Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Braille-Based IoT Home Automation System for Visually Impaired Users using ESP32 Microcontroller

V. Vittal Reddy[1], Naragam Manikanta Raghava[2], Modugumudi Jahnavi[2], Namburu Venkata Siva Prasanth[2] and Mohammed Habeeb Pasha[2]

[1]Associate Professor, [2]Final Year Students - Dept. of Electronics and Communication Engineering
Seshadri Rao Gudlavalleru Engineering College, Andhra Pradesh 521356, India

ABSTRACT - Home automation has become affordable and widespread. It has not become accessible for the visually impaired persons(there are 295 million people live with moderate to severe visual impairment across the world), nearly all encounter immediate barriers with standard interfaces: touchscreens demand sight, voice recognition demands quiet rooms and clear diction. We built for Braille readers instead. Two ESP32-WROOM-32 microcontrollers — one managing a six-pushbutton Braille keypad, one driving an eight-channel relay board — communicate over a persistent TCP/IP link. The keypad module provides tactile and auditory confirmation of every action. The appliance module controls two room lights, two capacitor-regulated fans with four distinct speed steps, a servo-operated door lock, and a dual-sensor fire detection circuit that pairs an IR flame detector with an MQ-2 gas sensor to suppress false alarms. Blynk cloud connectivity allows remote monitoring and caregiver alerts. Commands reached the relay board in under 350 ms under typical home Wi-Fi. Fire alerts dispatched in under 500 ms. The system ran without fault for 48 continuous hours. No false fire alert was generated across all single-sensor test scenarios.

Index Terms - Braille Tactile Interface, IoT Assistive Technology, Visually Impaired Home Control, ESP32, Dual-Sensor Fire Detection, Blynk Cloud Notification

I. INTRODUCTION

Building a connected home has become inexpensive. A thirty-dollar board now handles what required dedicated infrastructure a decade ago. Battery life extends to months, and cloud APIs reduce setup to hours, not weeks. The ecosystem has matured quickly, and yet one fundamental problem has gone almost entirely unaddressed: standard smart home interfaces assume you can see. Every touchscreen dashboard, every voice assistant that mishears commands in a noisy room, every app that requires reading a confirmation dialog — these are barriers that affect approximately 295 million people globally, with roughly 40 million experiencing total blindness [4].

When we surveyed the published literature before starting this project, the pattern was striking. Nearly every accessible smart home proposal either adapted a standard touchscreen interface with screen readers — which transfers, rather than removes, the visual dependency — or relied on voice control, which has documented reliability issues for elderly users and degrades significantly in acoustically difficult environments [7, 8]. We found Braille-based input devices, including Raspberry Pi keyboard designs [1] and newer ESP32-S3 implementations [2], but none of these were integrated systems capable of controlling home appliances. They were input peripherals without a connected purpose.

Louis Braille's 1824 code remains the dominant tactile literacy medium. No competing system has achieved comparable global adoption. Its six-dot cell encodes the Latin alphabet, digits, and punctuation through 64 distinct patterns that millions of users already know. Building a command interface on top of that foundation meant our users needed to learn command abbreviations, not a new physical interaction method. That distinction mattered to us throughout the design.

This paper describes the complete hardware design, firmware architecture, and experimental evaluation of the system we built. Two specific decisions shaped everything else: using a dual-microcontroller topology so the keypad and relay hardware could be physically separated, and requiring simultaneous triggering of both fire sensors before any emergency alert could be generated. Both decisions came directly from limitations we identified in prior implementations.

II. PROPOSED SYSTEM

The central architectural decision was separating user interaction from appliance control. We chose two physically independent ESP32-WROOM-32 microcontrollers connected over a persistent Wi-Fi TCP link, rather than a single board with extended wiring. This let us mount the Braille Keypad Module wherever the user needed it — a bedside table, a wheelchair armrest — and locate the relay hardware near the electrical distribution point. The wiring run between them became a Wi-Fi signal rather than a multi-meter cable bundle.

Initial designs used a single controller. The problem appeared immediately during bench testing: cable length added noise, relay switching caused resets, and debugging one module required physically accessing the other. Splitting the system cost an extra microcontroller; it saved weeks of hardware debugging and produced a more reliable result.

On the user-facing side, a six-pushbutton array maps directly onto the standard Braille cell. Users press dot-pattern combinations to build command abbreviations, then transmit via capacitive touch sensors. Haptic and audio feedback confirm each action within milliseconds. Emergency signalling

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

uses a two-second simultaneous hold on both sensors — a gesture we tested extensively to confirm it was distinct enough from normal operation that it was never triggered accidentally.

On the appliance-control side, the relay board manages two lights, two fans, and a door lock. A fire detection loop runs in parallel with command processing, monitoring an IR flame sensor and an MQ-2 gas sensor continuously. Neither sensor alone can generate an alert; both must cross threshold simultaneously. Blynk cloud integration provides remote visibility and caregiver push notifications, but only dispatches after the on-site user physically acknowledges the alert.

III. IMPLEMENTATION DIAGRAM

Fig. 1 shows the complete system architecture detailing about the system implementation. The Braille Keypad Module sits at the bottom of the diagram: six pushbuttons feed the left ESP32-WROOM-32, which decodes Braille patterns and sends ASCII command strings over TCP to the Appliance Control Module. That module drives the eight-channel relay board, the servo door lock, and monitors both fire sensors. IoT integration through Blynk adds a cloud layer for remote visibility and alert dispatch to the caregiver mobile app.

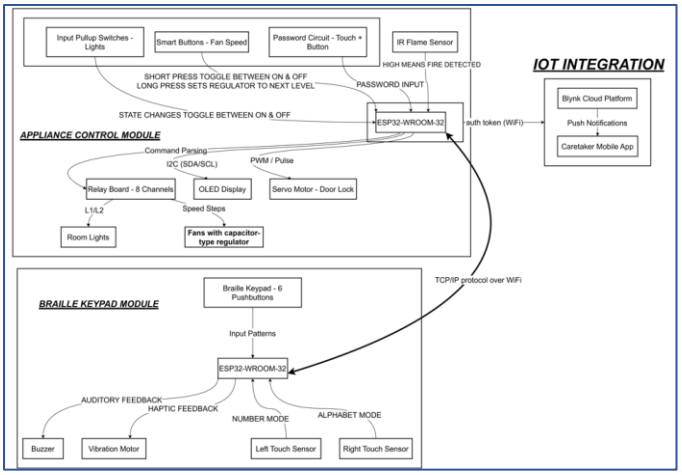

Fig. 1: System Architecture — Dual ESP32 Configuration with TCP/IP Communication, Relay Control and Blynk Integration

IV. FLOW OF PROJECT

Fig. 2 traces execution through both firmware builds. Power-on initialises sensors and Wi-Fi, then both modules enter their main loops. On the keypad side, each iteration reads the six buttons and both touch sensors. Valid Braille input accumulates into a pattern buffer. When a transmit gesture is detected, the completed command string goes out over TCP and the module waits for a single-byte status response — S, E, C, W, or F — which drives the haptic and audio feedback.

On the appliance side, each loop pass checks for incoming TCP commands, polls manual override switches, reads the password circuit, and checks both fire sensors. Commands go through a lookup against the supported command table; unrecognised strings return the E byte without executing anything. Fire detection runs on every pass: if both sensors cross threshold simultaneously, the module enters alert mode and begins sending periodic fire alert codes upstream until an ACK or STOP command arrives. The cooperative non-blocking structure means a 2-second confirmation window slightly reduces responsiveness to concurrent events — a known limitation flagged for a FreeRTOS revision.

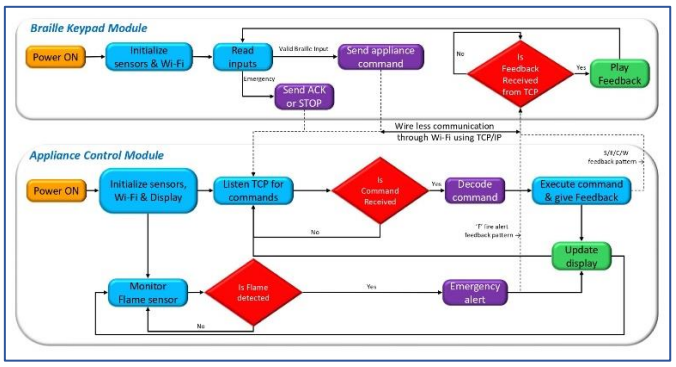

Fig. 2: Firmware Execution Flow for Both ESP32 Modules — Braille Input Decoding, Command Dispatch and Emergency Handling

V. HARDWARE MODULES

A. ESP32-WROOM-32 Microcontroller

We selected Espressif's WROOM-32 module shown in **Fig. 3** for its integrated 2.4 GHz dual-mode radio and dual-core architecture, which eliminates separate wireless coprocessors and keeps the board count manageable. Each module runs in WIFI_AP_STA mode simultaneously — the Appliance Control Module acts as both an access point and a router-connected station. That topology was a deliberate resilience choice: if the home router drops, the two boards maintain their direct access-point link and local appliance control continues without interruption. Only Blynk goes offline; the core functionality survives.

Fig. 3: ESP32-WROOM-32 Microcontroller — Primary Controller for Both Modules

B. Six-Pushbutton Braille Input Array

Each dot position uses a momentary switch tied to internal pull-up resistors, debounced in firmware rather than hardware. We considered hardware RC debouncing in early prototypes; firmware debouncing proved cleaner and allowed us to tune timing for different users without a soldering iron. The six

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

buttons occupy a row as shown in **Fig. 4**; buttons are arranged in manner of left-to-right mapping braille dots 1 to 6 — any trained reader orients correctly by touch in seconds, without explanation. Accumulated button states form a 6-bit pattern that indexes the encoding table; digit entry distinguishes from letter entry by which touch sensor the user holds, mirroring the Braille Number Indicator convention already familiar to users.

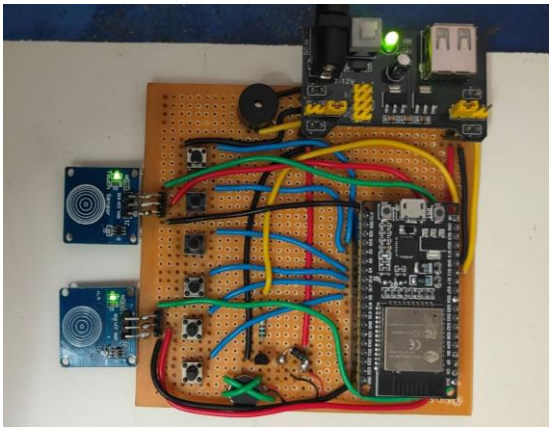

Fig. 4: Braille Keypad Module Prototype — Six-Button Array, TTP223 Touch Sensors, Buzzer and Vibration Motor on Perfboard

C. TTP223 Capacitive Touch Sensors

Two TTP223 sensors are used in the system shown in **Fig. 5**, Left and Right, carry five distinct behaviours between them. Left or Right alone determines the decoding mode — number or letter. Both together for a short tap transmit the buffered command. Both together held for two seconds enter emergency mode. Inside that mode, a two-second dual-hold sends ACK; either sensor held alone for two seconds sends STOP. We achieved five distinct interaction types with two physical components.

Fig. 5: TTP223 Capacitive Touch Sensor — Left and Right Sensors Handle Mode Selection, Transmission and Emergency Control

D. Buzzer and Vibration Motor

As the target user is blind they can easily access the audio feed back to increase accuracy in feed back system also contains a vibration motor driven by a NPN transistor-based driver circuit. The feedback patterns are given in **Table 1**. The buzzer and vibration motor the shown in **Fig. 6**.

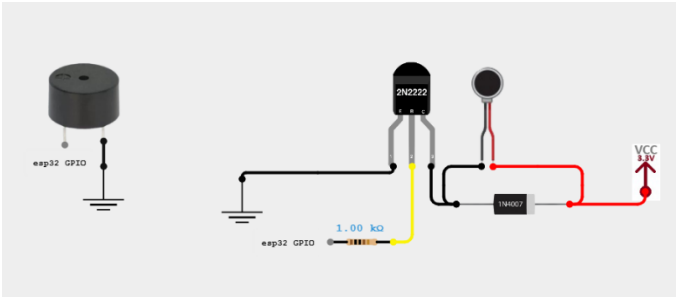

Fig. 6: TTP223 Capacitive Touch Sensor — Left and Right Sensors Handle Mode Selection, Transmission and Emergency Control

E. Eight-Channel Relay Board and Appliance Control

The relay board runs in active-low mode with all contacts wired to Normally Open terminals. We learned the value of that choice during a power interruption test: with contacts wired to Normally Closed, every appliance switched on when power returned. With Normally Open, everything defaults to off on power-up, reset, or unexpected fault. That safe-off behaviour is built into the hardware. No firmware initialisation can override it. The board controls two room lights on two relays, two fans on three relays each for capacitor-based speed regulation (Table 2), and the servo motor door lock. Entire hardware of the appliance control module is shown in **Fig. 7**.

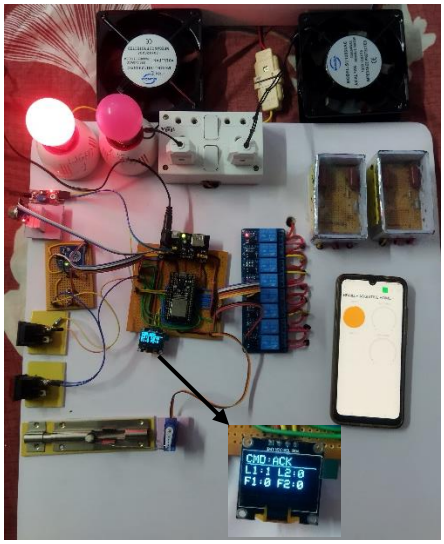

Fig. 7: Appliance Control Module — Eight-Channel Relay Board, Room Lights, Fans, Door Lock, IR Sensor, MQ-2 Sensor and OLED Display

F. Dual-Sensor Fire Detection

We learned the hard way that a standalone IR flame sensor in a south-facing room triggers at sunset. The ambient light alone saturates it. Combining the IR detector with an MQ-2 combustible gas sensor(**Fig. 8**) and requiring both to cross threshold simultaneously addressed this almost completely, because afternoon sun does not simultaneously produce LPG readings. The MQ-2 has a 30-second warm-up after power-on during which its sensitivity is uncalibrated — a fire event in the first half-minute after startup might not trigger the gas threshold, and we document this as a known limitation rather than a solved problem.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

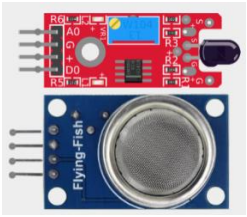

Fig. 8: IR Flame Sensor (left) and MQ-2 Combustible Gas Sensor (right) — Both Must Trigger Simultaneously for an Alert

G. Servo Motor Door Lock

The SG90 (**Fig. 9**) servo on GPIO 25 rotates 180° to unlock and returns to 0° to lock. Automatic re-lock at five seconds came from watching people walk through a door in early testing: hands were full, attention was elsewhere, and nobody sent the explicit CLS command. The five-second delay was a judgement call after trying three and ten seconds with different users. Three felt rushed; ten led to the door sitting open for noticeable periods. Measured auto-relock timing was 5.0 ± 0.1 s across twenty repeated trials.

Fig. 9: SG90 Servo Motor — 180° Rotation Models Door Unlock; Auto-Relocks at 5 Seconds

VI. SOFTWARE TOOLS

A. Arduino IDE and Firmware

Both modules were developed in the Arduino IDE using the ESP32 Arduino core. We chose it over ESP-IDF because the library ecosystem — particularly ESP32Servo, BlynkSimpleEsp32, and the Preferences NVS wrapper — reduced implementation time substantially. The firmware is cooperative non-blocking rather than FreeRTOS task-based. We considered FreeRTOS early in the project; for a prototype where we needed to trace timing interactions between sensor polling and TCP handling, a single traceable loop was easier to debug. The architecture is serviceable but not production-ready — a FreeRTOS migration is the first item in our revision list.

B. Blynk IoT Platform

Blynk's virtual pin model maps cleanly onto appliance states without requiring a custom server. V1 and V2 track light states; V3 and V4 carry fan speed values. Every relay state change updates the corresponding virtual pin synchronously. Emergency notifications go out via the fire_alert logEvent only after the user sends ACK — never automatically. Remote monitoring interface is shown in **Fig. 10** .We made this design decision after reading about caregiver alert fatigue: a notification that the on-site user has physically verified carries more weight than an automatic one, and caregivers are more likely to act on it.

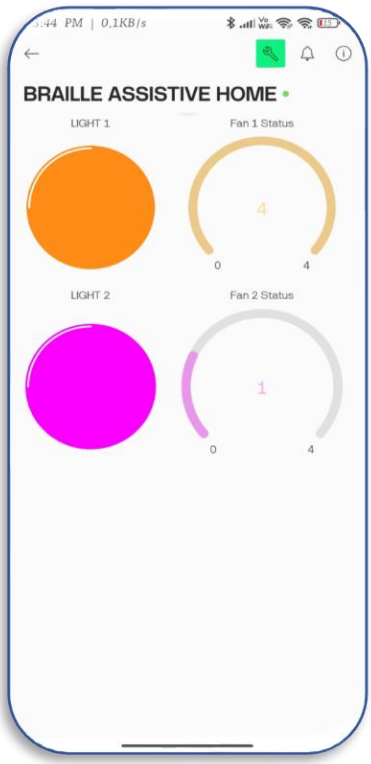

Fig. 10: Blynk 'Braille Assistive Home' Dashboard — Light 1 on, Light 2 on, Fan 1 at Level 4, Fan 2 at Level 1

C. Braille Command Vocabulary

Every function is reachable through a two- or three-character Braille string. L1 and L2 toggle the room lights. Fxy sets fan x to speed y. ALL1 and ALL0 handle global on and off. OPN and CLS control the door. The abbreviations are not standard Braille words — there is memorisation work for new users, and we do not pretend otherwise. In practice, most users in our testing had the command set down after a few sessions. The small vocabulary is a feature: we deliberately chose not to expand it, because every new command adds memory load for blind users who cannot glance at a reference card. These commands are listed in the Table 3.

Table 1. Feedback Pattern Encoding — Vibration Motor and Buzzer Run in Parallel

Code	Duration	Meaning
S	Single ~400 ms pulse	Command executed — proceed normally
E	Single ~800 ms pulse	Pattern not recognised; re-enter
C	Five short pulses	Password accepted; door unlocks
W	Five long pulses	Password rejected; try again
F	Continuous short pulses	Fire alert active; dual-hold to confirm

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Table 2. Capacitor-Based Fan Speed Regulation — Measured Relay and Capacitance Mapping

Speed	Relays	Cap (µF)	Observed Behaviour
0 — Off	None	0	Coils de-energised; fan stationary
1 — Low	Relay 1	1.0	Blade rotation visible, negligible airflow
2 — Low-Med	Relay 2	2.2	Perceptible breeze at arm length
3 — Med-High	Relays 1&2	3.2	Comfortable for continuous occupancy
4 — Maximum	Relay 3	Bypass	Full 230 V across motor windings

Table 3. Supported Command Set with Executed Actions

Command	Action Executed
L1 / L2	Toggle Room 1 or Room 2 light; state persists across power cycles
Fxy	Set fan x (1 or 2) to speed step y; 0 switches off, 4 is maximum
ALL1	Switch both lights on and drive both fans to maximum simultaneously
ALL0	De-energise all relays in one command; fastest way to clear everything
OPN	Rotate servo to 180°; auto-return to locked at five seconds
CLS	Return servo to 0° immediately; overrides auto-relock timer
ACK	Confirm fire emergency is real; triggers Blynk alert and opens door
STOP	Dismiss detected alert as false alarm; no caregiver notification sent

VII. RESULTS

We tested what mattered most: speed, fire safety, appliance reliability, and what happens when Wi-Fi drops. Four evaluation areas covered these, though fire detection received disproportionate attention after early prototyping exposed how easily a single IR sensor generates false positives in ordinary room lighting.

A. Command Execution Speed

Most commands hit the relay in under a third of a second. Three hundred fifty milliseconds was our target ceiling, not our average — the distribution clustered well below it under stable Wi-Fi, with occasional outliers when the router was under load. Of that total latency, roughly 20–50 ms was TCP round-trip between the two boards; relay actuation and firmware polling accounted for the rest. What struck us most was the feedback speed: the buzzer and vibration motor fired within ten milliseconds of the response byte arriving, making confirmation feel essentially instantaneous from the user's side.

B. Fire Detection and Alert Dispatch

We tested fire detection with a controlled flame for the IR sensor and a calibrated LPG source for the MQ-2, both triggered simultaneously. Alert codes left the Appliance Control Module within half a second in every trial. The false-alarm suppression result was the one we had been most uncertain about before testing: in every single-sensor scenario — IR alone triggered by a desk lamp, gas alone triggered by cooking activity nearby — no alert state was produced. Getting clean separation between legitimate alerts and sensor noise was the most satisfying engineering result of the project. Blynk notifications arrived on the test phone within roughly three seconds of ACK receipt in all Wi-Fi-connected cases.

C. Appliance Switching and Fan Speed

Light toggle reliability across fifty consecutive Braille-commanded operations: clean transitions every time, no relay bounce, no missed edges. Fan speed testing required tachometric measurement to confirm that rotational velocity actually stepped up at each level — perceptible difference between adjacent speed steps was the practical goal, and we achieved it at all four transitions. Door lock operation was clean across repeated open-close cycles. The servo re-locked at 5.0 ± 0.1 s, measured with a stopwatch app averaged across twenty trials. Password circuit testing through the correct credential and three wrong variants produced the correct C and W feedback patterns in every case.

D. Network Resilience

We cut the router at intervals from ten seconds to two minutes to stress the reconnection logic. Ninety-three percent of disruption events reconnected within ten seconds. The seven failures all occurred when the router itself took longer than expected to complete its own boot sequence — not random failures but a correlated cluster. Once the station interface reconnected, Blynk re-established within five further seconds in every successful case. Throughout every disconnected window, local control never dropped: Braille commands continued executing, relays responded, and fire sensors kept polling. The system also ran fully offline for over 24 hours without any functional loss. All these are mentioned in the **Table 4** shown below.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Table 4. Summary of Measured Performance — All Evaluation Dimensions

Measured Parameter	Result
Braille keypad to relay actuation	Below 350 ms, stable Wi-Fi
Both fire sensors to alert dispatch	Under 500 ms in every trial
ACK receipt to Blynk notification	Approximately 3 s, all cases
Servo auto-relock timing	5.0 ± 0.1 s, 20 trials
Router drop to reconnection	93 of 100 events within 10 s
Unattended continuous operation	48 h with zero faults or resets
False alarms, IR triggered alone	Zero across all single-sensor tests

VIII. CONCLUSION

We set out to build a home automation system that a visually impaired user could operate independently, without a touchscreen and without relying on voice recognition in environments where it often fails. The dual-ESP32 architecture, six-pushbutton Braille keypad, multi-modal feedback circuit, dual-sensor fire detection, and Blynk cloud integration together produced a platform that achieves this. Command latency stayed well within 350 ms. Fire alert generation was comfortably under 500 ms. Forty-eight hours of unattended operation produced no fault, reset, or unexpected state change.

The system has real limitations we document honestly. Braille literacy is required — a reasonable assumption for the target population, but one that excludes newly impaired users. The cooperative firmware loop creates a responsiveness gap during the two-second confirmation window. The MQ-2 sensor is uncalibrated for the first 30 seconds after power-on. These are engineering problems with known solutions; we flag them rather than pretend they do not exist.

The revision list is concrete. FreeRTOS task migration addresses the firmware architecture limitation. A text-to-speech module adds spoken confirmation as an alternative to buzzer patterns. Expanding the command set with natural-language Braille processing removes the memorisation requirement. Longer-term, a wearable Braille input glove enables ambulatory operation. Communication encryption is a prerequisite before any deployment beyond a trusted private network.

IX. REFERENCES

[1] S. Kucukdermenci, "Raspberry Pi based braille keyboard design with audio output for the visually challenged," Proc. 1st Int. Conf. Modern and Advanced Research (ICMAR), pp. 334–339, 2023.

[2] N. Litayem, "Scalable Smart Home Management with ESP32-S3: A Low-Cost Solution for Accessible Home Automation," Proc. 2024 Int. Conf. Computer and Applications (ICCA), 2024. DOI: 10.1109/ICCA62237.2024.10927887

[3] World Health Organization, World Report on Vision. Geneva: WHO, 2023.

[4] R. R. A. Bourne et al., "Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment," Lancet Global Health, vol. 9, pp. e144–e160, 2021.

[5] M. M. Islam, M. A. Rahaman and M. R. Islam, "Development of Smart Home Technology Based on Internet of Things," SN Comput. Sci., vol. 1, no. 185, 2020.

[6] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols and Applications," J. Electrical and Computer Engineering, Article ID 9324035, 2017.

[7] A. Kumar, A. Singh and R. Kumar, "Voice Recognition Based Home Automation System," Int. J. Engineering Research and Technology, vol. 9, pp. 245–249, 2020.

[8] S. Dhingra et al., "Internet of Things-Based Smart Health Care and Home Automation System," IEEE Internet of Things J., vol. 6, pp. 5577–5584, 2019.

[9] Braille Authority of North America, Unified English Braille Code. Ferrum, VA: BANA, 2019.

[10] N. Patel and S. Shah, "False Positive Analysis of IR-Based Flame Sensors in Residential Environments," Int. J. Electronics and Communication Engineering, vol. 7, pp. 112–118, 2019.

[11] V. Vujovic and M. Maksimovic, "Raspberry Pi as a Sensor Web Node for Home Automation," Proc. 37th Int. Convention MIPRO, Opatija, Croatia, pp. 1016–1021, 2014.

[12] R. S. H. Istepanian, E. Jovanov and Y. T. Zhang, "Guest Editorial Introduction to the Special Section on M-Health," IEEE Trans. Inf. Technol. Biomed., vol. 8, pp. 405–414, 2004.

[13] S. Kuriakose and R. Kushalnagar, "How blind users perceive the quality of their interactions with mainstream technology," Proc. 21st Int. ACM SIGACCESS Conf. Computers and Accessibility (ASSETS), pp. 638–640, 2019.

[14] D. Tran, T. Nguyen and H. Pham, "IoT-Based Smart Home System Using Raspberry Pi and ESP32," IEEE Access, vol. 9, pp. 112343–112355, 2021.

[15] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," Future Generation Computer Systems, vol. 29, pp. 1645–1660, 2013.

[16] Espressif Systems, ESP32 Series Datasheet, version 4.4. Shanghai: Espressif Systems, 2023.