Brailliant

Team A0: Samay Sahoo, Yujun Lee, Ziyu Li

18-500 Capstone Design, Spring 2024
Electrical and Computer Engineering Department
Carnegie Mellon University

Product Pitch

One of the main challenges faced by the visually impaired is the lack and the high cost of braille education tools resulting in their comparatively low literacy rate.

To combat such challenges, Brailliant employs mechanical engineering, software, and embedded systems to provide a refreshable braille display, at one-tenth of the cost, open source and fully DIY-able. Our most crucial requirements include affordability, portability, text-to-braille translation accuracy, and braille display speed.

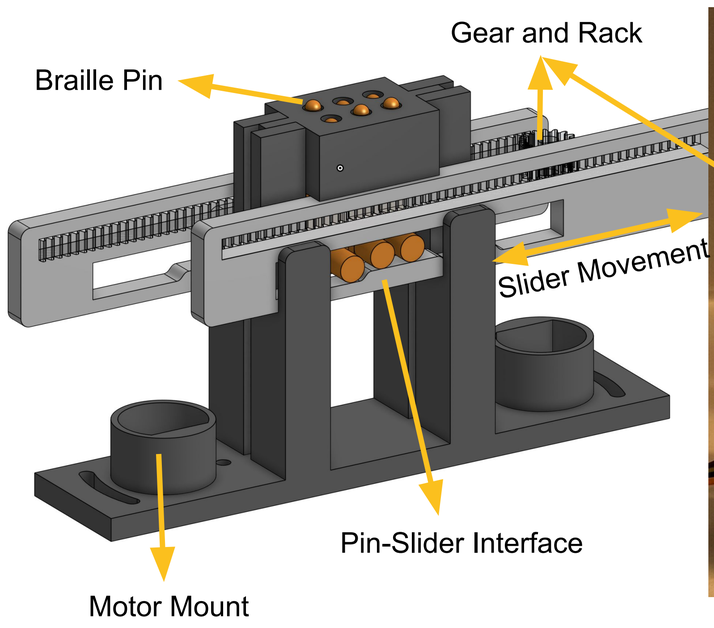
Our final product was able to utilize an innovative sliding actuation technique, minimizing the cost to **under $500, compared to $2000+ current solutions**, and our algorithm was able to reach a **100% braille translation accuracy.** Despite the average display speed of **~1s per a column of a cell**, we were able to detect and resolve motor stalling, providing a reliable solution.

System Description

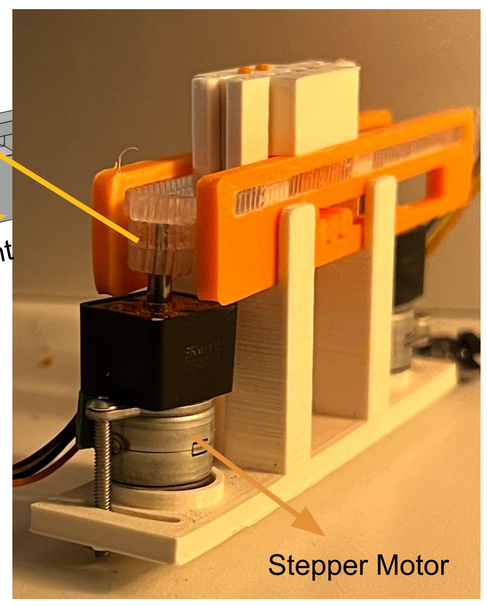
Users may enter any text via our web app, which then runs our Python translation program to convert into Braille character encodings. This data is sent to an Arduino to then actuate each physical braille cell.

Each braille cell is divided into two columns, each with three pins. Their eight unique up-and-down configuration can be achieved through the back-and-forth movement of the slider, which has a special curvature design that can precisely push a different subset of the pins up at different locations. A rack gear is embedded on the side of the slider, which is actuated by a stepper motor mounted on the inside of the two sliders. Gears are manufactured using laser cutter, while the rest of the braille cell is entirely 3D printed.

CAD View of a Single Braille Cell



Assembled Design (Horizontally Expandable)



System Architecture

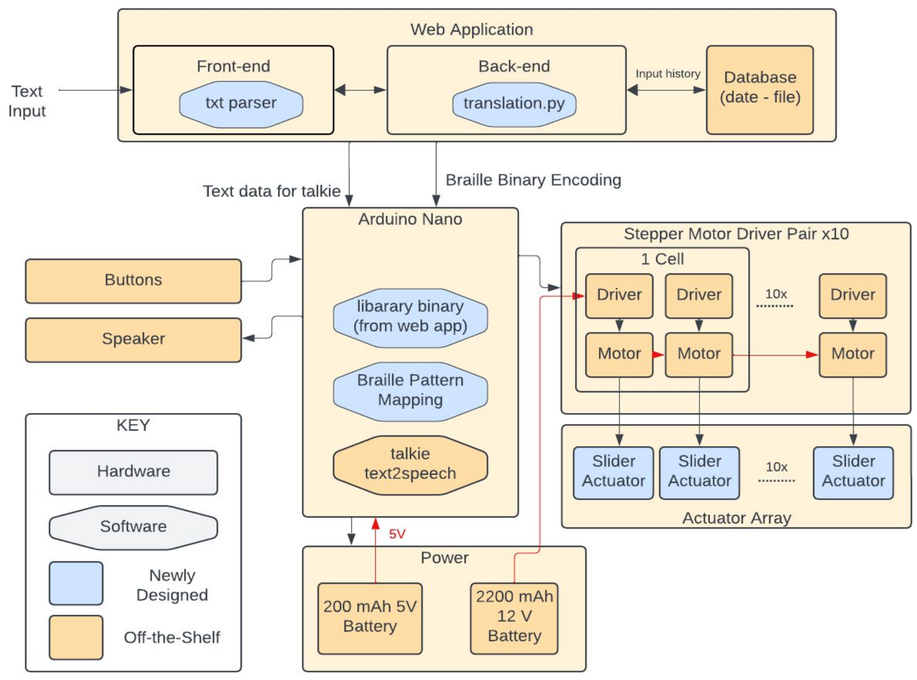


Figure A: Brailliant system architecture design

Software approach

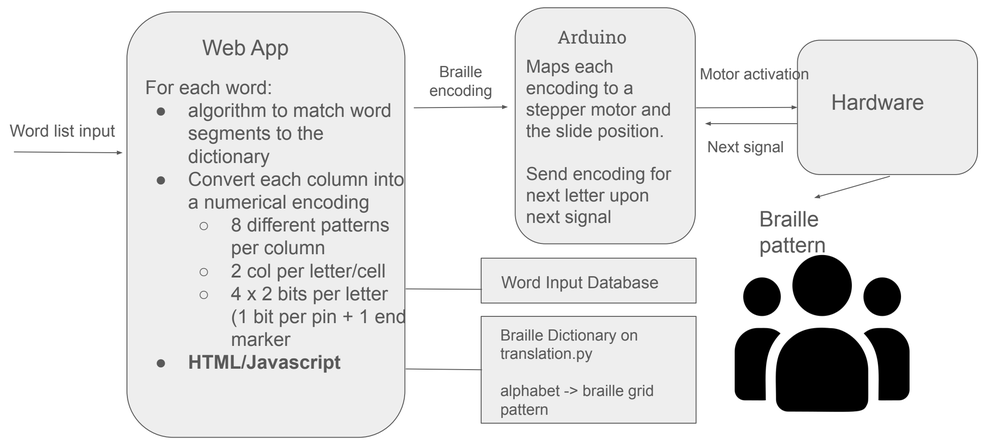
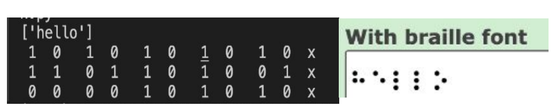


Figure B: Software design architecture

System Evaluation

Test	Testing Method	Requirement	Result
Actuator Response Speed	Record response time per column on 10 words	< 500ms actuation time	~1000ms average actuation time + 3s jamming resolution
Actuator Accuracy	Test grid on 50 braille pattern inputs	> 80% Accuracy	75% Accuracy
Text-to-braille algorithm accuracy	Test on 100 words and compare with online translator	> 100% Accuracy	100% Accuracy
Web-app user testing	User testing on 5 students (feedback on a scale of 0-10)	> 8/10 average for usability/accessibility	8.75/10 rating for usability/accessibility



Braille encoding output Reference solution

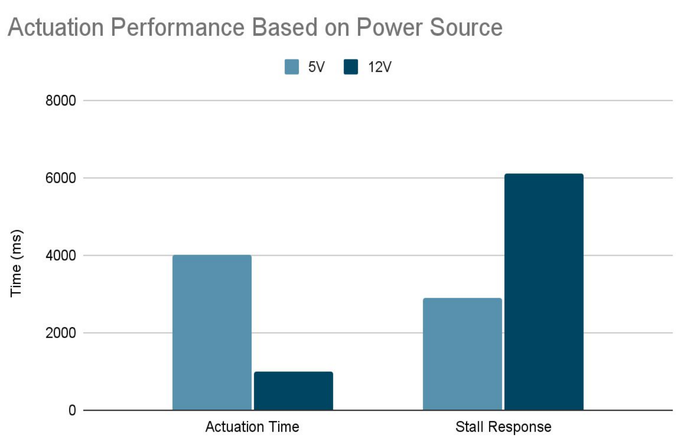


Figure C: Braille actuator performance with varying power source

Website and Conclusions

Our implementation has been successful at dynamically displaying braille, though is not as portable as hoped. Through this process, we've learned that the dependence on mechanical components leads to unexpected physical challenges, such as weight of 3D printed parts impacting required motor torque. The project could be furthered with more sophisticated machining in place of 3D printing to achieve much smaller and compact braille cells. Could then perhaps extend this idea into a large array of cells to display multiple sentences of text at once.