



BrailleLearn: A Low-Cost IoT-Based Braille Learning System with Dot-Level Error Analysis and Targeted Practice

Hari Shankar M¹, Prem Kumar H², and Dharmendran V³

^{1,2,3}B.Tech Computer Science and Engineering (Artificial Intelligence)

Vels Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, India

Project guide: Dr. A. Manikandan

Abstract- Learning Braille depends on repeated tactile practice, yet beginners often practise with printed material or with a teacher present to say whether a pattern is right. This paper describes the design of BrailleLearn, a low-cost console in which six push buttons, arranged as a Braille cell, and a confirm button are read by an ESP32 microcontroller. After each attempt the learner's six-bit pattern is compared with the target pattern, and the difference is resolved to individual dots, separating dots that were left out from dots that were pressed without being required. The accumulated per-dot error counts are intended to drive a targeted practice mode that favours characters involving the dots a learner finds hardest. Feedback is planned through LEDs and a buzzer, with an optional Bluetooth Low Energy link to a smartphone for spoken prompts and progress records. We review related work on refreshable displays, IoT assistive devices, and technology for Braille literacy, state the gap the console addresses, and present the hardware, software, and evaluation design. At the time of writing the physical prototype and firmware have not been completed; no measurements or user-study results are reported, and an evaluation plan is given in their place.

Keywords- Braille learning, assistive technology, ESP32, Internet of Things, tactile input, error analysis, Bluetooth Low Energy, educational device.

I. INTRODUCTION

Braille writes each character as a pattern of raised dots inside a cell of six positions, arranged in three rows and two columns. A beginner has to learn two things together: how to recognise the patterns by touch, and how to produce them, which means knowing which of the six positions belong to a given character. Both skills need repetition, and repetition needs someone, or something, to say whether an attempt was right. A systematic review of technology for Braille literacy education in children covered twelve studies with 176 participants in total and judged the quality of the evidence to be generally low [1], so firm guidance on which kinds of devices actually help is still thin.

Hardware research on the topic has leaned towards the output side, that is, towards refreshable displays that raise and lower pins under program control [2], [3], [4], [5]. Such devices are useful, but the prototypes also show what they cost to build. A single-character display driven by solenoids from an ESP32 ran into heat and power problems and had to be reduced in size [2], and the other low-cost designs we reviewed still involve actuator trade-offs or mechanical complexity [3], [4]. For a student team, an actuated display is a demanding first step.



BrailleLearn starts from the other end. The learner forms a pattern on six push buttons laid out like a Braille cell and presses a confirm button, and an ESP32 microcontroller checks the answer. Because the answer is already a six-bit value, the controller can say more than “right” or “wrong”. Comparing target and response bit by bit shows which dots were left out and which were pressed unnecessarily. A running count of such errors, kept for each dot, indicates which dots a particular learner finds difficult, and the next exercises can be chosen with those dots in mind. This idea is the technical centre of the paper.

Status of the Work- This is a design-stage report. As of the first project review, the literature study and component selection were the work in progress, and the physical prototype had not been started. The hardware, firmware, and practice-selection scheme in Sections V–XI are therefore described as designed, proposed, or planned, and Section XIII contains no measurements. We have preferred to say this plainly rather than present untested behaviour as a result.

The contributions we claim are limited to the following:

- a low-cost console design built around an ESP32, six tactile dot buttons, and a confirm button, with LED and buzzer feedback and an optional BLE link to a smartphone;
- a dot-level comparison of target and response that separates omitted dots from extra dots (Section X);
- a proposed rule that turns per-dot error counts into a choice of the next practice character (Section X);
- a modular architecture in which the learner-facing device and the supervisor-facing display can be developed independently; and
- an evaluation plan with concrete test cases and measurements, to be carried out once the prototype exists (Section XII).
- The rest of the paper reviews related work (Sections II–III), states the problem (IV), describes the proposed system, its architecture, hardware, and software (V–IX), explains the error analysis (X), reports implementation status and the evaluation plan (XI–XIII), and closes with limitations, future work, and conclusions.

II. BACKGROUND AND RELATED WORK

1. The Braille Cell as a Six-Bit Pattern

Every position in the cell is either raised or flat, so a cell can be stored as a six-bit number. We number the positions 1–3 down the left column and 4–6 down the right, which is the usual convention, and our lookup table (Section VIII) follows the standard patterns for the letters A to J. Treating the cell as a bit pattern is what makes the dot-level analysis in Section X cheap: the comparison of two cells reduces to bitwise operations rather than any form of recognition.

2. Output-Oriented Devices

Much of the reviewed work concerns displays that present Braille to the reader. Ramos-García et al. built a single-character display with an ESP32 and electromagnetic solenoids and linked it over MQTT [2]. Ghosh et al. used micro-servos with a slider mechanism for a single cell [4], Walker and Sangelkar examined affordable actuation for low-income users [3], and Saikot and Sanim adjusted the cell size to suit learners with different tactile sensitivity and added speech output [6]. Vibration has also been used as a tactile channel, in a display-and-keyboard combination [7]. Tactile perception depends on geometry: studies with pin-array tablets and children point to dot size and spacing as important for recognition [8]. These observations matter to us even though our first prototype has no raised-dot output, because button size and spacing will affect how reliably a child can find the right key.



3. Connected and Learning-Oriented Systems

The ESP32 is widely used in IoT prototypes because it combines a microcontroller with Wi-Fi and Bluetooth radios [9]. Connected assistive systems built on similar hardware include a speech-to-Braille printing system [10], an OCR-to-Braille system over a wireless sensor network [11], and general-purpose IoT aids for visually impaired users, such as navigation assistance [12], [13]. Closer to our goal, Latif et al. describe an IoT and AI-based Braille learning system that uses a camera and a transfer-learning CNN [14]. Didactic resources for visually impaired learners have also been surveyed more broadly [15]. We discuss how these relate to our design in the next section.

III. LITERATURE SURVEY

Summary of Surveyed Literature

Table I lists the papers we examined, with the aspects that bear on this project. The entries record what each paper reports about its own system; where a paper does not state something relevant to us, we do not infer it. Table II then contrasts the main families of approach with the proposed console.

TABLE I

Ref.	Authors (year)	Methodology	Main contribution	Limitation	Relevance to this project
[1]	Sutherland <i>et al.</i> (2022)	Systematic review; 12 studies, 176 participants	Synthesis of evidence on technology for children's Braille literacy education	Evidence quality generally low; evaluation standards needed	Motivates cautious claims and a properly consented evaluation
[2]	Ramos-García <i>et al.</i> (2022)	ESP32 with electromagnetic solenoids; MQTT; SCRUM-style iterative development; edge-oriented architecture	Affordable single-character refreshable Braille display with IoT communication; sequential and random character output checked for pin correctness	Solenoid heat and power problems; prototype size needed reduction; usability and effectiveness testing left as future work	Base paper. Same controller family and IoT idea, but we address practice input and feedback rather than raised-dot output
[3]	Walker and Sangelkar (2017)	Design exploration of low-cost actuators for refreshable displays	Cost-oriented options for low-income users	Refreshable displays remain costly; actuator trade-offs	Supports leaving actuated output out of the first prototype
[4]	Ghosh <i>et al.</i> (2023)	Micro-servos with a slider mechanism; single-cell display	Low-cost refreshable cell (Brindle)	Mechanical complexity; single-cell constraints	Shows a mechanical route we defer to future work
[5]	Liu <i>et al.</i> (2024)	Low-cost actuator display combined with an active touch-reading and audio system	Refreshable display for digital reading with touch and audio	Mechanical display and test set-up remain complex	Precedent for combining touch with audio, as in our planned app
[6]	Saikot and Sanim (2022)	Single-cell refreshable display with adjustable cell size and speech	Adapts to different tactile sensitivity; supports independent practice	More mechanical complexity than a beginner input console	Tactile dimensions as a learner-specific design variable
[7]	Aqel <i>et al.</i> (2019)	Vibro-tactile Braille display and keyboard	Vibration as a supplement to tactile communication	Hardware is more specialised	Basis for our optional vibration cue



Ref.	Authors (year)	Methodology	Main contribution	Limitation	Relevance to this project
[8]	Masclé <i>et al.</i> (2022)	Tactile pictograms on pin-array tablets, tested with blind and sighted children	Evidence that dot size and spacing affect tactile recognition	Needs specialised pin-array hardware	Guides future tactile output and button geometry
[10]	Ahmed <i>et al.</i> (2020)	IoT system converting speech to printed Braille	Low-cost connected Braille generation	Conversion and not printing, learning	Feasibility of Braille processing on low-cost connected hardware
[11]	Authors not yet recorded (2024)	OCR, IoT, and refreshable Braille over a wireless sensor network	Accessible conversion of printed text for visually impaired and deaf-blind users	Focus is information conversion, not beginner learning	Shows IoT use in Braille systems; different goal
[12]	Talukdar <i>et al.</i> (2020)	IoT-based smart assistive device	Connected embedded device for visually impaired users	Assistance, not Braille pedagogy	Role of low-cost networked embedded systems
[13]	Patil <i>et al.</i> (2023)	Low-cost IoT navigation aid	Navigation assistance for visually impaired users	Navigation, not Braille pedagogy	Same as above
[14]	Latif <i>et al.</i> (2023)	Camera input with a transfer-learning CNN; IoT	IoT and AI-based Braille learning system	Camera and ML pipeline add complexity	Closest in aim; we use direct button input and rule-based analysis instead
[16]	Udana <i>et al.</i> (2024)	Electromechanical device turning digital text into Braille	Tactile Braille communication for inclusive education	Primarily communication and text conversion	Educational setting similar; function differs

TABLE II Families of Existing Approaches Compared with the Proposed Console

Approach	Interaction / mechanism	Difference in BrailleLearn
Single-cell refreshable displays [2], [4], [6]	Learner reads dots produced by solenoids, servos, or similar actuators	Input side only; no actuated dots, so no actuator heat or mechanics, but also no reading practice
Cost and reading-system studies [3], [5]	Low-cost actuation; touch with audio	No dynamic tactile output in the first prototype
Vibro-tactile devices [7]	Vibration plus keyboard	Vibration only as an optional cue
Camera and CNN learning system [14]	Camera recognises Braille; IoT and AI	Direct button input; analysis is rule-based counting, not a trained model
Conversion and printing systems [10], [11], [16]	Text, speech, or print converted to Braille	Aim is practice and feedback, not conversion
BrailleLearn (proposed)	Six dot buttons and a confirm button; ESP32	Dot-level error analysis and targeted practice; LED, buzzer, optional BLE



Synthesis. Three themes emerge from the table. First, most of the hardware work is output-centred. Refreshable and pin-array displays give the reader something to feel [2], [4], [8], but the authors who report on practicalities mention heat, power, size, or mechanical complexity [2], [4], [5]. That is a reasonable price for a reading device, though it is not obviously the right starting point for a low-cost drill tool. Second, the connected systems built on low-cost boards are mostly about conversion or general assistance [10], [11], [12], [13]; they show that the platform is workable, not that it has been used for structured practice. Third, the learning-focused work either evaluates technology at a general level [1] or relies on a camera and machine-learning pipeline [14], and the review of Sutherland et al. reminds us that rigorous evaluation of such tools is still scarce [1].

Gap addressed. In the papers we reviewed, we did not find a low-cost microcontroller design that combines (i) six-button Braille pattern input, (ii) feedback resolved to individual dots, and (iii) practice selection driven by the learner's dot-level history. This is a statement about the set of papers we examined. We have not carried out a systematic search, so we do not claim that no such system exists elsewhere.

IV. PROBLEM STATEMENT

Beginners practising Braille need frequent feedback, but a teacher is not always available and printed drill sheets cannot respond. Existing hardware that does respond is often costly or mechanically involved to reproduce (Section III). We therefore set the problem as follows: design an inexpensive, reproducible device on which a beginner can form Braille patterns, receive immediate feedback that says which dots were wrong, and be steered towards the patterns that need more work. The objectives are:

- provide a six-dot tactile input with a separate confirm control;
- give immediate, unambiguous feedback for correct and incorrect answers;
- identify omitted and extra dots separately for every attempt;
- keep per-dot error counts and use them to choose follow-up exercises;
- let a supervisor see the target and the response, first over a wired or display link and later over BLE; and
- keep the design modular and low-cost enough for student-level prototyping.
- Learning effectiveness is not an objective of this stage; it can only be assessed by the evaluation described in Section XII.

V. PROPOSED SYSTEM

Feature	Category	Status
Six dot buttons and confirm button	Own design	Planned (hardware phase)
Six-bit lookup table, letters A–J	Own design, standard patterns	Designed; not in firmware
Dot-level comparison (omitted / extra)	Own design	Designed; not implemented
Per-dot error counters	Own design	Designed; not implemented
Targeted practice selection	Own design	Proposed rule; parameters unset; unvalidated
LED and buzzer feedback	Own design	Planned
Vibration motor cue	Optional, after [7]	Undecided
Serial or OLED supervisor view	Own design	Planned for first prototype
BLE link to smartphone	Own design	Planned proof of concept
Smartphone app with speech and lessons	Own design	Future work



Feature	Category	Status
Wi-Fi / MQTT logging	After [2]	Future work
Raised-dot or actuated output cell	Existing work [2], [4]	Out of scope; future work
Usability study with learners	Own plan	Not started; needs consent

The console consists of an ESP32 controller, six push buttons for dots 1–6, a confirm button, LED and buzzer outputs, and an optional supervisor interface. For each exercise the controller picks a target character, waits for the learner to press a dot combination and confirm it, compares the response with the target, signals the result, and updates its per-dot error record. Four learning modes are planned: letter identification, word spelling, number learning, and free practice (Section IX).

To keep the four categories apart, existing work is what the cited papers report (Section III); our contribution is the combination listed in Section I; implemented would mean built and run on hardware; and planned or future work covers everything else. By that definition nothing in Table III is yet implemented. The ideas in the first rows have been worked out on paper, but no firmware has been written or tested against a real device.

We make no claim that button-based Braille practice devices are new. The contribution lies in a specific combination: a low-cost ESP32 build whose feedback is resolved to individual dots and whose exercise selection is driven by those dot-level records.

VI. SYSTEM ARCHITECTURE

Fig. 1 shows the module structure. Five modules are involved, and only the first three are needed for the first prototype.

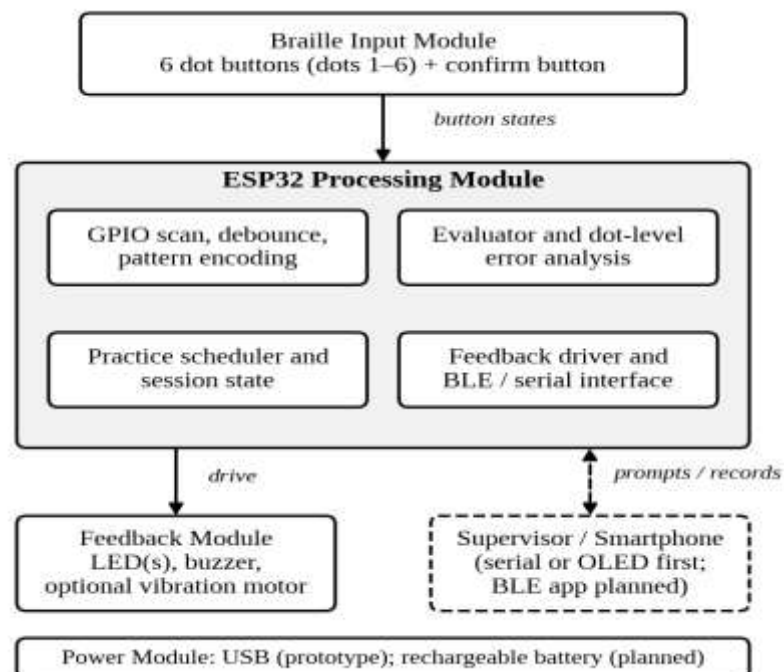


Fig. 1. Module architecture of BrailleLearn. Dashed outlines mark planned extensions.

Braille input module.

Six buttons mapped to dots 1–6 and a confirm button. The controller reads their states and treats them as one six-bit value once the confirm button is pressed.



ESP32 processing module. It holds the whole logic: scanning and debouncing, encoding, the evaluator and dot-level analysis, the practice scheduler, and the interface to the feedback hardware and to serial or BLE.

Feedback module.

LEDs and a buzzer signal the result. A small vibration motor is optional.

Supervisor / smartphone module. The learner mainly uses the buttons and hears the feedback, while the supervisor needs to see the target. The first prototype is meant to use a laptop on USB serial or a small OLED. A BLE link to a smartphone application, which could speak prompts and show progress, is planned but not built.

Power module

. USB during prototyping; a rechargeable battery is a later option.

VII. HARDWARE DESIGN

Table IV lists the components. They were chosen from commonly available development-board parts so that the prototype can be rebuilt by other students. None of them has yet been assembled into a working unit

TABLE IV Proposed Hardware Components

Component	Role in the design
ESP32 development board	Main controller; GPIO input and output; BLE radio
Six tactile push buttons	Dots 1–6 of the Braille cell
One push button	Confirm (submit the entered pattern)
LED(s) with resistors	Visual status for supervisor and low-vision users
Buzzer (type not yet fixed)	Audible success and error signals
Vibration motor (optional)	Non-audible cue, after the idea in [7]
I2C OLED (optional)	Show target and score to the supervisor
Breadboard, jumper wires	Prototype assembly
USB cable and laptop	Power, programming, serial debugging
Rechargeable battery (planned)	Portable use

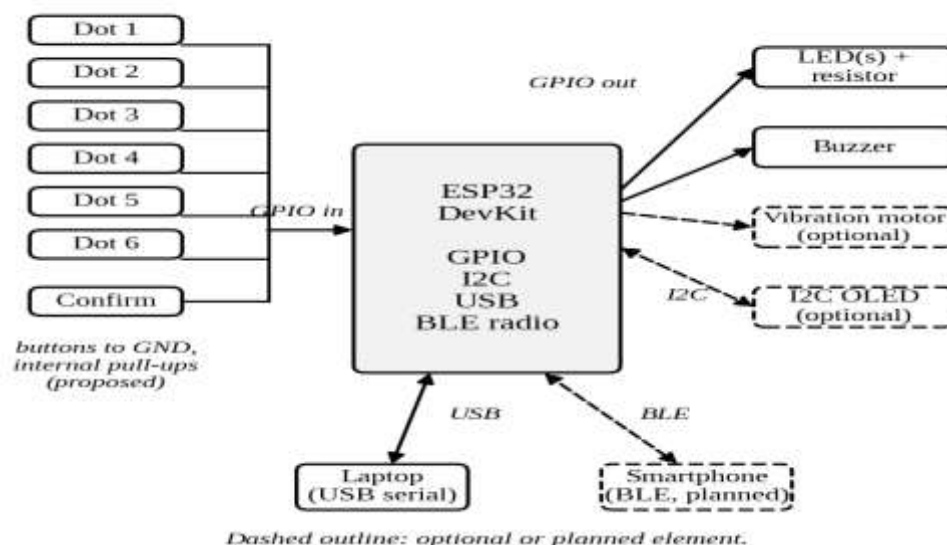


Fig. 2. Hardware connection diagram. Pin assignments have not been fixed



Input wiring. The seven buttons are intended to connect between a GPIO pin and ground, with the ESP32's internal pull-up resistors enabled, so that a pressed button reads low. This avoids external resistors on the input side; it is a design choice that will be confirmed when the circuit is built. The six dot buttons are to be arranged in a two-column, three-row layout that mirrors the cell, with the confirm button set apart from them.

Learner-facing ergonomics. Since tactile geometry affects recognition [8], button size, spacing, and a distinguishing mark (for instance on dot 1) will be examined during the usability tests rather than fixed in advance.

Feedback hardware. LEDs cannot help a learner who has no usable vision, so they serve mainly the supervisor and learners with some sight. For the learner, the buzzer is the primary channel, with the optional vibration motor as a second one. The buzzer type matters: a passive buzzer would allow different tones, an active one only on and off. This choice is open.

VIII. SOFTWARE DESIGN

1. Technologies

TABLE V Planned Software Technologies

Layer	Choice	Status
Firmware language	C/C++ in an Arduino-compatible environment for ESP32	Proposed
Debugging	USB serial output	Planned
Wireless link	BLE using the ESP32 radio [9]	Planned proof of concept
Smartphone application	Platform not chosen; speech output and lessons	Future work
Progress storage	Not decided (RAM only at first)	Open

No firmware has been written, so we do not list libraries or versions; doing so would amount to guessing.

2. Software Modules

- Input scanner: reads and debounces the seven buttons and latches dot presses.
- Encoder: turns the latched dots into the six-bit response R.
- Evaluator: compares R with the target T and returns the omitted and extra dot sets.
- Error record: per-dot counters for omissions and extras.
- Scheduler: picks the next target according to the active mode and the error record.
- Feedback driver: sets LEDs, buzzer tones, and the optional vibration motor.
- Communication: writes prompts and records to serial and, later, BLE.

3. Pattern Table

Characters are stored as masks in which bit $d-1$ stands for dot d . Table VI lists letters A–J with the mask written as dots 1 to 6 from left to right. Words are handled as sequences of such letters.



TABLE VI Dot Masks for Letters A–J,

Letter	Dots	Mask (1–6)	Letter	Dots	Mask (1–6)
A	1	100000	F	1,2,4	110100
B	1,2	110000	G	1,2,4,5	110110
C	1,4	100100	H	1,2,5	110010
D	1,4,5	100110	I	2,4	010100
E	1,5	100010	J	2,4,5	010110

IX. METHODOLOGY AND WORKING PRINCIPLE

1. Input Handling

A mechanical button can produce several rapid transitions during a single press, so a change of state is accepted only if the new level persists for a short interval, the length of which will be set experimentally. We propose to latch each dot as soon as it is pressed and to keep it latched until the confirm button is pressed. The learner then does not have to hold all the dots at the same moment, which would be awkward for a child with small hands. The alternative, reading the buttons at the instant of confirmation, would demand a chord, and we will compare the two in the usability tests. After evaluation the latches are cleared.

2. Session Flow

- The scheduler selects a target T according to the current mode.
- The target is shown or announced to the supervisor (serial, OLED, or later the app).
- The learner presses the dot buttons and then confirm.
- The encoder produces R and the evaluator compares it with T (Section X).
- The feedback driver signals the result; an incorrect answer allows another attempt.
- The error record is updated and the scheduler is consulted for the next target..

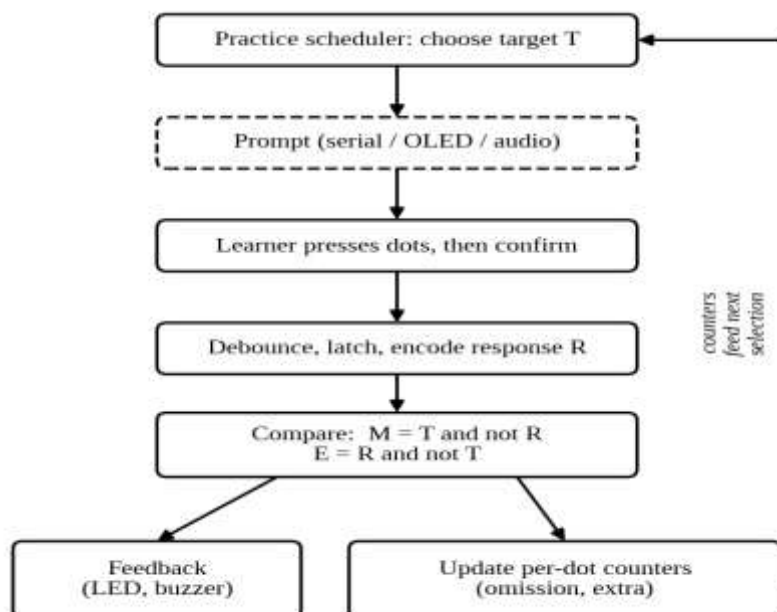


Fig. 3. Data flow of one practice attempt.



3. Learning Modes

- Guess the letter: a target letter is selected and the learner enters its dots.
- Spell words: a sequence of letters is presented and entered one cell at a time.
- Learn numbers: number symbols are introduced as a separate exercise set; their encoding will be fixed when this mode is designed.
- Free practice: the learner enters patterns without scoring, to get used to the buttons.

How many wrong attempts are allowed before the answer is revealed, and how the score is computed, are not yet decided.

X. DOT-LEVEL ERROR ANALYSIS AND TARGETED PRACTICE

This section describes our own proposed scheme. It is rule-based; nothing in it is a trained model. It has been worked out on paper only and has not been run on hardware or tested with learners.

1. Omitted and Extra Dots

Let T and R be the six-bit target and response. The omitted dots M and the extra dots E are

$$M = T \wedge \neg R, \quad E = R \wedge \neg T$$

and the response is correct exactly when both sets are empty:

$$\text{correct} \Leftrightarrow M = E = \emptyset \Leftrightarrow T = R \quad (2)$$

Fig. 4 gives an illustrative example: for target B (dots 1 and 2), a response of dots 1 and 4 gives $M = \{2\}$ and $E = \{4\}$. A plain pass/fail check would only report an error; the two sets tell the learner or supervisor what kind of error it was. The example is illustrative and does not come from any experiment.

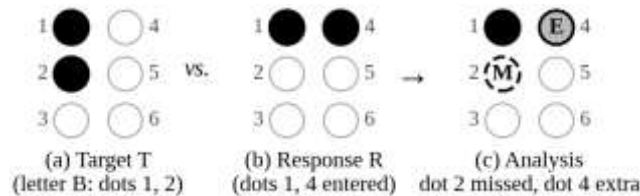


Fig. 4. Illustrative dot-level analysis for target B and a response of dots 1 and 4.

2. Per-Dot Error Record

For each dot d the controller keeps an omission score m_d and an extra-press score e_d . After every attempt, with a forgetting factor $0 < \lambda \leq 1$,

$$m_d \leftarrow \lambda m_d + [d \in M] \quad (3)$$

$$e_d \leftarrow \lambda e_d + [d \in E] \quad (4)$$

where $[*]$ is 1 if the condition holds and 0 otherwise. With $\lambda = 1$ the scores are plain counts; a smaller λ lets older mistakes fade. We have not chosen λ and expect to tune it during testing.

3. Choosing the Next Target

Let U be the set of characters the learner has unlocked and let c also denote the dot set of a character. A dot that the learner often omits is a risk for characters containing it; a dot that the learner often adds by mistake is a risk for characters that do not contain it. We therefore define



$$P(c) = (r(c) + \epsilon) / \sum_{c' \in U} (r(c') + \epsilon) \quad (6)$$

and draw the next target with probability

$$P(c) = (r(c) + \epsilon) / \sum_{c' \in U} (r(c') + \epsilon) \quad (6)$$

where a small $\epsilon > 0$ keeps every unlocked character possible, so that the learner does not meet only the hardest items and can still revise the others. Immediate repeats of the same target would be avoided by a separate rule. Whether this rule helps learners more than uniformly random selection is a question for the evaluation in Section XII, not something we assert.

4. Pseudocode and Code Sketch

Algorithm 1: one attempt

input: target T, response R (6-bit masks)

M $\leftarrow$ T and (not R); E $\leftarrow$ R and (not T)

if M = 0 and E = 0: signal success

else: signal error (dot detail via chosen channel)

for d in 1..6:

 m[d] $\leftarrow$ lambda*m[d] + (d in M ? 1 : 0)

 e[d] $\leftarrow$ lambda*e[d] + (d in E ? 1 : 0)

for c in U:

 r[c] $\leftarrow$ sum m[d] (d in c) + sum e[d] (d not in c)

next T $\leftarrow$ draw c in U with weight r[c] + eps

The pseudocode follows Eqs. (1)–(6).

// sketch only: not yet compiled or run on a board

uint8_t missed = target & ~resp & 0x3F;

uint8_t extra = resp & ~target & 0x3F;

bool correct = (missed | extra) == 0;

5. Reporting the Dot Detail

How much of M and E to tell the learner, and by what channel, is an open design question. Options include a spoken message from the smartphone app, a number of buzzer pulses, or showing the sets only to the supervisor. Which of these children can understand is something the usability tests are intended to settle.

6. Cost of the Computation

Per attempt the scheme needs a few bitwise operations, twelve counter updates, and a sum over at most a few dozen characters. We expect this to be negligible for the ESP32, but this is an expectation from the operation count and has not been measured.

Implementation Status and Plan

Current status- At the first review the project stood at planning, theory, and component selection; the physical prototype had not been started and no photographs or wiring records existed. Accordingly, this paper includes no photographs, screenshots, or circuit pictures. The figures are design diagrams drawn for this paper.

Planned build sequence- The project timeline for July–October 2026 groups the remaining work into five stages: (1) component selection and procurement; (2) ESP32 and button testing; (3) a BLE communication proof of concept; (4) a learning-logic and quiz prototype, in which the dot-level



analysis and practice selection of Section X would be implemented; and (5) initial testing and refinement.

Relation of code to hardware- The input scanner talks to the button GPIOs and returns latched dots; the evaluator and error record are hardware-independent and can be tested on a host computer before the board exists; the feedback driver is the only part that touches the LED and buzzer pins. This split is intended to let the logic be verified early. A sketch of the scanner follows.

```
// sketch only: pin numbers and debounce not yet defined
```

```
uint8_t latched = 0;
```

```
void scanDots() {
```

```
    for (int i = 0; i < 6; i++)
```

```
        if (debounced(DOT_PIN[i]) == LOW) // pressed
```

```
            latched |= (1 << i);
```

```
}
```

Input latching sketch (hypothetical debounced() helper).

XI. TESTING AND EVALUATION

This section is a proposed evaluation plan. None of the tests below has been run.

1. Functional Tests

TABLE VII Planned Functional Test Cases (Not Yet Run)

ID	Test	Expected outcome
T1	Press each dot button alone	Only that bit is latched
T2	Press one button repeatedly and quickly	One latched press; no spurious bits
T3	Enter the correct pattern for each of A–J	Success signal; <i>M</i> and <i>E</i> empty
T4	Omit one dot of a target	Error; <i>M</i> names the dot, <i>E</i> empty
T5	Add one wrong dot	Error; <i>E</i> names the dot, <i>M</i> empty
T6	Omit one dot and add another	Both sets correct
T7	Repeated errors on one dot	Its counter rises; others unchanged
T8	Counters set for one weak dot	Characters involving it are drawn more often
T9	Connect a phone over BLE	Target and result reach the phone
T10	Cycle correct and wrong answers	Distinct, consistent feedback each time

2. Exhaustive Check of the Logic

Because the comparison works on six bits, it can be checked exhaustively. There are 63 non-blank targets and 64 possible responses, so 4032 target–response pairs. The plan is to compare the evaluator’s output for every pair with an independent reference written separately, on a host computer first and then on the board.



3. Measurements

TABLE VIII Planned Measurements

Measure	How it would be obtained
Button-detection accuracy	Scripted presses against logged states
Press-to-feedback latency	Timestamps on the board, cross-checked externally
False-trigger rate after debouncing	Counted over long idle and rapid-press runs
Dot-level classification correctness	Exhaustive check above
Selection behaviour	Frequency of weak-dot characters before and after errors
BLE message delivery	Messages sent against received
Usability	Button discrimination and feedback comprehension, by observation

4. Usability Study

A small supervised session is planned to check whether users can tell the six buttons apart, whether the feedback is understood, and whether targeted selection behaves sensibly in practice. A comparison of targeted against uniformly random selection is also planned. Any involvement of children or visually impaired participants requires institutional permission, informed consent, and the advice of accessibility experts. No participants have been recruited. We make no claim about clinical or educational effectiveness, and any figures will be reported only as measured.

XII. RESULTS AND DISCUSSION

No experimental results are available. The prototype has not been built, so there are no accuracy, latency, or user-study figures. We have deliberately not estimated them. Once Section XII has been carried out, the paper will report the measured values, with the number of trials and conditions, and the cost of the parts as actually purchased.

- What the design work does allow is a discussion of problems that the prototype must address:
- Feedback for non-sighted users. LEDs help supervisors only; the audible and vibration channels carry the real feedback.
- Latching versus chording. Latching is kinder to small hands but may be less like writing Braille on a slate; we plan to compare them.
- Volatile error record. If counters live in RAM they vanish at power-off. Persistent storage or the smartphone app would be needed for long-term tracking.
- Cold start. With empty counters, the rule in Eq. (6) behaves like uniform selection; targeting emerges only after some attempts.
- Choice of λ and ϵ . Too small a λ forgets too quickly; too small an ϵ over-concentrates practice. Both need tuning.

Limitations

- The console trains forming Braille patterns. Buttons do not reproduce raised dots, so reading by touch is not practised.
- Nothing has been built or tested; every behaviour described is a design intention.
- The planned content is limited to single-cell letters and numbers; contractions, punctuation, and other languages are not covered.
- The practice rule is a heuristic with unset parameters, and its benefit is unproven.
- The supervisor display, BLE link, and smartphone app are not yet available.



- No usability or learning-outcome data exist, and the wider literature on such outcomes is itself weak [1].

Future Enhancements

- Build and test the prototype, then publish the measurements of Section XII.
- Replace the breadboard with a custom PCB in a child-safe enclosure.
- Develop the smartphone application with spoken prompts, lessons, and progress reports over BLE.
- Add Wi-Fi logging, as in MQTT-based designs [2], for teacher dashboards.
- Add a small tactile output cell for two-way practice, drawing on the display work reviewed in Section III and on the tactile-geometry findings of [8].
- Extend the content to punctuation, contractions, and other languages once the curriculum is validated.
- Run a consented usability study with accessibility experts and, if appropriate, special-education staff

XIII. CONCLUSION

We have described the design of BrailleLearn, a low-cost ESP32-based console in which six buttons and a confirm key stand in for the Braille cell. The central idea is simple: since a response is a six-bit value, it can be compared with the target dot by dot, omitted and extra dots can be told apart, and per-dot error counts can steer the choice of the next exercise. The reviewed literature is mostly about refreshable displays and conversion systems, and we did not find this combination of features in it, though our survey was not exhaustive. The project is at the design stage: the prototype, firmware, and evaluation are still to come, and we have kept this paper to what the supplied project material supports.

Acknowledgment

The authors thank their guide, Dr. A. Manikandan, for guidance on the project

REFERENCES

1. A. Sutherland et al., "Effectiveness of technology for braille literacy education for children: A systematic review," *Disabil. Rehabil. Assist. Technol.*, 2022.
2. O. I. Ramos-García et al., "An IoT Braille display towards assisting visually impaired students in Mexico," *Eng. Proc.*, vol. 27, no. 1, p. 11, 2022.
3. A. S. Walker and S. Sangelkar, "Design exploration of affordable refreshable Braille display technology for low-income visually impaired users," in *Proc. ASME IDETC/CIE*, 2017.
4. A. Ghosh et al., "Brindle: A low-cost refreshable Braille display with slider mechanism," in *Proc. IEEE WIECON-ECE*, 2023.
5. Liu et al., "Development and evaluation of refreshable Braille display and active touch-reading system for digital reading of the visually impaired," 2024. [Full author list and publication venue to be added.]
6. M. M. H. Saikot and K. R. I. Sanim, "Refreshable Braille display with adjustable cell size for learners with different tactile sensitivity," *IEEE Trans. Haptics*, vol. 15, no. 3, pp. 582–591, 2022, doi: 10.1109/TOH.2022.3184265.
7. M. O. A. Aqel, A. Issa, A. W. Harb, and J. Shehada, "Development of vibro-tactile Braille display and keyboard," in *Proc. Int. Conf. Promising Electron. Technol. (ICPET)*, 2019, pp. 28–33, doi: 10.1109/ICPET.2019.00013.



8. C. Mascle, C. Jouffrais, G. Kaminski, and F. Bara, "Tactile pictograms displayed on pin array tablets for blind children," in Proc. IEEE Haptics Symp. (HAPTICS), 2022, pp. 1–5, doi: 10.1109/HAPTICS52432.2022.9765607.
9. Maier, A. Sharp, and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the Internet of Things," in Proc. 7th Int. Conf. Internet Technol. Appl. (ITA), 2017, doi: 10.1109/ITECHA.2017.8101926.
10. F. Ahmed, A. R. Choudhury, A. Rakshit, and M. Z. Hasan, "An IoT based system for printing Braille letter from speech," in Proc. IEEE Region 10 Symp. (TENSYP), 2020, doi: 10.1109/TENSYP50017.2020.9230734.
11. [Authors to be added], "IoT-driven accessibility: A refreshable OCR-Braille solution for visually impaired and deaf-blind users through WSN," J. Economy Technol., vol. 2, pp. 128–137, 2024.
12. P. S. Talukdar, B. Hazarika, J. Sharma, and R. Matam, "An IoT based smart assistive device for the visually impaired," in Proc. IEEE Region 10 Symp. (TENSYP), 2020, doi: 10.1109/TENSYP50017.2020.9231026.
13. A. Patil, A. Borse, A. Borse, A. Phad, and S. Khiani, "A low-cost IoT based navigation assistance for visually impaired person," in Proc. 8th Int. Conf. Commun. Electron. Syst. (ICCES), 2023, doi: 10.1109/ICCES57224.2023.10192887.
14. G. Latif, G. Ben Brahim, S. E. Abdelhamid, R. Alghazo, G. Alhabib, and K. Alnujaidi, "Learning at your fingertips: An innovative IoT-based AI-powered Braille learning system," Appl. Syst. Innov., vol. 6, no. 5, art. 91, 2023.
15. D. M. de Lemos, R. Barwaldt, and C. R. Rocha, "Didactic resources for the visually impaired: A systematic review of the literature," in Proc. IEEE Frontiers in Education Conf. (FIE), 2023.
16. H. L. A. I. Udana, B. M. N. U. Mendis, K. Herath, et al., "Tactile device for Braille-based communication in inclusive education," in Proc. 9th Int. Conf. Inf. Technol. Res. (ICITR), 2024, doi: 10.1109/ICITR64794.2024.10857797.