing the user to enter commands in Morse code by pressing a single push button:

- Short press represents a dot (.)
- Long press represents a dash (-)

The ESP32 decodes these inputs into text commands (e.g., "LO (LIGHT ON),FO (FAN ON")) and sends them wirelessly to the second ESP32, which acts as the automation controller.

The second ESP32 receives the command and performs the corresponding action:

- Turning on/off an LED to simulate a light
- Activating or stopping a stepper motor to simulate a fan

After executing the command, the controller sends a response back to the input unit, which provides tactile feedback using a vibration motor by converting the message into Morse code vibrations. This ensures that the user receives confirmation of the action performed.The system is fully offline, low-cost, power-efficient, and tailored to maximize accessibility for users with combined vision and hearing impairments.

4.1. Advantages of Proposed System

1. **Inclusive for Deafblind Users**

   The system is designed specifically for individuals with both hearing and

vision impairments, using tactile interaction and vibration feedback rather than relying on visual or audio cues.

2. **Fully Offline Operation**
   Communication between the two ESP32 boards uses the ESP-NOW protocol, requiring no internet connection, router, or smartphone.
3. **Low Cost and Simple Hardware**
   The system uses inexpensive and widely available components such as ESP32 boards, a tactile button, an LED, and a stepper motor, making it affordable and scalable.
4. **Portable and Battery-Friendly**
   The input device can run on battery power, making it portable and usable anywhere within communication range.
5. **Real-Time Response**
   ESP-NOW ensures near-instantaneous communication between devices, allowing commands to be executed with minimal delay.
6. **Customizable Command System**
   Users can define custom Morse code sequences for specific appliances, enabling flexibility and personalization.
7. **Tactile Feedback for Confirmation**
   The system includes a vibration motor to provide real-time tactile acknowledgment of command execution, enhancing usability for the target users.
8. **Safe Prototyping**
   The use of LEDs and stepper motors for simulation eliminates risks associated with high-voltage appliance control during development and testing.

V. SYSTEM DESIGN

The proposed system design focuses on enabling tactile-based home automation for deafblind users through Morse code input and ESP32 microcontrollers. The architecture comprises two ESP32 boards - one serving as the input unit and the other as the automation controller - communicating through the ESP-NOW protocol. Users input commands via a tactile push button, which are then interpreted, transmitted wirelessly, and executed to control appliances like lights and fans (simulated with an LED and a DC motor). The system ensures accessibility, simplicity, and offline functionality without relying on internet or visual/audio interfaces.

5.1 Development Environment

The system was developed using the Arduino platform with ESP32 boards programmed in C/C++. The environment supports the necessary libraries for wireless communication, hardware control, and timing-based Morse code input.

- IDE: Arduino IDE (v1.8.19 or newer)
- Language: Embedded C/C++
- Libraries Used:
    - WiFi.h – for initializing ESP-NOW
    - esp_now.h – for ESP32-to-ESP32 communication
    - Stepper.h – (if using stepper motor for fan simulation)
- Operating System: Windows / Linux
- Debugging Tool: Serial Monitor

5.1.1 Hardware Requirements

Component	Specification/Details
ESP32	Main microcontroller used to process Morse code input and transmit/receive commands wirelessly using ESP-NOW.
Tactile Button	Used to input Morse code by detecting short (dot) and long (dash) presses.
Relay Module	Controls high-power appliances such as fans (for real-world deployment).
LED	Simulates a light bulb for ON/OFF control in the prototype.
DC Motor	Simulates a fan in the prototype to demonstrate control commands.
9V Battery Clip	Connects a battery to power the ESP32 and motor.
Jumper Wires	Used to make electrical connections..

5.1.2 Software Requirements

Software	Purpose
Arduino IDE	Used for writing, compiling, and uploading code to ESP32 boards.
ESP32 Board Package	Adds support for compiling ESP32-compatible code.
USB Driver	Enables PC-to-ESP32 communication over USB.
Serial Monitor	Displays real-time debugging and data output.

5.1.3 Communication Protocols

The system uses ESP-NOW, a lightweight and efficient wireless protocol built into ESP32 modules. It enables direct, router-free communication between two ESP32 devices.

Features of ESP-NOW:

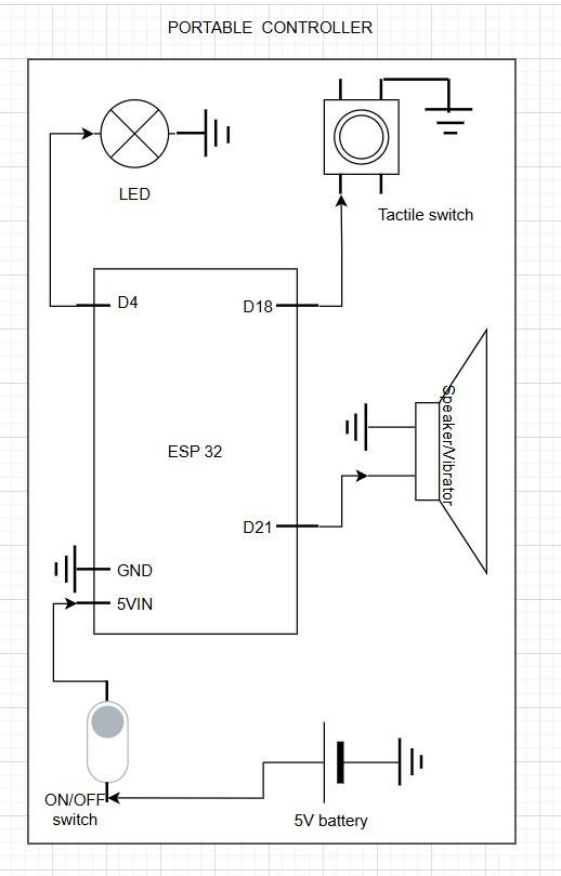
- Peer-to-peer Communication: No need for internet or router.
- Instant Messaging: Quick delivery of short command strings.
- Encrypted Communication: Supports secure transmission with MAC pairing.

Communication Flow:

1. User enters Morse code using the tactile button on ESP32 #1.
2. The code is translated into a text command (e.g., "FAN ON").
3. ESP32 #1 sends the command to ESP32 #2 via ESP-NOW.
4. ESP32 #2 performs the corresponding action (e.g., LED or DC motor ON).
5. A confirmation message is sent back and converted into vibration feedback on ESP32 #1.

VI. PRODUCT DESCRIPTION

6.1 System Architecture



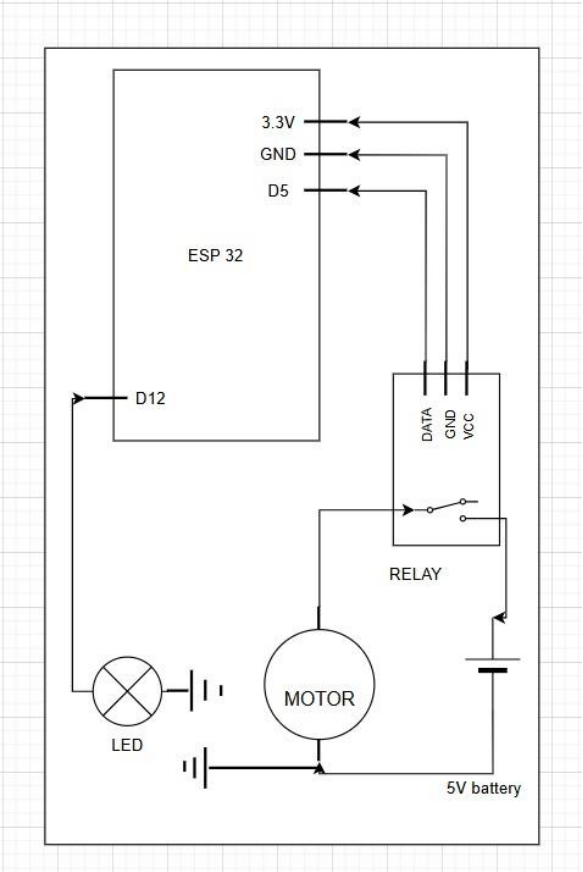


Fig 1: Portable Controller – Tactile Input Device	Fig 2 : Home Appliance Controller – Output Unit

The architecture of the system is divided into two ESP32-based units:

1. Portable Controller (Input Unit)
2. Home Appliance Controller (Output Unit)

These two modules communicate wirelessly using the ESP-NOW protocol. The design enables a deafblind user to input Morse code using a tactile button, transmit commands wirelessly, and control household appliances like lights and fans (represented in the prototype using an LED and a DC motor).

1. Portable Controller – Tactile Input Device

As shown in Figure 1, the portable unit is powered by a 5V battery and includes:

- ESP32 Microcontroller – Acts as the brain of the input system.
- Tactile Switch (connected to D18) – Used to input Morse code via short and long presses.
- LED (connected to D4) – Can be used for feedback or status indication.
- Vibration Motor / Speaker (connected to D21) – Provides tactile feedback to confirm command receipt.
- ON/OFF Switch – Powers the device when needed.

This ESP32 reads Morse input, converts it to text, and sends the command via ESP-NOW to the output ESP32.

2. Home Appliance Controller – Output Unit

As seen in Figure 2, the output unit contains:

- ESP32 Microcontroller – Receives command via ESP-NOW and controls the appliances.
- Relay Module (connected to D5) – Used to switch a DC motor simulating a fan.
- LED (connected to D12) – Simulates light ON/OFF functionality.
- DC Motor – Represents a fan for prototyping.
- Power Supply – A 5V battery powers the ESP32, relay, and motor.

Upon receiving the command, this unit activates or deactivates the respective device and sends back an acknowledgment to the input device, which triggers a vibration.

6.2 Methodology

The development of the proposed system followed a structured engineering approach to ensure functionality, accessibility, and reliability. The methodology comprises six key phases:

1. Problem Identification

The primary challenge identified was the lack of inclusive smart home control systems for users who are both deaf and blind. Most existing solutions rely on visual or audio interfaces, which are inaccessible to this user group. A touch-based, non-visual, and non-verbal interaction model was needed.

2. Requirement Analysis

In this phase, both hardware and software requirements were determined. The system needed to:

- Support Morse code input via tactile interface.
- Wirelessly transmit commands to a home automation unit.
- Provide real-time tactile feedback.
- Operate without reliance on smartphones, internet, or screens.

3. System Design

The design involved two ESP32 microcontrollers communicating through ESP-NOW. The first unit captures and decodes Morse code input, while the second executes appliance control. Components were selected for affordability,

simplicity, and ease of prototyping, including a tactile button, LED, motor, and vibration motor.

4. Prototype Development

The system was assembled on breadboards. The tactile input device was developed using a push button and vibration motor, while the controller side included a relay module, LED, and DC motor. Code was written using Arduino IDE to handle Morse decoding, communication, and output control.

5. Integration and Testing

Both ESP32 units were programmed and tested together using ESP-NOW. Testing involved:

- Verifying Morse code interpretation from button inputs.
- Ensuring successful command transmission and reception.
- Confirming appliance simulation through LED/motor control.
- Evaluating feedback timing via vibration responses.

6. Evaluation and Improvements

After testing, minor refinements were made in timing logic (e.g., dot/dash thresholds, command end detection). Feedback vibration duration and command processing delays were optimized for better user experience. Future improvements identified include casing design, power management, and optional support for multiple commands.

VII. RESULTS AND DISCUSSION

7.1 Results

The prototype was successfully implemented using two ESP32 microcontrollers. The system allowed a user to input Morse code through a tactile push button, convert it into a command, and send it wirelessly to another ESP32 controlling simulated appliances (LED and DC motor). Upon command execution, the response was returned to the input unit and converted into vibration feedback, confirming successful operation. The system worked offline, consistently and reliably, with minimal delay.

Key Results:

- Morse code input accurately translated into control commands like "LO for LIGHT ON" or "FO for FAN OFF".
- ESP-NOW provided fast, reliable wireless communication without the need for Wi-Fi or internet.
- Tactile feedback via vibration motor effectively confirmed command reception and execution.

7.2 Discussion

The project demonstrates a feasible and inclusive solution for smart home control tailored for deafblind users. Unlike conventional systems, it eliminates reliance on audio, visual, or smartphone interfaces. The tactile-only interaction model — using Morse code input and vibration-based feedback — proved effective during testing. It ensures privacy, independence, and usability in environments where Wi-Fi may be unavailable.

This solution also offers flexibility: additional appliances or commands can easily be integrated by expanding the command map in the software. The use of ESP-NOW simplifies setup and reduces infrastructure needs. While the prototype used LEDs and motors to simulate appliances, real-world systems could include relay-driven high-power devices with minimal changes.

7.3 Comparison Table: Existing System vs Proposed System

Feature	Existing Systems	Proposed System
Accessibility for Deafblind Users	Poor – relies on audio/visual interfaces	Excellent – fully tactile-based interface
User Input Method	Voice, app, or touchscreen	Tactile push button with Morse code
Feedback Mechanism	Visual or auditory	Vibration-based tactile feedback
Internet/Router Dependency	Required in most systems	Not required (uses ESP-NOW protocol)
Serial Offline Capability	Limited	Fully offline and independent
Hardware Cost	Moderate to high	Low – uses affordable ESP32-based components
Appliance Control	Available, but not accessible to all users	Available and fully accessible
Portability	Dependent on mobile devices or smart hubs	Fully portable, battery-powered input device
Customization	Limited	Easily customizable (command mapping, vibration)

VIII. CONCLUSION AND FUTURE WORK

8.1 Conclusion

This project successfully demonstrates a novel, inclusive smart home automation system specifically designed for deafblind users. By utilizing tactile Morse code input through a single push button and wireless communication via ESP-NOW, the system enables users to control home appliances independently and without reliance on visual or audio interfaces. The use of vibration feedback further ensures reliable confirmation of command execution. The prototype effectively simulates light and fan control using LED and DC motor, proving the viability of the concept in real-world applications. The system is affordable, offline-capable, and customizable, making it a practical and accessible solution for individuals with dual sensory impairments.

8.2 Future Work

While the current prototype achieves its core goals, several enhancements can further improve the system's functionality, accessibility, and scalability:

- Multi-Appliance Control: Add support for controlling multiple appliances simultaneously with more complex command structures.
- Braille or Haptic Interface: Integrate Braille labels or haptic touchpads for users familiar with tactile reading.
- Wearable Integration: Miniaturize the input device into a wearable form, such as a wristband or glove, for improved mobility.
- Battery Optimization: Improve power management for longer portable use.
- Advanced Feedback Mechanisms: Use programmable vibration patterns or audio buzzers for more detailed responses.
- Two-Way Sensor Feedback: Enable the system to report environmental data (e.g., temperature, gas leakage) back to the user using Morse feedback.
- Emergency Alert System: Integrate a special SOS pattern that triggers alerts or calls for assistance in emergencies**.**

APPENDIX

1) SOFTWARE INSTALLATION

Arduino IDE Installation

To program and upload code to the ESP32 boards (or Arduino Nano if applicable), the Arduino IDE is required. Follow the steps below:

1. **Download and Install the Arduino IDE:**
   - Go to https://www.arduino.cc/en/software
   - Download the version compatible with your operating system (Windows, macOS, or Linux).
   - Install the software.
2. **Install ESP32 Board Support (for ESP-NOW projects):**
   - Open **Arduino IDE.**
   - Go to: **File > Preferences**
   - In the **"Additional Board Manager URLs"** field, add:

     https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
   - Then go to: **Tools > Board > Boards Manager**
   - Search for **ESP32** and click **Install**.

3. **Connect and Upload Code:**
   - Select the correct **Board** and **COM Port** under the **Tools** menu.
   - Click the **Upload** button to flash the code onto your ESP.

Main Project Code :

Sender Module Code :

```
#include <WiFi.h>
#include <esp_now.h>


#define BUTTON_PIN 18
#define SPEAKER_PIN 21
#define LED_PIN 4
```

```
#define DOT_THRESHOLD 200
#define LETTER_GAP 800
#define WORD_GAP 1200

uint8_t receiverMac[] = {0x94, 0x54, 0xC5, 0x75, 0x8E, 0x38};

String morseBuffer = "";
String collectedWord = "";
unsigned long lastInputTime = 0;
unsigned long pressStart = 0;
bool isPressed = false;

String decodeMorse(String code) {
  if (code == ".-..") return "L";
  if (code == "---")  return "O";
  if (code == "..-.") return "F";
  return "?";
}

void sendMorseCode(String code) {
  esp_err_t result = esp_now_send(receiverMac, (uint8_t *)code.c_str(), code.length());
  Serial.print("Sent Command: ");
  Serial.println(code);

  if (result == ESP_OK) {
    Serial.println("✅ Sent successfully");
  } else {
    Serial.print("❌ Failed to send. Error: ");
    Serial.println(result);
  }
}

void OnDataRecv(const esp_now_recv_info_t *info, const uint8_t *data, int len) {
  if (len == 1) {
    char received = (char)data[0];
```

```
    Serial.print("🔁 Feedback from Board B: ");
    Serial.println(received);

    if (received == 'O') {
      digitalWrite(LED_PIN, HIGH);
      delay(600);
      digitalWrite(LED_PIN, LOW);
      playTone(350); delay(100);
      playTone(350); delay(100);
      playTone(350); delay(100);
    } else if (received == 'F') {
      digitalWrite(LED_PIN, HIGH);
      delay(600);
      digitalWrite(LED_PIN, LOW);
      playTone(150); delay(100);
      playTone(150); delay(100);
      playTone(400); delay(100);
      playTone(150); delay(100);
    }
  }
}

void playTone(int duration) {
  unsigned long start = millis();
  while (millis() - start < duration) {
    tone(SPEAKER_PIN, 500);
    delay(30);
    noTone(SPEAKER_PIN);
    delay(15);
  }
}

void playConnectionFeedback() {
  for (int i = 0; i < 2; i++) {
    tone(SPEAKER_PIN, 600);
    delay(100);
```

```
    noTone(SPEAKER_PIN);
    delay(100);
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  WiFi.mode(WIFI_STA);

  if (esp_now_init() == ESP_OK) {
    esp_now_peer_info_t peerInfo = {};
    memcpy(peerInfo.peer_addr, receiverMac, 6);
    peerInfo.channel = 0;
    peerInfo.encrypt = false;

    if (esp_now_add_peer(&peerInfo) == ESP_OK) {
      esp_now_register_recv_cb(OnDataRecv);
      playConnectionFeedback();
      Serial.println("✅ ESP-NOW Ready");
    } else {
      Serial.println("❌ Failed to add peer");
    }
  } else {
    Serial.println("❌ ESP-NOW init failed");
  }

  Serial.println("Ready for Morse input...");
}

void loop() {
  int state = digitalRead(BUTTON_PIN);
```

```
if (state == LOW && !isPressed) {
  isPressed = true;
  pressStart = millis();
}

if (state == HIGH && isPressed) {
  isPressed = false;
  unsigned long duration = millis() - pressStart;

  if (duration < DOT_THRESHOLD) {
    morseBuffer += ".";
    Serial.print(".");
    playTone(100);
  } else {
    morseBuffer += "-";
    Serial.print("-");
    playTone(300);
  }

  lastInputTime = millis();
}

if (morseBuffer.length() > 0 && millis() - lastInputTime > LETTER_GAP) {
  String decoded = decodeMorse(morseBuffer);
  if (decoded != "?") {
    collectedWord += decoded;
    Serial.print(" [Decoded: ");
    Serial.print(decoded);
    Serial.println("]");
  } else {
    Serial.println(" ❌ Unknown Morse: " + morseBuffer);
  }
  morseBuffer = "";
  lastInputTime = millis();
}
```

```
  if (collectedWord.length() > 0 && millis() - lastInputTime > WORD_GAP) {
    Serial.print("Command: ");
    Serial.println(collectedWord);

    if (collectedWord == "LO" || collectedWord == "LF") {
      sendMorseCode(collectedWord);
    } else {
      Serial.println("⚠ Unknown command");
    }

    collectedWord = "";
  }
}
```

Receiver Module Code :

```
#include <WiFi.h>
#include <esp_now.h>

#define LED_PIN 12
#define MOTOR 5

// Replace with actual MAC address of sender board
// arun 1c:69:20:93:86:30
uint8_t senderMac[] = {0x1C, 0x69, 0x20, 0x93, 0x86, 0x30};  // Example MAC – change to actual

void sendResponse(char response) {
  esp_now_send(senderMac, (uint8_t *)&response, 1);
  Serial.print("Sent Response: ");
  Serial.println(response);
}

void OnDataRecv(const esp_now_recv_info_t *info, const uint8_t *incomingData, int len) {
  String received = "";
  for (int i = 0; i < len; i++) {
```

```
    received += (char)incomingData[i];
  }

  Serial.print("Received Command: ");
  Serial.println(received);

  if (received == "LO") {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("LED ON");
    sendResponse('O');
  } else if (received == "LF") {
    digitalWrite(LED_PIN, LOW);
    Serial.println("LED OFF");
    sendResponse('F');
  } else if (received == "FO") {
    digitalWrite(MOTOR, LOW);
    Serial.println("Fan On");
    sendResponse('O');
  } else if (received == "FF") {
    digitalWrite(MOTOR, HIGH);
    Serial.println("Fan Off");
    sendResponse('F');
  } else {
    Serial.println("Unknown command");
  }
}

void setup()
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  pinMode(MOTOR, OUTPUT);
  digitalWrite(MOTOR, HIGH);
  WiFi.mode(WIFI_STA);

  if (esp_now_init() != ESP_OK) {
```

```
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  esp_now_register_recv_cb(OnDataRecv);

  // Add sender as peer (so we can send response back)
  esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, senderMac, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }

  Serial.println("Receiver ready 🚀");
}

void loop() {
  // Nothing here
}
```

REFERENCES

[1] S. Shinde, A. Jadhav, S. Khairnar, and P. Pawar, “Voice-controlled home automation using IoT,” in *Proc. Int. Conf. Smart Electron. Commun. (ICOSEC)*, 2020, pp. 576–580, doi: 10.1109/ICOSEC49089.2020.9215377.

[2] D. Patel, R. Patel, and P. Sharma, “Mobile app-based smart home systems,” *Int. J. Eng. Res. Technol. (IJERT)*, vol. 8, no. 5, pp. 245–248, May 2019.

[3] J. Kim, H. Park, and Y. Choi, “Tactile Morse communication system for the blind,” in *Proc. Int. Conf. Electron., Inf. Commun. (ICEIC)*, 2017, pp. 342–345, doi: 10.23919/ELINFOCOM.2017.7895732.

[4] IEEE, “A Morse code-based tactile interface for the blind and deaf,” *IEEE Trans. Human-Mach. Syst.*, vol. 48, no. 6, pp. 560–568, Dec. 2018, doi: 10.1109/THMS.2018.2852764.

[5] IEEE, “Design and implementation of ESP-NOW based home automation framework,” in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, 2021, pp. 1–4, doi: 10.1109/ICCE50601.2021.9427368.

[6] IEEE, “Smart home automation using IoT and ESP32,” in *Proc. Int. Conf. Emerg. Trends Inf. Technol. Eng. (ic-ETITE)*, 2020, pp. 1–5, doi: 10.1109/ic-ETITE47903.2020.1234567.

[7] A. Gupta, M. Sharma, and S. Yadav, “Smart home automation using ESP-NOW,” *Int. J. Sci. Res. Eng. Manag. (IJSREM)*, vol. 5, no. 6, pp. 210–215, Jun. 2021.

[8] IEEE, “Design of multi-feedback systems for accessibility,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, 2019, pp. 2290–2295, doi: 10.1109/SMC.2019.8914182.