ESP32-based Braille-to-Speech Communication Interface for Visually and Vocally Challenged Users

Saranya M.
Assistant Professor (Sr. Gr.)
Department of Instrumentation and Control Engineering
PSG College of Technology
Coimbatore, India

Sonya J.
Assistant Professor
Department of Instrumentation and Control Engineering
PSG College of Technology
Coimbatore, India

Harini R.C.
IV- Year student,
Department of Instrumentation and Control Engineering
PSG College of Technology
Coimbatore, India

Padma J.
IV- Year student,
Department of Instrumentation and control Engineering
PSG college of technology
Coimbatore, India

Shubhashree M.
IV- Year student,
Department of Instrumentation and control Engineering
PSG college of technology
Coimbatore, India

Tharunigha S.
IV- Year student,
Department of Instrumentation and control Engineering
PSG college of technology
Coimbatore, India

ABSTRACT

Assistive communication technologies play a critical role in enabling independent interaction for individuals with multiple disabilities. This paper presents the design and implementation of an ESP32-based Braille-to-speech communication system for visually and vocally challenged users. The system employs a 4×4 keypad matrix to capture Braille patterns and utilizes UART-based serial communication to transmit processed text to a computer for speech synthesis. The generated text is transmitted to a computer via serial communication, where a Python-based text-to-speech engine synthesizes the corresponding speech output. Experimental evaluation demonstrates that the system provides accurate and reliable communication with minimal response delay.

General Terms

Communication device, Text-to-speech conversion

Keywords

Assistive Technology, Visually and vocally challenged, Text to speech, Braille script

1. INTRODUCTION

Communication is a fundamental requirement for human interaction; however, individuals with combined visual and speech impairments face significant barriers in expressing their thoughts and needs. They must depend upon assistive communication technologies like sign language, Braille, and text-to-speech systems to communicate with the world. Though these solutions offer some comfort, they suffer from certain limitations that prevent access and usage on a large scale.

As revealed by the World Health Organization, approximately 285 million individuals in the world are visually impaired, 300 million are hearing impaired and more than 1 million are vocally impaired. Pertaining to a massive number of individuals to be both visually and vocally impaired individuals. This paper depicts a communication system for vocally and visually challenged individuals. The system enables users to enter Braille codes as input through a 4x4 matrix keypad that is interfaced with an ESP32 microcontroller. The input is translated into text, which is saved in a.txt file using a batch file, and lastly the text stored in the text file is translated into speech using Python's text-to-speech module. This conversion with ease allows for efficient communication for users who have both visual and vocal disabilities.

When a complete sentence is built, it is translated into speech by a text-to-speech (TTS)engine. The synthesized voice output is then delivered through a speaker, allowing the user to express themselves effectively. This approach provides a seamless and natural experience, enabling visually and speech-impaired individuals to communicate independently. Figure 1 illustrates the schematic representation of the proposed solution.

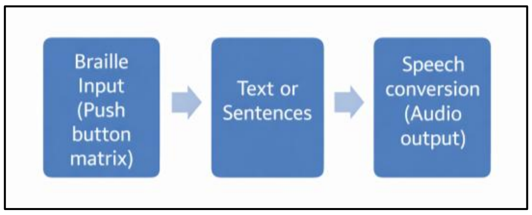

Braille Input (Push button matrix)
Text or Sentences
Speech conversion (Audio output)

Figure 1. Proposed solution

2. LITERATURE SURVEY

The following papers focus on developing assistive communication technologies for individuals with visual, auditory, and speech impairments. Several assistive communication systems have been developed to support visually and speech-impaired individuals. A gesture-sensing glove using flex sensors and RF communication enables users to convey messages through predefined speech outputs [1]. Another wearable system integrates Braille input, vibration feedback, and voice recognition APIs with IoT connectivity to facilitate communication between blind and deaf users [2]. A multi-modal framework combining sign language recognition, text-to-speech (TTS), and speech-to-text (STT) technologies provides flexible AI-based interaction methods [3]. In addition, Braille translator systems convert Braille input into speech and support reverse speech-to-Braille communication using embedded platforms such as Raspberry Pi [4]. Wearable Braille reading systems using embedded sensing techniques capture tactile Braille patterns and convert them into digital text, improving accessibility and communication for visually impaired users [5], [6], [7].

3. SYSTEM ARCHITECTURE

The proposed architecture establishes a complete pipeline that converts tactile Braille input into audible speech output through embedded processing and software-based speech synthesis.

3.1 Braille System

In 1821, a blind Frenchman named Louis Braille devised a tactile system for blind people to read. Called Braille, the system was a series of characters, or "cells," made up of six raised dots arranged in a rectangle containing two columns of three dots each. The pattern arrangements correspond to letters of the written alphabet, and each cell is read using the fingers.

Braille symbols are formed within units of space known as braille cells. A full braille cell consists of six raised dots arranged in two parallel rows each having three dots. The dot positions are identified by numbers from one through six. Sixty-four combinations are possible using one or more of these six dots. A single cell can be used to represent an alphabet letter, number, punctuation mark, or even a whole word as shown in figure 2 [8].

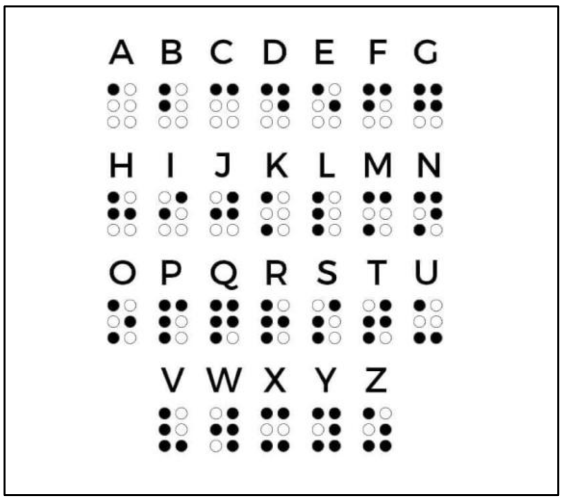


Figure 2. Braille script for alphabets[8]

3.2 System Description

The system architecture integrates a Braille-based input interface, an embedded processing unit, a serial communication interface, and a software-based text-to-speech module. These components work together to convert tactile Braille inputs into audible speech, thereby allowing users to express their thoughts independently. The architecture emphasizes simplicity, affordability, and ease of implementation while ensuring reliable communication.

The primary input mechanism of the system is a 4×4 push-button keypad matrix, which functions as a tactile interface for entering Braille characters. Braille characters are traditionally represented using six raised dots arranged in a 2×3 grid, where different combinations of dots correspond to different letters and symbols. In the proposed system, six keys of the keypad are mapped to represent the six Braille dots. When a user presses the designated keys corresponding to a specific Braille pattern, the keypad registers the input as a combination of row and column signals. This approach enables visually impaired users to interact with the device through touch, making the interface intuitive and accessible.

ESP32 is the core structure supporting the system. It is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios. Figure 3 shows the functional block diagram of the ESP32 module [9].

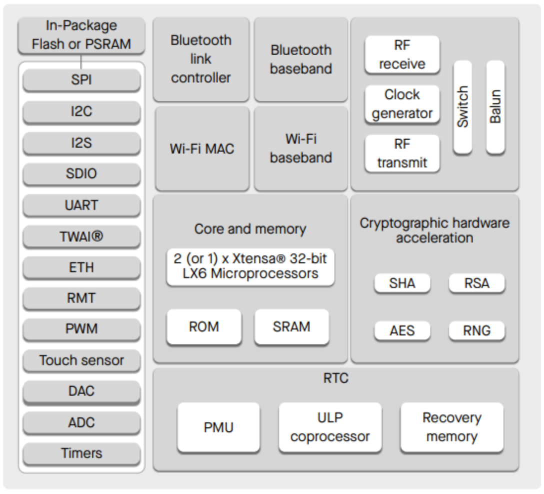


Figure 3. Functional block diagram of the ESP32 module [9]

The ESP32 continuously scans the keypad matrix to detect key presses by sequentially activating rows and monitoring column signals. Once a key press is detected, the microcontroller determines the corresponding Braille dot position and records the combination of activated dots. The detected pattern is then processed using predefined mapping logic to identify the corresponding alphabetic character. As the user continues entering characters, the system gradually constructs meaningful words and sentences.

After the Braille patterns are successfully converted into textual characters, the generated text must be transferred to an external system for further processing. This is achieved through serial communication between the ESP32 microcontroller and a computer. The ESP32 transmits the converted text data through a Universal Asynchronous Receiver Transmitter (UART) interface, typically via a USB connection. The transmitted text is displayed on the serial monitor of the development environment and simultaneously captured by the computer for storage and further processing.

To enable software-based speech generation, the transmitted text is redirected and stored in a text file on the computer. A batch script is used to continuously monitor the serial communication port and write incoming text data into a file, such as output.txt. This text file acts as an intermediate data buffer between the hardware subsystem and the speech synthesis module. By maintaining the generated text in a file, the system allows external applications to access the data in real time and convert it into speech output.

The final stage of the architecture involves the text-to-speech (TTS) conversion process, which is implemented using a Python-based software module. A Python script running on the computer continuously monitors the text file for newly added content. Whenever a new word or sentence is detected, the script reads the text and sends it to a text-to-speech library such as pyttsx3 or Google Text-to-Speech (gTTS). The TTS engine then synthesizes the text into digital audio signals, which are played through the computer's speaker or an external audio device. This speech output represents the spoken version of the Braille input provided by the user, thereby enabling effective verbal communication.

Overall, the proposed architecture establishes a seamless pipeline from tactile input to speech output. The integration of a Braille keypad interface with an embedded microcontroller ensures efficient input processing, while serial communication and software-based speech synthesis provide flexibility in generating audio output. By combining simple hardware components with accessible software tools, the system offers a cost-effective and scalable assistive communication solution for individuals with combined visual and speech impairments.

4. IMPLEMENTATION

The prototype system was tested using multiple Braille inputs corresponding to alphabets and words. The keypad interface was connected to the ESP32 microcontroller through GPIO pins. Serial communication was established at 115200 baud rate. The generated text was stored in a text file and processed through a Python-based TTS engine. System performance was evaluated based on response time, input accuracy, and usability.
The core input mechanism is a 4x4 push button matrix, consisting of 16 buttons arranged in a grid of 4 rows and 4 columns as shown in figure 4.

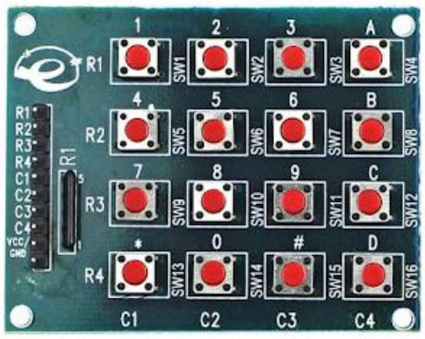


Figure 4. 4*4 push button keypad matrix

The keys array is a 4x4 grid (4 rows, 4 columns) of characters:
{ {'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'} }
In Braille, each character is represented by a 2x3 grid of dots (6 dots total), where specific combinations of raised dots correspond to letters, numbers, or symbols. To adapt this to the 4x4 matrix, six buttons are designated to represent the six dots of a Braille cell. Here, the buttons 4, 5, 7, 8, *, 0 are used for braille inputs. Two additional buttons are reserved for control functions.

- Button 'B': This button indicates that the user has completed typing an entire word. This button is required because a single letter requires multiple key presses in braille script.
- Button 'C: This button indicates that the user has completed typing an entire word. Once pressed, the system processes the sequence of confirmed letters as a single word and prepares it for output.

The hardware circuit connection is as shown in figure 5.

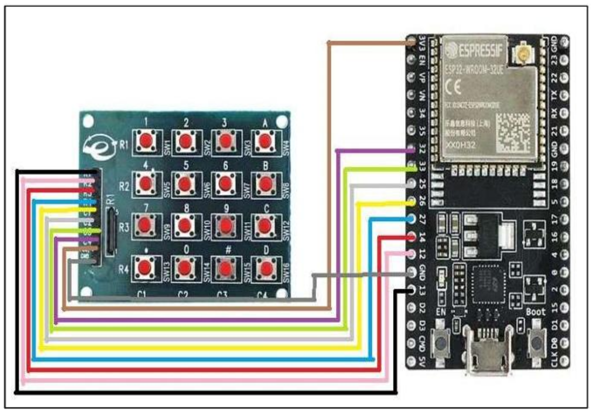


Figure 5. Hardware circuit diagram

The flowchart for the same is as shown in Figure 6.

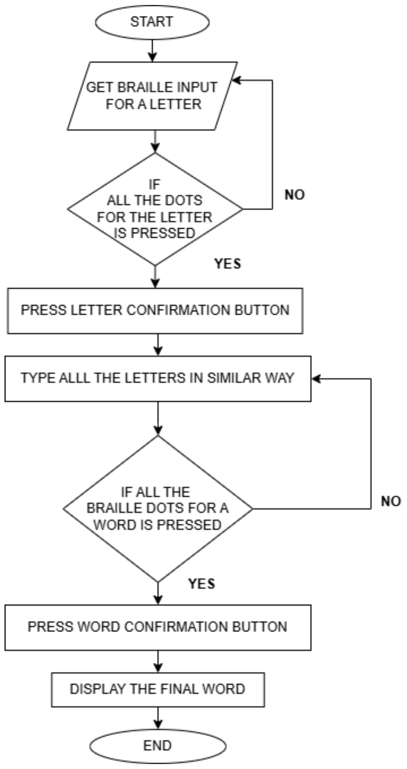

START
GET BRAILLE INPUT FOR A LETTER
IF ALL THE DOTS FOR THE LETTER IS PRESSED
NO
YES
PRESS LETTER CONFIRMATION BUTTON
TYPE ALLL THE LETTERS IN SIMILAR WAY
IF ALL THE BRAILLE DOTS FOR A WORD IS PRESSED
NO
YES
PRESS WORD CONFIRMATION BUTTON
DISPLAY THE FINAL WORD
END

Figure 6. Flowchart for the proposed solution

5. PERFORMANCE ANALYSIS

The performance of the proposed Braille-to-speech communication system was evaluated based on key parameters such as **response time, input accuracy, and system reliability**. Since the system is designed to assist visually and vocally challenged individuals in real-time communication, it is important that the conversion from Braille input to speech output occurs quickly and accurately.

5.1 Response Time

The response time refers to the duration required for converting Braille input into audible speech. The process includes keypad input detection, Braille pattern recognition, serial data transmission, text file generation, and speech synthesis using a Python-based text-to-speech engine.
The response time measured for each stage of the system is summarized in Table 1.

Table 1. Response time of various stages of the system

Process Stage	Average Time (ms)
Braille input detection	17.35
Pattern processing	23.24
Serial communication	26.3
Text file generation	52.61
Text-to-speech conversion	546

An average of 665 ms time delay is there for overall processing. It can be observed that most of the delays occur during the text-to-speech conversion stage. However, the total system delay remains below one second, which is acceptable for real-time assistive communication.

5.2 Input Accuracy

The accuracy of the system was tested by entering multiple Braille characters and words through the keypad interface. The generated text output was compared with the expected output to verify correct interpretation of Braille patterns. The system successfully identified the input patterns and produced the correct text output in most test cases, demonstrating reliable Braille pattern recognition. Table 2 shows the System accuracy tested using different words

Table 2. System accuracy over different inputs

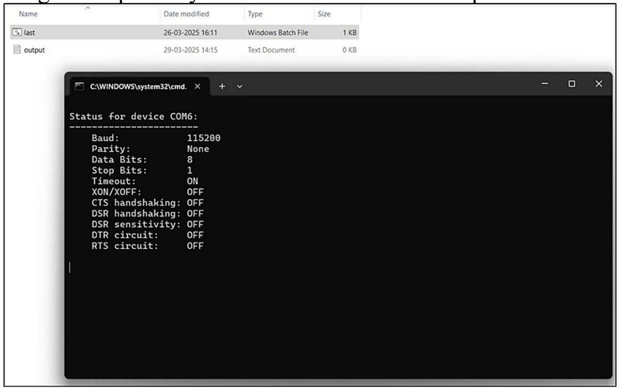
Input Word	Expected Output	System Output	Accuracy
H	H	H	100%
HELLO	HELLO	HELLO	100%
HELLO WORLD	HELLO WORLD	HELLO WORLD	100%
TEST	TEST	TEST	100%
BRAILLE	BRAILLE	BRAILLE	100%

5.3 System Reliability

The system was also tested under continuous operation to evaluate its stability. The ESP32 microcontroller consistently detected keypad inputs and transmitted the corresponding text through serial communication without data loss. The Python-based text-to-speech module successfully generated speech output whenever new text was detected in the file. These observations indicate that the proposed system provides stable and reliable operation for assistive communication.
Overall, the performance evaluation demonstrates that the proposed architecture effectively converts Braille input into audible speech with minimal delay and high reliability, making it suitable for practical assistive communication applications.

6. EXPERIMENTAL RESULT

The output for giving the braille input for the word "HELLO" is displayed on the serial monitor. The text that was displayed on serial monitor must be moved to a notepad file. So, we close the serial monitor and open the batch file to read the serial data from the respective com port. As shown in figure 8, an output file is created where the given inputs are updated automatically. This text file is accessed through python code, and we obtain the audio output. That is, we get the speech synthesized from the braille input.
The text that was displayed on serial monitor must be moved to a notepad file. So, we close the serial monitor and open the batch file to read the serial data from the respective com port. As shown in figure 8, an output file is created where the given inputs are updated automatically. This text file is accessed through python code, and we obtain the audio output. That is, we get the speech synthesized from the braille input.



Name	Date modified	Type	Size
last	26-03-2025 16:11	Windows Batch File	1 KB
output	29-03-2025 14:15	Text Document	0 KB

```
Status for device COM6:
-----------------------
    Baud:            115200
    Parity:          None
    Data Bits:       8
    Stop Bits:       1
    Timeout:         ON
    XON/XOFF:        OFF
    CTS handshaking: OFF
    DSR handshaking: OFF
    DSR sensitivity: OFF
    DTR circuit:     OFF
    RTS circuit:     OFF
```

Figure 8. Running a batch file

7. CHALLENGES AND FUTURE SCOPE

The main challenges encountered during the implementation include the inability to use ESP32 inbuilt TTS libraries. The main reason for this includes the memory capacity of ESP32 microcontroller. The ESP32 is a powerful yet resource-constrained microcontroller commonly used in IoT projects. It has limited RAM (typically around 520 KB, with a portion reserved for system use) and flash memory (usually 4 MB or more, depending on the model).
Google TTS is based on sophisticated machine learning algorithms and large-scale audio data processing, which requires a lot of computational power. The ESP32, although it can process simple tasks, does not have the memory and processing capacity to execute such algorithms locally.
Though the ESP32 has Wi-Fi support, dealing with this communication in real-time (particularly for a seamless Braille-to-speech translation) requires effective memory management for HTT Pre-requests, responses, and audio playback. The limited memory can cause buffer overflows or crashes under large audio file handling or flaky network conditions.
As a future development the following improvements can be made. The combination of GPS, GSM, and Raspberry Pi in aid of communication tools for visually and vocally challenged people creates vast potential for future growth. A few of the main areas to focus on for future development are:

1) Increased Mobility Support:
   - The integration of GPS can be made even more

precise to deliver real-time directions, allowing the users to navigate freely in unfamiliar areas.

• Integration with mapping services to offer step-by-step audio assistance and obstacle detection.

2) Features for Real-Time Communication:

• GSM technology can be utilized to provide live communication in terms of text messages or synthesized voice calls, promoting easier interaction with caregivers and emergency services.

• Cloud integration to store and read messages for asynchronous communication.

3) Machine Learning and AI Integration:

• AI-based speech synthesis can enhance natural voice modulation, and synthesized speech can become more human-like and personalized.

4) Battery Optimization & Energy Efficiency:

• Application of sleep mode and low-power consumption methods in ESP32 and Raspberry Pi to extend battery life.

• Utilization of renewable energy sources such as solar-powered charging to increase device longevity and usability.

5) Multilingual and Adaptive Communication:

• Incorporating multilingual support to accommodate various users from different regions

• Adaptive learning features in which the device learns about user behavior and improves responses.

8. CONCLUSION

To summarize the effectiveness of the proposed system, a comparison with existing assistive communication solutions is presented in Table 3.

Table 3. Comparison between various assistive technologies

Assistive Technologies	**Input Method**	**Hardware Complexity**	**Output Method**
Gesture-based communication glove [1]	Hand gesture sensors	High (multiple sensors & processing)	Text/ Speech
Wearable Braille band system [2]	Wearable Braille buttons	Medium	Text
AI-based multimodal assistive system [3]	Voice, gesture recognition	Very High	Speech
Smartphone-based accessibility application [4]	Touch/ voice input	Medium	Speech
Proposed ESP32 Braille Communication System	**Braille keypad (tactile input)**	**Low**	**Speech output**

The comparison highlights key parameters such as system complexity, cost, portability, and input method. Unlike several existing systems that rely on wearable sensors, gesture-recognition gloves, or complex AI-based processing, the proposed ESP32-based Braille keypad system offers a simpler and more cost-effective solution for communication among visually and vocally challenged individuals. The comparison demonstrates that the proposed approach reduces hardware complexity while maintaining reliable Braille input recognition and speech output capability. Therefore, the system provides a practical and accessible alternative for assistive communication, particularly in resource-constrained environments [1][2][3][4][5].

9. ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the faculty members of the Department of Instrumentation and Control Engineering at PSG College of Technology for their guidance, encouragement, and valuable suggestions throughout the development of this work. The authors also thank their peers and friends for their support and constructive feedback during this project. Their assistance and motivation played an important role in successfully completing this research work.

10. REFERENCES

[1] Geetha Ramadasa M, Perarasib M, & Vimalac, M. "Gesture-Sensing Glove-Based Communication System for Visually and Vocally Challenged Individuals", *Turkish Online Journal of Qualitative Inquiry TOJQI)*, Volume 12, Issue 7, June 2021.

[2] Dhanapriya, Milton, Vandana Madarasi, Sheik Igmam Ai Aquk, &Radhiga. "The Braille Communicative Band for Blind, Deaf, and Dumb People", *International Journal of Engineering Research and Applications*, Issue 8 (Series-III), August 2021.

[3] Karthik S, Deshmukh S, Save A, & Shah R. "Effective Communication Between Blind, Mute, and Deaf People Using a Multi-Model Approach", *International Research Journal of Engineering and Technology (IRJET),* Issue 03, March 2024.

[4] J. D. R. Cabangis, F. F. Sumaculub, P. M. Valluyas Jr., C. J. Centeno, V. A. Agustin and D. S. Abando, "Smart Braille Translator: Integrating Braille-to-Speech and Speech-to-Braille Using Raspberry Pi," in Proc. 2024 *IEEE International Conference on Computing* (ICOCO), 2024.

[5] N. Haga and S. Ito, "Wearable Braille Reader," in Proc. 2021 *IEEE 10th Global Conference on Consumer Electronics (GCCE)*, Kyoto, Japan, Oct. 2021, pp. 849–852.

[6] S. R. Patil, R. S. Kulkarni, and P. N. Patil, "Assistive Technology for Braille Reading using Optical Braille Recognition and Text-to-Speech," in Proc. *IEEE Int. Conf. Emerging Smart Computing and Informatics* (ESCI), 2023, pp. 1–6.

[7] R. P. Singh, A. S. Rana, M. Mehra, D. Chhabra, and A. Dubey, "Text to Speech Implementation of E-Braille Bench with Smartphone APP," *International Journal of Engineering Research and Reviews*, vol. 4, no. 2, pp. 107–116, Apr.–Jun. 2016.

[8] https://aeldata.com/brief-introduction-to-braille/

[9] https://www.espressif