Braillear: An Assistive Touch-Based Interactive Learning System Using TTP223 and Raspberry Pi Pico for the Visually Impaired

Abstract

Braillear is a smart assistive device designed to transform the way visually impaired individuals learn alphabets and numbers. The system combines capacitive touch sensors (TTP223) with a Raspberry Pi Pico microcontroller to offer tactile input and corresponding auditory feedback. This innovative approach facilitates an intuitive and inclusive educational environment, enabling users to interact with the system by touch and receive real-time audio responses. The primary motivation stems from the global need for accessible learning solutions, particularly in regions with limited access to specialized tools. Braillear is cost-effective, scalable, and capable of standalone operation with future enhancements.

Keywords

Assistive Technology, Raspberry Pi Pico, TTP223 Touch Sensor, Braille Learning, Visual Impairment, Embedded Systems, Human-Centered Design

I. Introduction

According to the World Health Organization, more than **19 million children worldwide** suffer from visual impairment, with **1.4 million classified as blind**. In India, the number is approximately **320,000 children**. Despite technological advancements, there remains a significant gap in accessible education tools for the blind. This project, *Braillear*, addresses this challenge by creating a system that bridges touch interaction with audio output, allowing blind learners to identify letters and numbers through tactile sensors and hear their auditory equivalents instantly.

II. Problem Statement and Motivation

The lack of engaging, portable, and easy-to-use Braille learning devices limits the academic growth of visually impaired children. Traditional Braille books are bulky, and most digital Braille solutions are expensive. *Braillear* aims to:

- Provide an **interactive audio-assisted Braille learning tool**
- Utilize **cost-efficient components**

- Allow **real-time learning** without requiring sight
- Encourage **inclusive education** at grassroots levels

III. System Overview

A. Key Functionalities:

- Touch any letter (A–Z) or number (0–9) via the capacitive sensors
- The system identifies the touch and sends it to a microcontroller
- An audio file is played stating the touched character

B. System Workflow:

1. **User Touches**: A letter mapped TTP223 sensor.
2. **Signal Sent**: Touch is detected as HIGH signal.
3. **Pico Reads GPIO**: Each GPIO pin is mapped to a character.
4. **UART Transfers Data**: Pico sends data to laptop via serial port.
5. **Laptop Audio Output**: Python program plays respective audio.

IV. Hardware Components

S. No.	Component	Quantity	Description
1	Raspberry Pi Pico	1	Dual-core ARM Cortex-M0+
2	TTP223 Touch Sensors	36	Capacitive input sensors for A–Z, 0–9
3	Rechargeable Battery	2	Power source
4	Buttons	3	Input control
5	Battery Indicator	1	Monitors battery status
6	Jumper Wires	—	Connection medium
7	Type-C Charger	1	Charging cable

V. Hardware Implementation

A. TTP223 Capacitive Touch Sensor

- Voltage: 2.0V – 5.5V
- Outputs HIGH when touched
- Digital output mode
- Low power consumption
- Used to detect and respond to user finger touch

Each sensor is connected to an individual GPIO pin on the Raspberry Pi Pico or via the ESP32 controller. A total of 26 sensors represent A–Z letters.

B. Raspberry Pi Pico

- **Microcontroller**: Dual-core ARM Cortex-M0+
- **GPIO Pins**: 26 usable for touch inputs
- **Chosen for**: Cost-efficiency, flexibility, and compactness
- Each GPIO pin (GP0–GP25) maps to a letter (A–Z)

VI. Software Design

A. Microcontroller (Pico) Code

Written in MicroPython, the code initializes pins, reads touch inputs, applies debouncing logic, and communicates via UART to the host system.

python

CopyEdit

```
# Key imports and initializations

from machine import Pin, UART

import utime


# GPIO to Letter Mapping

pin_numbers = list(range(26))

pin_letter_mapping = {i: chr(65+i) for i in range(26)}
```

```
pins = [Pin(i, Pin.IN, Pin.PULL_UP) for i in pin_numbers]
uart = UART(0, 9600)

# Debounce logic
def button_pressed(pin):
    ...
    uart.write(triggered_letter.encode())

# Interrupts and Loop
for pin in pins:
    pin.irq(...)

while True:
    utime.sleep_ms(1000)
```

B. Laptop Python Code (Audio Playback)

Utilizes the PyDub and Serial libraries to play the correct audio file when a letter is received via the serial port.

```
python
CopyEdit
from pydub import AudioSegment
from pydub.playback import play
import serial

ser = serial.Serial('COM3', 9600)
while True:
    data = ser.readline().strip().decode('utf-8')
```

```
if data == 'A':
    sound = AudioSegment.from_mp3('A.mp3')
    play(sound)
```

VII. Circuit Design and GPIO Mapping

Each GPIO pin of the Raspberry Pi Pico connects to a TTP223 sensor. Power and GND are routed carefully to maintain voltage stability. The expansion board enables interfacing more sensors without GPIO overflow.

Scan QR code from the presentation to see the detailed circuit.

VIII. Challenges Faced

Issue	Description
Wire Maintenance	Tangling and sensor misfiring during long use
Cross-Connections	Misreading of sensors due to improper wiring
Sensor Overheating	Faulty connections led to sensor damage
Serial Port Access	Laptop permissions restricted serial communication
Debugging GPIO Pins	Specific pins (F, G, H...) required custom handling
Code Repetition	Needed functions to handle each sensor individually
Debouncing & False Triggers	Multiple unintended reads on single press
UART Instability	Data loss in long usage sessions

IX. Future Enhancements

A. Mobile App Integration

- Audio feedback customization
- Sensitivity settings and language switching
- Braille practice modules and user forums

B. Standalone Operation

- Eliminate the need for external laptop
- Onboard memory for storing audio files and lessons
- Small display and navigation buttons

C. Hardware Enhancements

- Compact, portable casing
- Efficient power management
- Wireless audio (Bluetooth earphones)

X. Conclusion

Braillear is a step toward democratizing education for visually impaired individuals through technology. With a thoughtful blend of affordability, simplicity, and accessibility, it demonstrates how embedded systems can empower underserved communities. Future iterations promise to enhance autonomy, connectivity, and learning flexibility.

References

1. World Health Organization. "World Report on Vision", [Online]. Available: https://www.who.int
2. Raspberry Pi Foundation, "Raspberry Pi Pico Documentation", [Online]. Available: https://www.raspberrypi.com/documentation/
3. PyDub Library, [Online]. Available: https://pydub.com
4. TTP223 Datasheet, [Online]. Available: https://www.electronicwings.com
5. MicroPython Documentation, [Online]. Available: https://docs.micropython.org