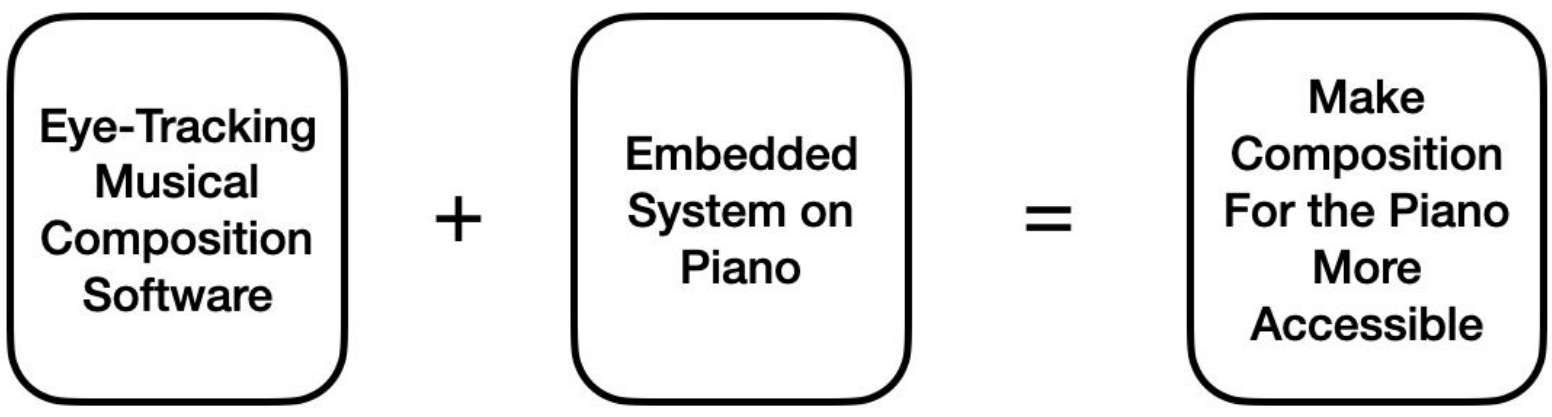
Team C5: The Sound of Sight

Project Overview

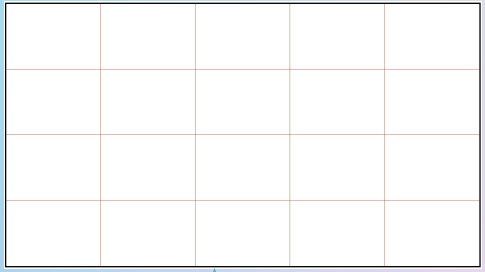


Shravya Sai Koushik, Fiona Fisher, Peter Ragone

Use Case Requirements

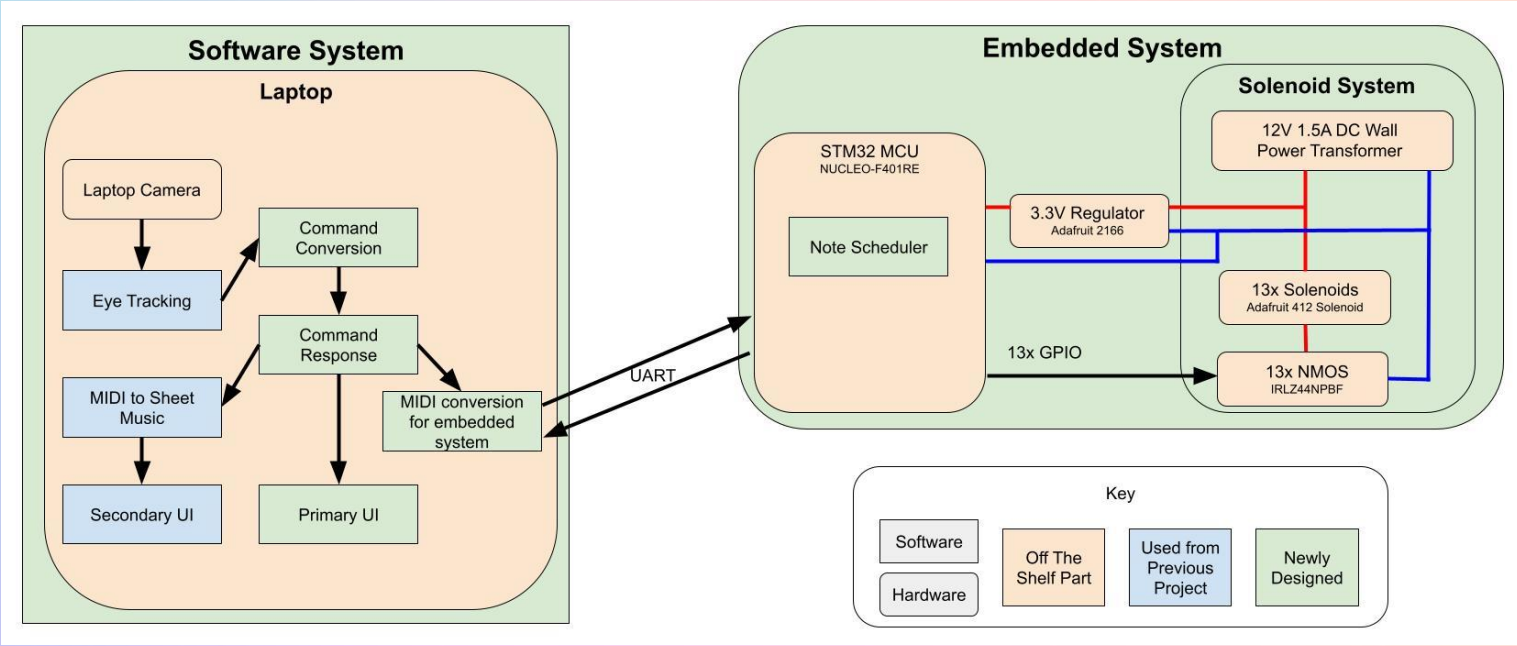
Type	Requirement	Metric
Accuracy	Software; identifying eye-commands made by the user.	>75%
	Hardware; producing the correct *sequence* of notes on the piano.	100%
Latency	Software; perform required command & be ready to recognize the next.	<500ms
	Software; secondary UI to update after user confirms command.	<100ms
	Hardware; solenoids actuate within (+/-) a certain % of expected duration on each note.	<10%
Coverage	Software & Hardware; one octave of notes & chords <4 notes.	N/A
Accessibility	Software & Hardware; can be used hands-free after being set-up.	N/A
Installation	Hardware; non-destructive & de-installation/re-installation possible.	N/A

Design Requirements



Type	Requirement	Metric
Accuracy	Software; identifying eye-commands made by the user (within a target region.)	>95%
Accessibility	Software; local application, uses computer camera instead of (expensive) eye-tracking camera.	N/A
	Hardware; solenoid system uses wall power.	N/A
Power Consumption	Hardware; power consumption of solenoids (in total).	<9W

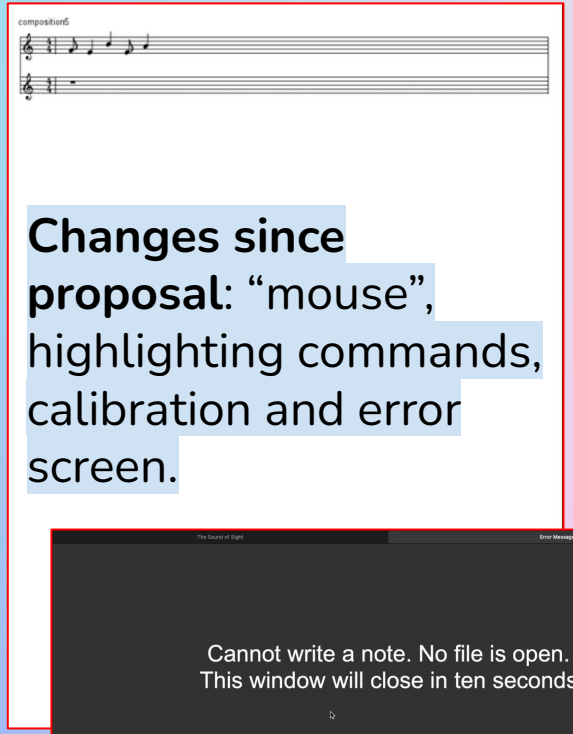
Solution Approach



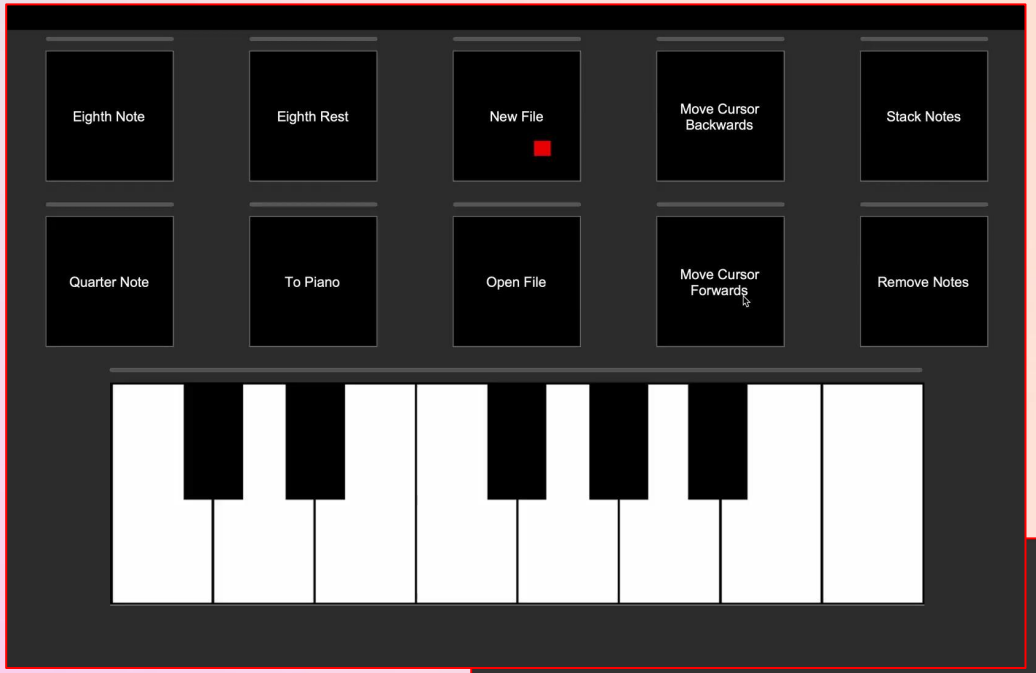
Changes since proposal: some changes to the UI & interface between the secondary and primary UI.

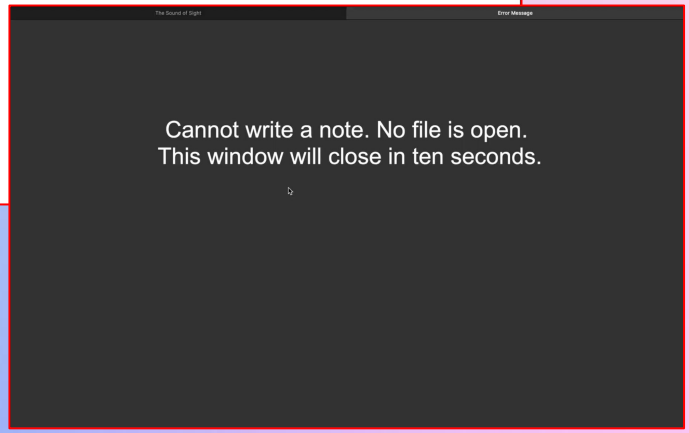
Considerations: music is linked to mental *health* & is important is many *social* and *cultural* traditions.

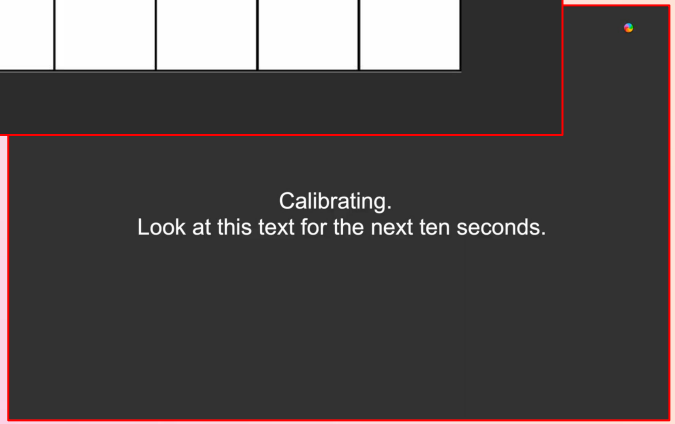
Complete Solution - Software



Changes since proposal: “mouse”, highlighting commands, calibration and error screen.







Complete Solution - Hardware

pythonmidi.py

Get Started | pythonmidi.py 2 | firmware.c

Users > shravyaks > Documents > pythonmidi.py > ...

```
119     for event in events:
120         if event['type'] == 'note':
121             formatted_data += f"Note: {event['note']}, Duration: {event['duration']} ticks
122         elif event['type'] == 'chord':
123             formatted_data += f"Chord: {event['notes']}, Duration: {event['duration']} tic
124         elif event['type'] == 'rest':
125             formatted_data += f"Rest: {event['duration']} ticks\n"
126     return formatted_data
127
128
129 if __name__ == "__main__":
130     midi_file_path = "/Users/shravyaks/Documents/composition4.mid"
131     events = parse_midi_file(midi_file_path)
132     print("Parsed events:", events)
133
134     simulate_note_playback(events)
135
136     # Format data as it would be for firmware transmission
137     firmware_data = format_notes_for_firmware(events)
138     print("\nFormatted data for firmware transmission:\n", firmware_data)
```

PROBLEMS 2 | OUTPUT | DEBUG CONSOLE | TERMINAL

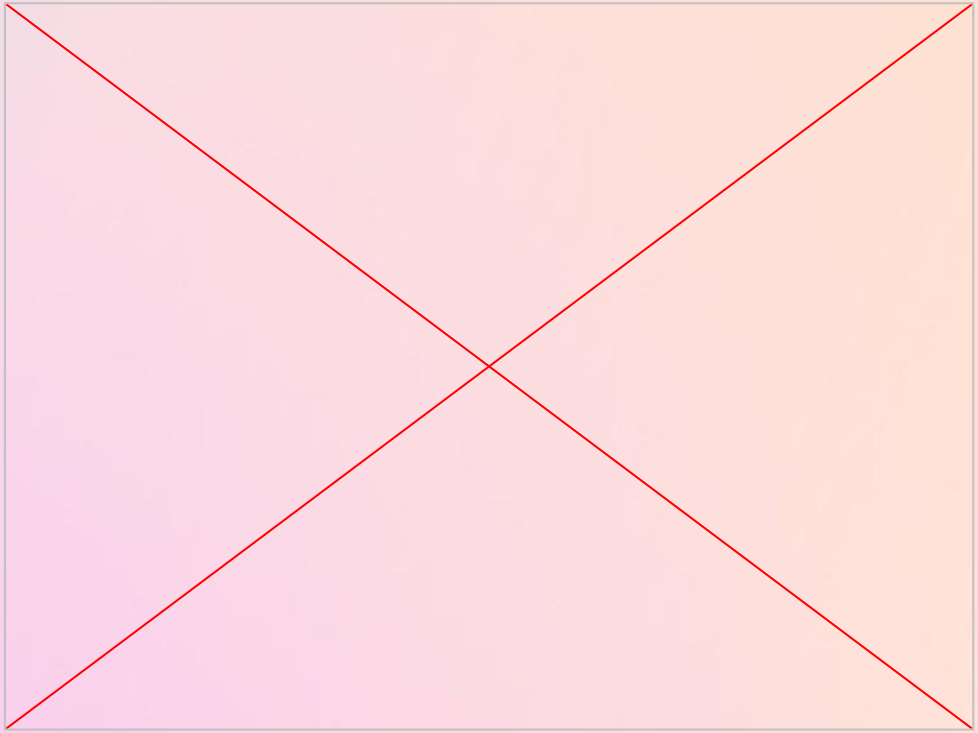
```
Playing chord: [72, 62] for 480 ticks
Playback complete.

Formatted data for firmware transmission:
 Chord: [72, 71], Duration: 480 ticks
Chord: [72, 69], Duration: 480 ticks
Chord: [72, 67], Duration: 480 ticks
Chord: [72, 65], Duration: 480 ticks
Chord: [72, 64], Duration: 480 ticks
Chord: [72, 62], Duration: 480 ticks

(myenv) (base) Shravyas-Air:Documents shravyaks$
(myenv) (base) Shravyas-Air:Documents shravyaks$

Session contents restored from 29/11/2024 at 14:03:58


The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Shravyas-Air:Documents shravyaks$
```



Formal Testing Plans: UI & Eye-Tracking

$$f(c) = 1 \text{ if accurate, } 0 \text{ if not}$$

$$accuracy = \frac{\sum_{i=0}^{x} f(c_i)}{x}$$

$$avg\ latency = \frac{\sum_{i=0}^{x} (R_i - V_i)}{x}$$

Initial Results: UI & Eye-Tracking

- Eye-Command Accuracy - Tested 2.5 feet away from screen
  - Configuration 1 - configured for eye-tracking with head movement
    - 100% with head movement
    - 0% with eyes only
  - Configuration 2 - configured for eyes only
    - 16/16 correct command presses
    - 26/29 correct key presses
    - 3/29 incorrect key presses
    - 89.6% key presses eyes only
    - However, requires keeping head impractically still

Hardware: Testing so far + Remaining Testing Plans

- Plan for Solenoid Functionality:
  - Feed in MIDI files and observe solenoids to ensure mechanical actuation works as expected.
  - Expect 100% accuracy on sequence, and <10% latency on note press durations (can use metronome to cross-verify)
- Parsing Correctness:
  - Used pre-recorded MIDI files with various complexity (various note lengths, use of chords) to ensure parsed notes match our input 100%.
- Power consumption testing results

Scenario	Current (A)	Voltage (V)	Power (W)
Idle	0.15	11.8	1.77
Single solenoid actuates	0.25	11.8	2.95
Chord actuation (3 solenoids)	0.74	11.8	8.73

Design Trade-offs

- Note lengths we can accomodate are physically limited by solenoids' actuation speeds
  - Solenoids have trouble on note presses shorter than 100ms (a 32nd note)
  - Shortest note we will allow is an 8th note (250 ms)
- We will not be implementing note dynamics (varying solenoid velocity) or varying tempo of piece.
  - It is complex enough to figure out pitch and note length.
- Doesn't work on compositions from the internet - logic is specific to our design.

Lessons Learned

- Breaking the system into smaller pieces was vital.
- Open source tools can have niche issues and minimal documentation.
  - Nuisance when generating/parsing MIDI files: MIDO uses relative, not absolute time.
  - Our external sheet music generation tool is 12 years old w/ little documentation.
- Integration took longer than expected.
- Testing plans were very reliant on integration.
- Communication is integral - especially when writing software that needs to integrate.

Final Weeks

Peter	Fiona	Shravya
Optimize eye-tracking.	Fix remaining bugs in UI.	Debug UART.
Finish 3D printed case.	Add missing UI functionality.	Formal hardware testing.
	Finish integrating UART connection to UI.	
Formal software testing.		
Full system integration test.		

Main change: planned to have testing done *before* the interim demo.