The Sound of Sight

C5: Fiona Fisher, Shravya Sai Koushik, Peter Ragone
18-500 Capstone Design, Fall 2024
Electrical and Computer Engineering Department
Carnegie Mellon University

https://course.ece.cmu.edu/~ece500/projects/f24-teamc5/

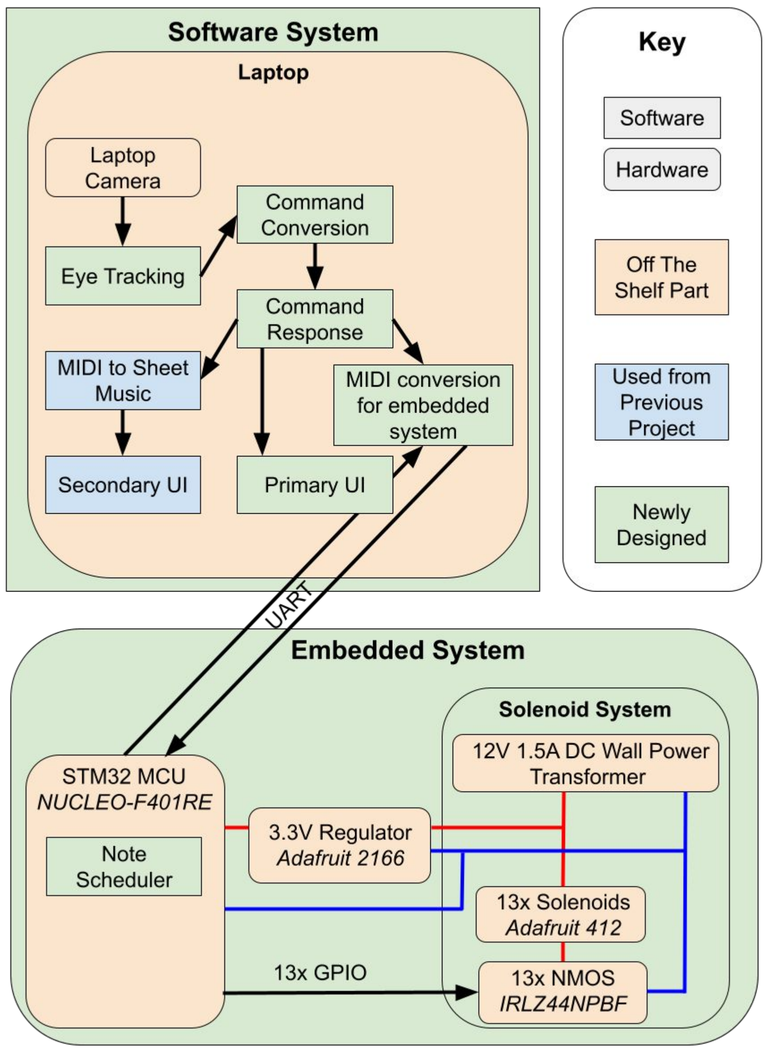
Product Pitch

The goal of our product is make music composition for the piano more accessible by allowing users to compose music with eye-tracking and send it to a hardware system that will play the composition on the piano.

Based on preliminary testing of the software without integration with the hardware, command latency is meeting the use case requirement for “short” commands (**~209ms**) and is not for “long” commands (**~970ms**). Additionally, accuracy also meets our target metric at **~79%**. Based on testing of hardware without integration with software, accuracy of note sequence placed is **96.7%**, which is near-full accuracy except for one inevitable edge case. Latency of solenoid presses will be tested once 3D printed housing is completed.

System Architecture

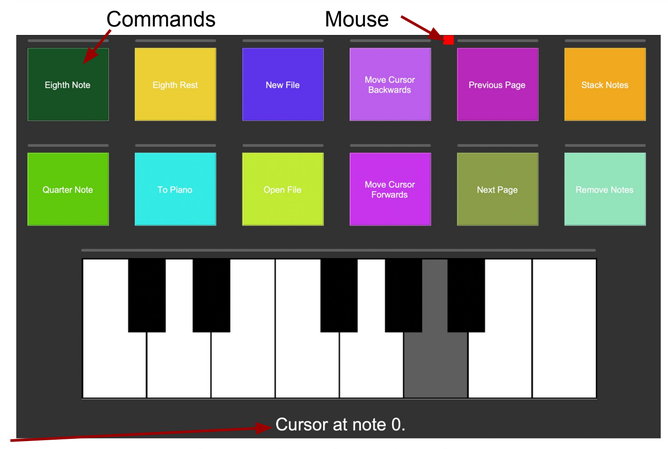
Our system includes a software system and a hardware system working together. The software system will run on the user’s laptop and use eye-tracking to perform commands. The hardware system is an STM32 microcontroller that will communicate with the laptop through UART and command 13 solenoids to press the keys on the piano.



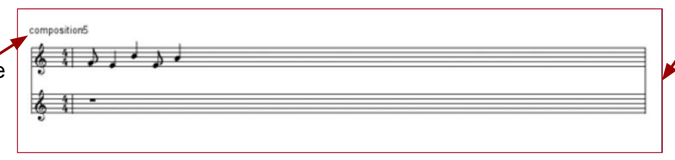
Conclusions & Additional Information

By combining eye-tracking software with solenoid-controlled piano keys, we have created a system that makes playing and composing music accessible for those without traditional hand or arm mobility. While our prototype effectively bridges accessibility gaps, it can be upscaled in the future by incorporating more octaves, allowing various tempos, and implementing note dynamics by controlling solenoid actuation velocity. Integrating machine learning models could personalize the experience by adapting to each users’ eye-tracking patterns.

System Description



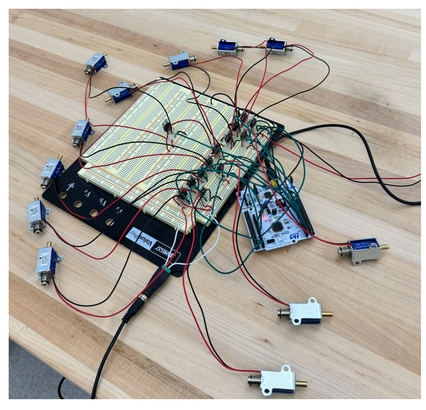
Screenshot of user interface



Screenshot (partial) of secondary user interface

The **User Interface** *samples* the user’s eye-coordinates at a rate designated by the user. With each update, the mouse on the screen will move, and after a number of iterations within a specific command, the command is performed.

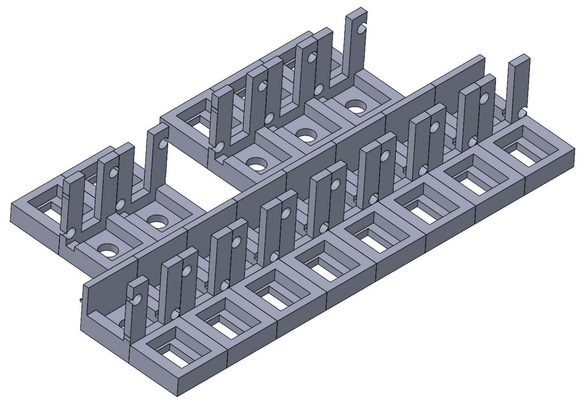
- *Eye-tracking* is performed on the backend with OpenCV and the MediaPipe library.
- The UI is designed in Python with the *Tkinter* library.
- Errors/edge cases are handled within the program for hands-free use.
- Real-time sheet music is generated by an open-source program and can be displayed on a second monitor.
- Internal composition is stored as a *MIDI file* and generated/parsed with the Python MIDO library.



Solenoid control circuitry on breadboard

Note scheduling and hardware execution:

- We use Python's Mido library to process MIDI files, extracting essential note scheduling and pitch data while filtering out irrelevant details.
- This parsed data is sent to the STM32 via UART, enabling precise solenoid control through GPIO signals to execute the note schedule.



3D printed housing for the solenoids, weights, and circuit board.

- Rectangular slots hold the weights to aid the solenoids in pressing the keys.
- Tall brackets interface with the solenoids allowing for them to be screwed down to restrict movement of the solenoids’ body as it actuates.

System Evaluation

Total Budget: ~$182

Preliminary results of testing on *quantitative* requirements (may change with more tests).

Requirement Type	Metric	Target	Actual
Use Case	Command *Latency*	<500ms	**~209ms** for “short” commands **~970ms** for “long” commands
Design	UI Update *Latency*	<100ms	**~777ms**
Use Case	Software *Accuracy*	>75%	**~79%**
Design	Software *Accuracy*	>95%	**Testing In Progress**
Use Case	Hardware *Accuracy*	100%	**96.7%**
Use Case	Hardware *Latency*	<10%	**Testing in Progress**
Design	Solenoid Power Consumption	<9W	**8.73W at maximum**

Design trade-offs

Design choice	Reasoning for trade-off made
Minimum note length	8th note (250 ms); physically limited by solenoid actuation speed (<100 ms presses = unreliable)
Limited to 1 octave, fixed 120 BPM, no note dynamics	Out of scope (complexity, timeline, budget)
Microcontroller selection (STM32 > Rpi)	Better library support for UART, peripherals operate independently of CPU
Solenoid force	Applied weights as part of solenoid housing instead of purchasing more forceful solenoids (expensive)