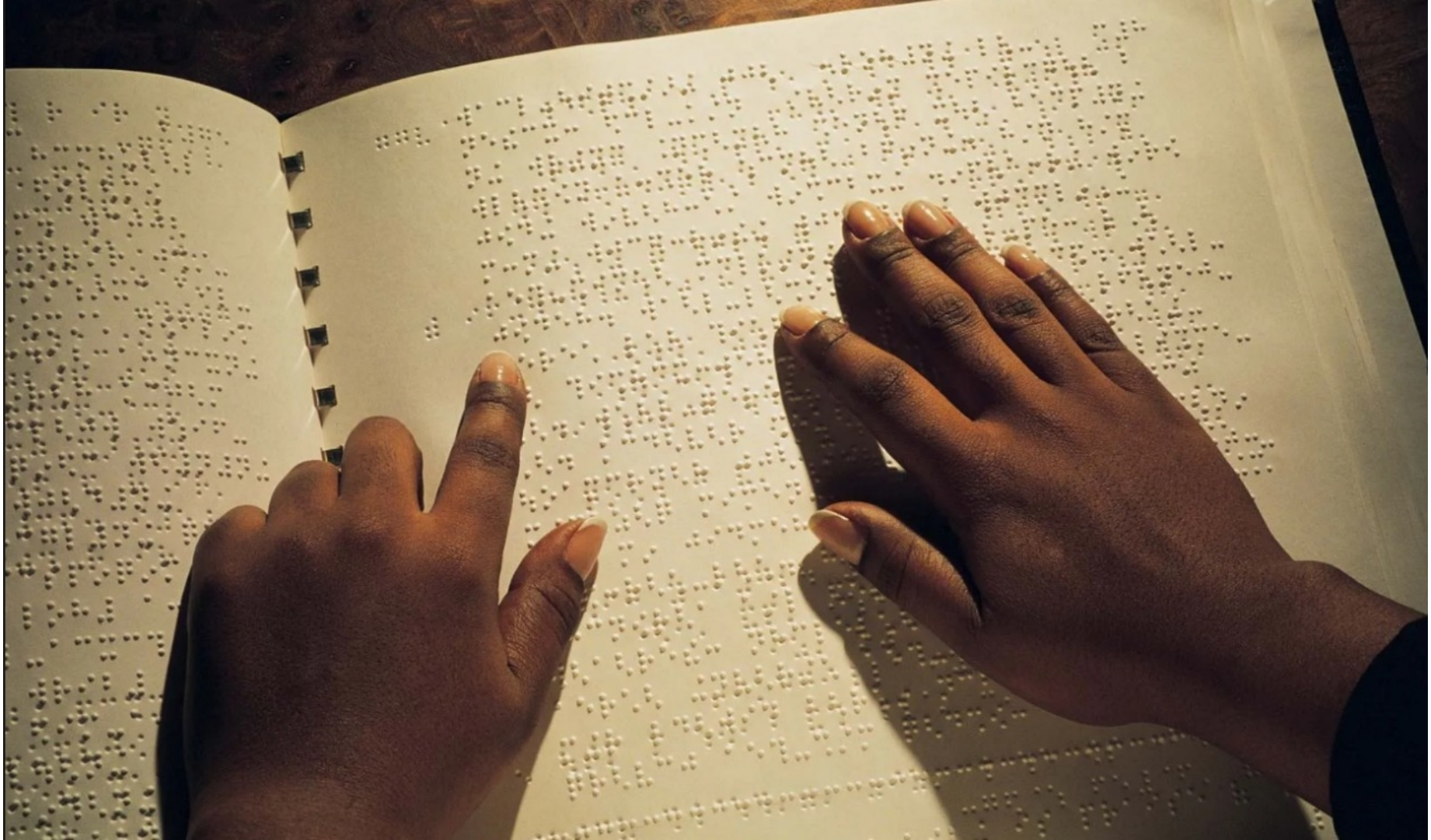
Senses. Are. Precious.


24
23



# S.E.N.S.E

*SERVO ENABLED NAVIGATION SYSTEM FOR EMPOWERMENT*

Under The Supervision of
Mr. Soumya Suvra Khan, Assistant Professor, MSIT


- Neeraj P Pillai
- Kuntal Barik
- Harshil D Sheth
- Abdul Basit
- Ahona Banerjee
- Aryan Singh

# Content

# Introduction

This project transforms website text into tactile Braille in real time for visually and auditory impaired users

A Python script scrapes text from a given URL and sends it to an Arduino, which uses servo motors to raise and lower pins to form 3x2 Braille cells.

Each letter appears sequentially on a tactile display, enabling interactive, accessible reading.

# Field Survey

| SL No. | Reference | Focus | Technological Approach | Unique Features | Research Gap |
|---|---|---|---|---|---|
| 01 | R. Patel, "Performance Evaluation of Servo Motors in Robotic Applications" | Evaluates the performance of servo motors in robotic systems, with a focus on speed, accuracy, and torque characteristics. | Examines servo motors in robotic arms, comparing various models and their performance in real-world tasks. | Provides empirical performance data on servo motors in robotics, offering valuable insights for selecting appropriate motors for specific tasks. | Does not address advanced servo motor control methods or applications in complex multi-robot systems. |
| 02 | Glez-Peña et al., "Web Scraping Techniques and Applications: A Literature Review" | Provides an extensive review of various web scraping techniques, tools, and applications in different domains. | Surveys the evolution of web scraping methods, focusing on the tools and frameworks used for large-scale data extraction. | Comprehensive review of various techniques used across industries such as finance, bioinformatics, and more. | Lacks real-time web scraping applications; does not explore specific challenges in dynamically changing content. |
| 03 | H. L. Zohdi, J. M. Harris, "Arduino as a Tool for Teaching Embedded Systems: A Case Study" | Focuses on using Arduino as an educational tool for teaching embedded systems, emphasizing its accessibility and simplicity. | Uses Arduino as a hands-on platform to teach embedded systems and programming. | Emphasizes Arduino's value in education due to its affordability, ease of use, and versatility in learning embedded systems. | Does not explore how Arduino compares to other platforms like Raspberry Pi in terms of educational effectiveness. |
| 04 | Sawant et al., "Text to Braille Conversion System" | Focuses on developing a system to convert standard text to Braille to assist visually impaired users in accessing written content. | Utilizes algorithmic approaches for character-to-Braille mapping and translation. | Simple and scalable conversion methodology suitable for text from various sources, ensuring a broad application. | Does not explore real-time web-based content conversion or the integration with dynamic systems like web scraping. |

# Proposed Work

This project presents an innovative assistive technology that extracts textual content from websites and converts it into tactile Braille output , It does so by executing the following steps:-

1. Web Scraping - Takes URL as input and extracts all the text from the website and stores it in a text file (python code).
2. Send to Arduino - processes the scrapped text and sends it to Arduino line by line (python code).
3. Braille representation- the line is then represented in braille with the help of servo motors (c code).

sense/web_scrapping.py

```python
import requests
from bs4 import BeautifulSoup

def scrape_text_from_url(url, output_file):
    try:
        response = requests.get(url)
        response.raise_for_status()
    except Exception as e:
        print(f"Error fetching URL: {e}")
        return

    soup = BeautifulSoup(response.text, "html.parser")

    for script_or_style in soup(["script", "style"]):
        script_or_style.decompose()

    text = soup.get_text(separator='\n')
    lines = [line.strip() for line in text.splitlines()]
    text = '\n'.join(line for line in lines if line)

    with open(output_file, "w", encoding="utf-8") as f:
        f.write(text)
    print(f"Text content saved to {output_file}")

if __name__ == "__main__":
    url = input("Enter the URL to scrape: ")
    scrape_text_from_url(url, "scraped_text.txt")
```

sense/send_to_ard.py

```python
import serial
import time

def send_file_to_arduino(serial_port, baud_rate, filename):
    try:
        ser = serial.Serial(serial_port, baud_rate, timeout=1)
        time.sleep(2)  # Let Arduino reset
    except Exception as e:
        print(f"Error opening serial port: {e}")
        return

    try:
        with open(filename, 'r', encoding='utf-8') as f:
            for line in f:
                line = line.strip()
                if not line:
                    continue
                ser.write((line + '\n').encode())
                print(f"Sent: {line}")
                time.sleep(1)  # Delay to let Arduino display Braille
    except Exception as e:
        print(f"Error: {e}")
    finally:
        ser.close()

if __name__ == "__main__":
    port = input("Enter COM port (e.g., COM3): ")
    send_file_to_arduino(port, 9600, "scraped_text.txt")
```

sense/main/main.ino

```
#include <Servo.h>

const int numDots = 6;
const int servoPins[numDots] = {2, 3, 4, 5, 6, 7};
Servo brailleServos[numDots];

const int dotUp = 90;
const int dotDown = 0;

struct Braille {
  char character;
  byte pattern;
};

const Braille brailleMap[] = {
  {'a', B000001}, {'b', B000011}, {'c', B001001}, {'d', B011001},
  {'e', B010001}, {'f', B001011}, {'g', B011011}, {'h', B010011},
  {'i', B001010}, {'j', B011010}, {'k', B000101}, {'l', B000111},
  {'m', B001101}, {'n', B011101}, {'o', B010101}, {'p', B001111},
  {'q', B011111}, {'r', B010111}, {'s', B001110}, {'t', B011110},
  {'u', B100101}, {'v', B100111}, {'w', B111010}, {'x', B101101},
  {'y', B111101}, {'z', B110101},
  {'.', B010011}, {',', B000011}, {';', B000111}, {':', B010011},
  {'!', B010111}, {'?', B010110}, {'-', B000100}, {'\'', B000010},
  {'"', B000110}, {'(', B011110}, {')', B011110}, {' ', B000000}
};

const int brailleCount = sizeof(brailleMap) / sizeof(Braille);
```

```
void loop() {
  if (Serial.available()) {
    String line = Serial.readStringUntil('\n');
    line.trim();
    if (line.length() > 0) {
      // For each character in the input line
      for (int i = 0; i < line.length(); i++) {
        char c = tolower(line.charAt(i));
        byte pattern = getBraillePattern(c);

        for (int j = 0; j < numDots; j++) {
          if ((pattern >> j) & 1) {
            brailleServos[j].write(dotUp);
          } else {
            brailleServos[j].write(dotDown);
          }
        }

        delay(1500);
      }
      // After a full line, lower all dots before next line
      for (int i = 0; i < numDots; i++) {
        brailleServos[i].write(dotDown);
      }
      delay(1000);
    }
  }
}
```

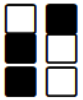
```
void setup() {
  Serial.begin(9600);

  for (int i = 0; i < numDots; i++) {
    brailleServos[i].attach(servoPins[i]);
    brailleServos[i].write(dotDown);
  }
}

byte getBraillePattern(char c) {
  for (int i = 0; i < brailleCount; i++) {
    if (brailleMap[i].character == c) {
      return brailleMap[i].pattern;
    }
  }
 return B000000;
}}
```

# Experimental Result:

During the demonstration of our Braille output device using the sample web page

https://0thorderlogic.github.io/sense , the following key steps and corresponding outputs were observed:

1. Input URL
2. Web Scraping and Text Processing
3. Text sent to circuit for Braille output
4. Output corresponding to input

| Letter | Binary Braille Code | 3x2 Braille Matrix (Dots) |
|---|---|---|
| S | 001110 |  |
| E | 100010 |  |
| N | 101110 |  |
| S | 001110 |  |
| E | 100010 |  |

**Legend:**

■ — Raised dot (High voltage signal)

□ — Flat dot (No signal)

# Performance Analysis

The performance analysis of the S.E.N.S.E prototype showcases promising outcomes in transforming web content into tactile Braille output. Our testing phase validated the system's core functionality—accurately scraping, processing, and converting real-time online text into mechanical Braille letters using servo-actuated hardware. The integration of Python scripting with Arduino-based control ensured precise signal translation, resulting in consistent and responsive Braille representation. Despite hardware limitations, such as restricted character output and latency in servo response, the device successfully demonstrated real-time tactile translation, indicating the feasibility of scalable implementation. These findings reinforce the potential of combining web technologies and embedded systems to create accessible solutions for **deaf-blind** users.

# Limitations and Improvements

1. Can display only one word at a time(can be solved using Arduino Mega)
2. Requires intervention to run initially
3. Although produced in accordance with first principles, the cost can still be a limiting factor to adoption
4. Lack of funding for such programs
5. Limited to ‘Latin script’
6. Present prototype has much margin between the braille representation dots

# Conclusion

The S.E.N.S.E project bridges accessibility gaps by converting web content into tactile Braille for users with visual and hearing impairments. It combines web scraping, a Python script, and Arduino-controlled servo motors to render 3x2 Braille in real time. This cost-effective, scalable device empowers users with independence and seamless access to vast online information, promoting digital inclusion and equity.

# Reference


1. A. Morgavi and A. Morando, "Dynamic Braille Translation: A Real-Time Assistive Web Tool for the Visually Impaired," *Journal of Assistive Technologies*, vol. 15, no. 2, pp. 100–112, 2019.
2. M. Rivera and E. Morales, "IoT-Based Educational Braille Display for Visually Impaired Students," *International Conference on Human-Centered Computing*, pp. 89–95, 2021.
3. N. Sawant and M. Deshmukh, "A Rule-Based Text-to-Braille Converter Using Java for Low-Resource Environments," *International Journal of Assistive Technology*, vol. 4, no. 1, pp. 27–34, 2022.
4. S. Khandare and S. Dorle, "Braille Accessibility through OCR-Based Printed Material Interpretation," *IEEE Int. Conf. on Smart Systems and Applications*, pp. 225–231, 2020.
5. J. Zhang, L. Chen, and W. Liu, "A Compact Electromagnetic Braille Display Using Layered Actuators," *Sensors and Actuators A: Physical*, vol. 316, pp. 112–119, 2021.
6. R. Teixeira, M. Silva, and A. Dias, "Survey on Tactile Technologies for Refreshable Braille Displays," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–29, 2020.