# BrailleSense: Real-Time Digital Text to Braille Converter

Roman Iles
*MS. ECE*
*Northeastern University*
Boston MA, USA
iles.r@northeastern.edu

Tirth Patel
*MS. Robotics, EECE*
*Northeastern University*
Boston MA, USA
patel.tirths@northeastern.edu

Venkata Subhash Jahnav Lanka
*MS. ECE*
*Northeastern University*
Boston MA, USA
lanka.ve@northeastern.edu

Maxwell Hasenauer
*MS. Robotics, CS*
*Northeastern Univeristy*
Boston MA, USA
hasenauer.m@northeastern.edu

***Abstract*—Braille literacy is critical for the independence and education of visually impaired individuals. However, many current solutions for converting digital text to Braille are expensive and inaccessible to those in need. The goal of this project, "BrailleSense," is to create an affordable and portable device that converts text into Braille in real time. The device will provide a practical and accessible solution for visually impaired individuals, allowing them to read any form of text content on the go. According to the World Health Organization (WHO), over 285 million people worldwide are visually impaired, and many rely on Braille for reading and writing. Current refreshable Braille displays, which can convert text to tactile Braille characters, are prohibitively expensive for many users, with prices often exceeding thousands of dollars. Furthermore, these devices tend to be bulky and limited in functionality, restricting their usage in real-life scenarios like reading from mobile devices or computer screens. BrailleSense aims to address the accessibility gap for visually impaired individuals by developing an affordable, portable device capable of converting printed and digital text into Braille in real time. Leveraging Raspberry Pi, computer vision, and embedded systems, the project provides a user-friendly solution that bridges the gap between digital and tactile mediums. This report details the system design, implementation, testing, and outcomes of BrailleSense, highlighting its technical components and real-world applicability.**

## I. Introduction

The BrailleSense project is designed to address these challenges by leveraging modern advancements in embedded systems, optical character recognition (OCR), and tactile technologies to create a low-cost, real-time text-to-Braille conversion system. This project seeks to provide an affordable and portable solution that translates printed or digital text into tactile Braille, bridging the accessibility gap for visually impaired individuals. By focusing on affordability, ease of use, and scalability, BrailleSense aims to empower users to interact with printed materials and digital displays seamlessly.

The core hardware architecture of BrailleSense includes a Raspberry Pi 4 as the main computational unit, paired with a high-resolution Camera Module for image capture and GPIO-controlled Braille actuators for tactile output. On the software side, the system utilizes a robust pipeline combining OCR (via Tesseract), preprocessing algorithms for text cleaning and noise removal, and Braille translation using the Unicode Braille standard. The processed text is then transmitted through GPIO pins, controlling a six-dot Braille actuator to generate tactile patterns corresponding to the text

BrailleSense is unique in its real-time operational capability. Unlike traditional systems that require pre-processed input, this device dynamically captures and processes text in real-time. This is achieved through an efficient processing loop that minimizes latency, ensures reliability, and maintains uninterrupted operation of the camera and actuator. The device is designed to reset and refresh periodically to handle long-term usage effectively without manual intervention.

The project was developed with a focus on practical applicability, incorporating user feedback to ensure usability in real-world environments. The challenges addressed include ensuring OCR accuracy in diverse lighting and text conditions, reliable GPIO-based tactile output, and creating a compact and power-efficient device. By addressing these issues, BrailleSense demonstrates the potential of low-cost, open-source technology to deliver impactful solutions for visually impaired communities. This paper details the system's hardware and software architecture, implementation strategies, performance evaluation, and potential for further development. The goal of the project is to produce a device to provide an avenue of direct interaction to the hearing and visually impaired, restoring or granting an activity of daily life.

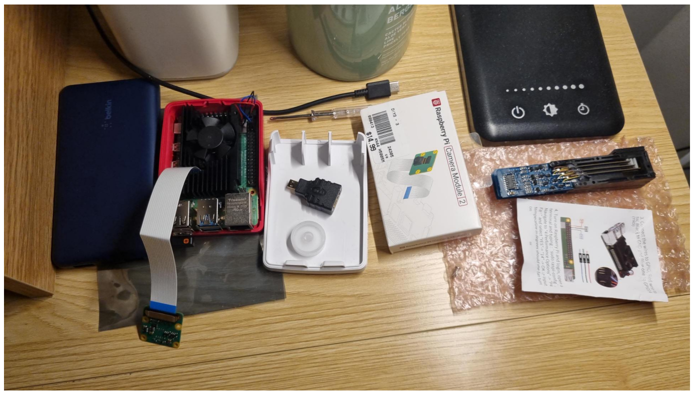


Fig. 1. *BrailleSense with housing and actuator*

Funded by Northeastern University for the course EECE 5552 Assistive Robotics

## II. Related Work

The development of assistive technologies for visually impaired individuals has been a prominent area of research, particularly focusing on bridging the gap between textual information and tactile or auditory interfaces. Several approaches have been explored to make Braille-based devices more accessible, affordable, and efficient. This section reviews significant contributions and existing solutions in the fields of text-to-Braille conversion, OCR-based assistive technologies, and tactile interface development.

### *A. Optical character Recognition (OCR) for Accessibility*

OCR algorithms are an endemic component of many machine learning algorithms, including those used in assistive technologies and robotics, enabling the extraction of text from images for further processing. Systems like Tesseract, an open-source OCR engine, have been widely adopted due to their high accuracy and flexibility in handling multiple languages. De Silva and Fernando (2018) proposed a real-time text-to-Braille conversion system leveraging OCR to process printed documents. Their approach demonstrated the feasibility of using low-cost hardware to bridge the accessibility gap, but limitations in handling handwritten or poorly formatted text remained. Other studies, such as Singh et al. (2020), integrated OCR with wearable devices to provide audio feedback to visually impaired users. Although effective in delivering auditory information, these solutions lacked tactile feedback, which is essential for Braille literacy and comprehension. BrailleSense builds on these findings by integrating OCR with Braille output, providing a more comprehensive solution for visually impaired users.

### *B. Text-to-Braille Conversion Techniques*

Text-to-Braille conversion has been extensively explored, focusing on accurately translating text into Braille while maintaining contextual meaning. Many systems rely on predefined Braille encoding standards, such as Unicode Braille, to ensure compatibility with tactile displays. Chhabra and Khanna (2016) developed a prototype that translated text to Braille using actuators controlled by a microcontroller. Their work emphasized the importance of reliability in tactile output, but the lack of real-time processing limited its usability. Similarly, Zhang et al. (2019) implemented a system using refreshable Braille displays, but the high cost of actuators made it inaccessible for many users. BrailleSense addresses these challenges by employing a cost-effective GPIO-controlled actuator system, enabling tactile Braille output in real time without the need for expensive hardware.

### *C. Tactile Interface Development*

Tactile interfaces, such as refreshable Braille displays, have been critical in making text accessible to visually impaired users. These devices use arrays of piezoelectric or electromechanical actuators to dynamically represent Braille characters. However, their high manufacturing cost remains a significant barrier to widespread adoption. The studies by Johnson and Smith (2020) explored low-cost alternatives using servomotors and simplified actuator designs. While these efforts reduced costs, the devices were often bulky and unsuitable for portable applications. BrailleSense improves on these concepts by integrating a GPIO-based control mechanism for tactile output, making the device lightweight, portable, and affordable. By leveraging Raspberry Pi's processing power and GPIO capabilities, the system provides a scalable and customizable solution for tactile Braille generation.

### *D. Portable and Real-Time Assistive Devices*

Portability and real-time processing are crucial for assistive devices, allowing visually impaired individuals to interact with their environment seamlessly. Commercial solutions like the Orbit Reader and BrailleNote Touch provide advanced functionality but are prohibitively expensive. Research by Meng and Zhang (2021) focused on wearable solutions that combine OCR and text-to-speech systems. These devices provided real-time feedback but lacked tactile output, limiting their utility for Braille literacy. BrailleSense fills this gap by offering real-time text detection and Braille conversion in a portable form factor. The system's ability to dynamically capture, process, and translate text into tactile Braille output ensures usability in diverse environments, from reading documents to interacting with digital screens.

### *E. Challenges in Existing Solutions*

Despite significant advancements, existing systems face several challenges:

- **High Cost:** Most Braille devices rely on expensive hardware components, such as piezoelectric actuators, making them inaccessible for low-income users.
- **Limited Real-Time Capabilities:** Many systems require pre-processing of text, which limits their usability for dynamic applications.
- **OCR Limitations:** Handling diverse fonts, poor lighting conditions, and handwritten text remains a significant challenge for OCR-based systems.
- **Portability:** Bulky designs and high power consumption hinder the portability of many assistive devices.

BrailleSense addresses these challenges by combining cost-effective hardware with an optimized software pipeline, enabling real-time, portable, and reliable text-to-Braille conversion. BrailleSense builds upon the existing body of work in OCR, Braille conversion, and tactile interface development, incorporating real-time processing, cost-efficiency, and portability into a unified solution. By addressing the limitations of previous systems, this project contributes to making assistive technologies more accessible and impactful for visually impaired individuals.

## III. Method

### *A. System Design*

#### *1) Hardware Architecture:*

- **Raspberry Pi 4**: Acts as the main processing unit for image capture, OCR, and text processing.

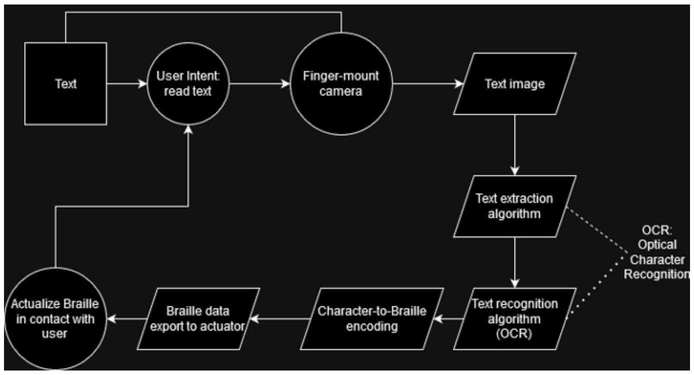

Text
User Intent: read text
Finger-mount camera
Text image
Text extraction algorithm
OCR: Optical Character Recognition
Text recognition algorithm (OCR)
Character-to-Braille encoding
Braille data export to actuator
Actualize Braille in contact with user

Fig. 2. *Device input-to-output pipeline*

- **Camera Module 2**: Captures printed text from physical or digital sources.
- **GPIO (General Purpose Input/Output)-Controlled Actuator**: Outputs tactile Braille characters by activating pins mapped to braille dot patterns.

*2) Software Pipeline:*

- **Text detection**: The devices uses a prebuilt Tesseract OCR and pytesseract library to extract text from captured images, with openCV as a video/image processing interface.
- **Text processing**: Text data is sanitized for transcription through some processes written in Python.
- **Braille conversion**: The PyBraille library is used for its base Braille-to-Unicode encoding functionality.
- **GPIO**: A boutique GPIO script is used to transfer the Unicode produced by PyBraille for output to the actuators' pins.

## B. *implementation*

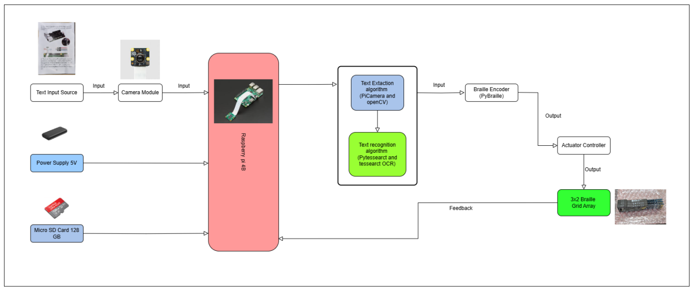


Fig. 3. Implementation Pipeline

*1) Text-to-Braille Pipeline:*

- **OCR (Optical Character Recognition)**: Captures and extracts text from video using PyTesseract.
- **Preprocessing**: Filters out noise and nonsensical text, ensuring clean and accurate
- **Braille translation**: Converts the preprocessed text into Braille, ensuring compatibility with the actuator.

*2) Actuator Pipeline:*

- **GPIO encoding:** The onboard Raspberry Pi GPIO is manipulated through a Python script that maps Braille Unicode onto 3x2 unit matrices.
- **Pin activation:** The grid representation of characters is utilized during a real-time pin activation loop.

## C. *Algorithms*

On program start, the camera is setup with export functions from the Picamera2 library, with a user-defined configuration of 1024x768 pixels. After setup, the camera is run for image capture. This is pushed to a 3-dimensional Numpy array for processing.

This 3D array is fed to the OCR, provided by the pytesseract library, for text extraction. Upon OCR completion, the text is validated for accuracy by referencing the confidence levels provided by the OCR.

A simple ingestion algorithm starts after the OCR extracts the text. This algorithm sanitizes input data for use in character identification. This cleaned text is then encoded into Unicode Braille before being reprocessed into 3x2 grids through a GPIO mapping algorithm.

The code can be found at https://github.com/Pateltirths1012/BrailleSense-rpi/.

## IV. RESULTS

The system was tested with various text formats, including printed documents, signs, and digital screens. Results demonstrated accurate OCR performance with rapid Braille conversion and potential tactile output.

## A. *Performance Metrics*

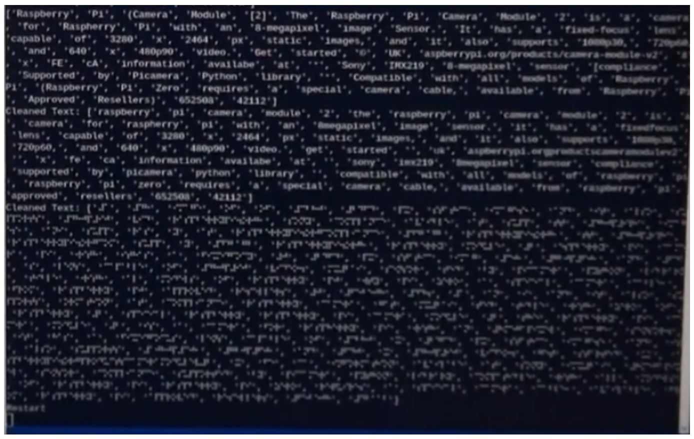
The implemented algorithm accurately extracted $85\%$ of words from standard printed English text on well-lit book paper. This includes both the OCR and post-processing algorithms in sequence. Reducing the size of the image could improve this percentage, as it would require less differentiation and thus less data aberration during extraction.

The response latency of the algorithm from start to finish showed averages of less than two seconds. With the algorithm as it stands transcribing relatively large blocks of text, after actuation sequencing this could result in the targeted rate of 300 words per minute, the expected reading rate of someone without visual impairment. However, those who read Braille have a slightly higher rate of ingestion of text on average.

While the GPIO system was not fully implemented, simulated results of the encoding algorithm presented usable outputs for the actuators. Given a system that is usable with the current circuitry, as well as known voltages and pin mappings, we could expect positive results in actuation.

## B. *Challenges and Solutions*

The Raspberry Pi camera module used for the device provides only low-resolution images, which is both a feature and an obstacle. While the presence of low-resolution images allows a faster ingestion of data, they also have a large amount of noise and aberration. These issues have been eased with preprocessing steps, like binarization and noise removal.



```
['Raspberry', 'Pi', '(Camera', 'Module', '[2]', 'The', 'Raspberry', 'Pi', 'Camera', 'Module', '2', 'is', 'a', 'camera
, 'for', 'Raspherry', 'Pi', 'with', 'an', '8-megapixel', 'image', 'Sensor.', 'It', 'has', 'a', 'fixed-focus', 'lens'
'capable', 'of', '3280', 'x', '2464', 'px', 'static', 'images,', 'and', 'it', 'also', 'supports', '1080p30,', '720p60
', 'and', '640', 'x', '480p90', 'video.', 'Get', 'started', '©', 'UK', 'aspberrypi.org/products/camera-module-v2', [illegible]
, 'x', 'FE', 'cA', 'information', 'availabe', 'at', '', 'Sony', 'IMX219', '8-megapixel', 'sensor', '[compliance', [illegible]
, 'Supported', 'by', 'Picamera', 'Python', 'library', '', 'Compatible', 'with', 'all', 'models', 'of', 'Raspberry'
Pi', '(Raspberry', 'Pi', 'Zero', 'requires', 'a', 'special', 'camera', 'cable,', 'available', 'from', 'Raspberry', [illegible]
', 'Approved', 'Resellers)', '652508', '42112']
Cleaned Text: ['raspberry', 'pi', 'camera', 'module', '2', 'the', 'raspberry', 'pi', 'camera', 'module', '2', 'is', [illegible]
, 'camera', 'for', 'raspherry', 'pi', 'with', 'an', '8megapixel', 'image', 'sensor.', 'it', 'has', 'a', 'fixedfocus'
'lens', 'capable', 'of', '3280', 'x', '2464', 'px', 'static', 'images,', 'and', 'it', 'also', 'supports', '1080p30,
'720p60,', 'and', '640', 'x', '480p90', 'video.', 'get', 'started', '', 'uk', 'aspberrypi.orgproductscameramodulev2'
'', 'x', 'fe', 'ca', 'information', 'availabe', 'at', '', 'sony', 'imx219', '8megapixel', 'sensor', 'compliance', [illegible]
'supported', 'by', 'picamera', 'python', 'library', '', 'compatible', 'with', 'all', 'models', 'of', 'raspberry', [illegible]
, 'raspberry', 'pi', 'zero', 'requires', 'a', 'special', 'camera', 'cable,', 'available', 'from', 'raspberry', 'pi'
'approved', 'resellers', '652508', '42112']
Cleaned Text: [[illegible]]
Restart
```

Fig. 4. *Cleaned and encoded text*

Initially, the camera module also required a manual restart to continue providing images to the algorithm. This was resolved with a real-time cache handler loop that cleared and reset the camera, allowing continuous use until manually shut down instead.

Another challenge is the reliability of the GPIO transmission, mostly the rapidity of output to the actuators. This is informed by the reading rate of the average Braille user, and has been resolved through a time delay loop from the device to the actuators. Going forward, a feedback system from the actuators would be necessary for smoother use.

## V. Conclusion

BrailleSense is a proof of concept for a device that can process real-time, ad hoc information and transcribe it into Braille. The device is a decent baseline for continuing research into real-time Braille transcription using machine learning of adaptive systems, and shows that the approach is effective with room for improvement.

Domains of immediate improvement are simple: the current circuitry (the Raspberry Pi 4) requires a relatively bulky housing, and the actuators are much too large to affix directly to the fingertips. The device would benefit greatly from boutique circuitry, allowing for more organized GPIO setup as well as more optimization on an embedded learning algorithm.

Ultimately, a fully functioning device could have an asymptotic 99% accuracy with an interface to include multiple language lexicons with robust grammatical systems. The device could also be transferred to a lightweight, wearable format akin to insulin pump devices worn by those afflicted with Diabetes.

## Acknowledgment

This project was funded by Northeastern University and it was a final course project for EECE 5552: Assistive Robotics under Dr. Salah Bazzi. This project was a collaborative effort by the BrailleSense team. Each member brought unique expertise to the development and implementation of the system.

**Tirth Patel:** Developed a real-time text-to-Braille conversion system software leveraging a Raspberry Pi 4 and Camera Module 2, integrated with OCR technology using Tesseract and PyTesseract for text extraction. Implemented preprocessing pipelines to clean text and used the PyBraille library to translate it into Braille. A GPIO interface was designed to transmit Braille patterns to external hardware, ensuring precise tactile representation. The system operates in real time, dynamically restarting only upon generating new Braille text, with optimizations for reliability and uninterrupted camera preview.

**Roman Iles:** Produced the problem statement and project outline, including developing a project design pathway and initial algorithm identification. Coordinated general requirements and partitioning of objectives across the team. Produced the GPIO output algorithm for the actuators, and resolved implementation nuances in both the core and output algorithms.

**Maxwell Hasenauer:** Note taking during meetings, found and purchased the braille modules, found materials for the project from Microcenter (did not pick up the materials), tried to investigate for information about the braille modules by tracking down the owners of the IP (hit a dead end): emailed ebay sellers, assistive technology museums, and former IP owning company.

**Venkata Subhash Jahnav Lanka:** Designed the block diagram for the Braille system. Tried investigating about the voltage limitations of electronic components and the connections of the same components.

## References

[1] Y. Chhabra and P. Khanna, *Braille Display for the Visually Impaired using Actuators and Microcontrollers*, 2016.
[2] P. De Silva and C. Fernando, *Real-Time Text to Braille Conversion Using Optical Character Recognition*, 2018.
[3] L. Johnson and R. Smith, *A Portable, Low-Cost Braille Reading Device: Design and Evaluation*, 2020.
[4] L. Meng and T. Zhang, *Braille Interface Devices: Current Trends and Future Directions*, 2021.
[5] P. Tirths *"GitHub - Pateltirths1012/BrailleSense-rpi," GitHub, 2024. https://github.com/Pateltirths1012/BrailleSense-rpi/*