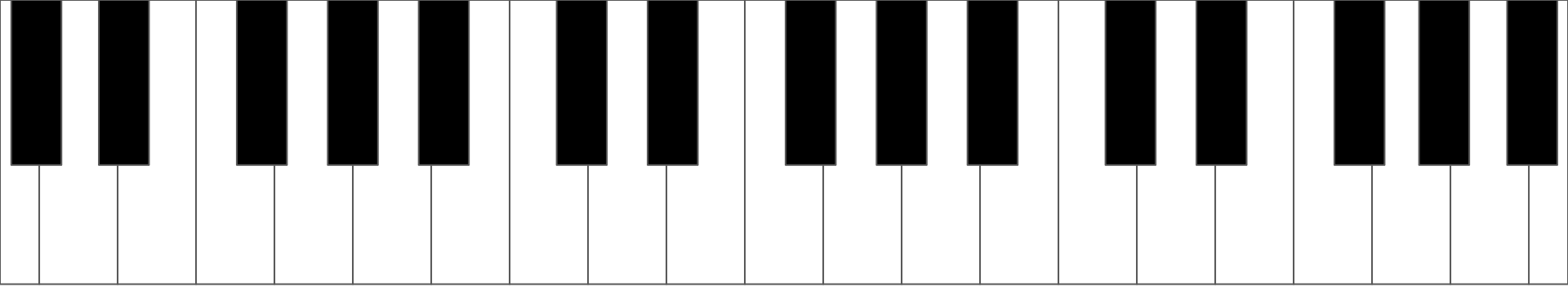


accompanyBot

Aden Fiol, Rahul Khandelwal, Nora Wan

Use Case

Finding professional piano accompanists can be difficult and costly, while digital recordings lack the acoustics and sound quality of a physical piano.

Our solution: accompanyBot can read sheet music and reproduce the notes by pressing keys on a piano so that you can sing/play along!

ECE Areas:

- Software: Interface application, machine learning model
- Hardware: Microcontroller integration with actuators

Use Case Requirements

Requirement	Metric
Accurate sheet music reading	> 95% accuracy of note pitches and note values
Tempo variability	Ability to speed up/slow down playback tempo with exact BPM accuracy
Low latency between UI and piano player	Piano player should start/stop playing within 150 milliseconds of pressing play/pause
Reasonable frequency of key presses	Max key press frequency of 6 notes per second[1]
Reasonable power consumption	< 60W average power

[1] “Fastest piano key hitting - Guinness World Records.” YouTube, uploaded by Guinness World Records, 6 June 2017, https://www.youtube.com/watch?v=vLgtqDkapQU.

Solution Approach (Software)

Music Sheet Parser

- Optical Music Recognition software to parse jpeg music scores
- Converting parsed music to desired file format (XML, MIDI, etc.)

User Interface

- Interface that allows users to upload sheets of music constrained to a certain set of notes
- Allows for pausing and resuming of the accompanyBot

Solution Approach (Hardware)

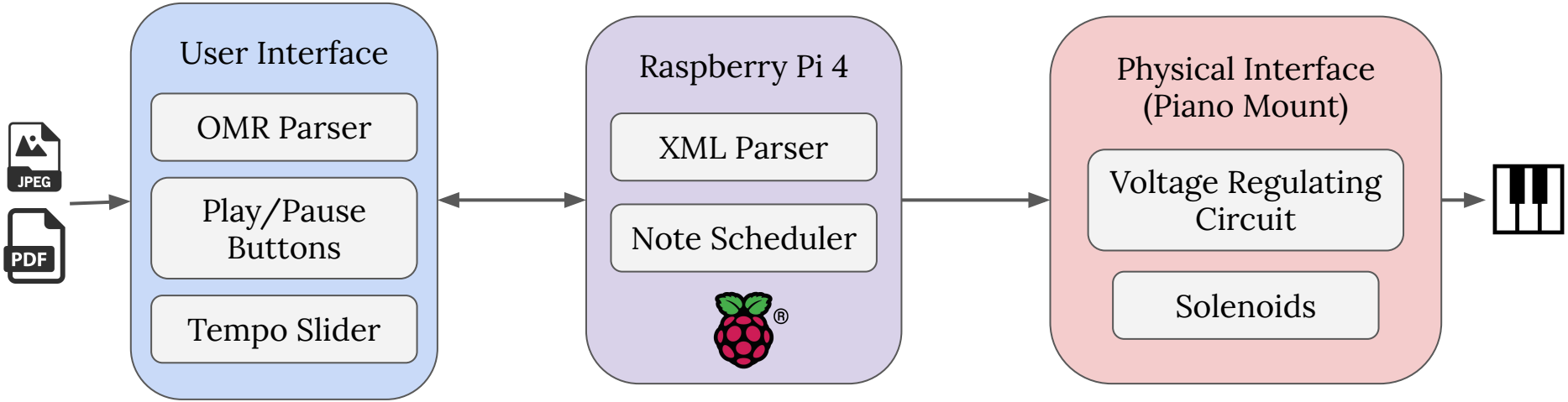
Note Scheduling

- Microcontroller takes in the XML file and parses for notes and tempo.
- Translates notes into correctly-timed signals that are sent to the piano player

Physical Interface

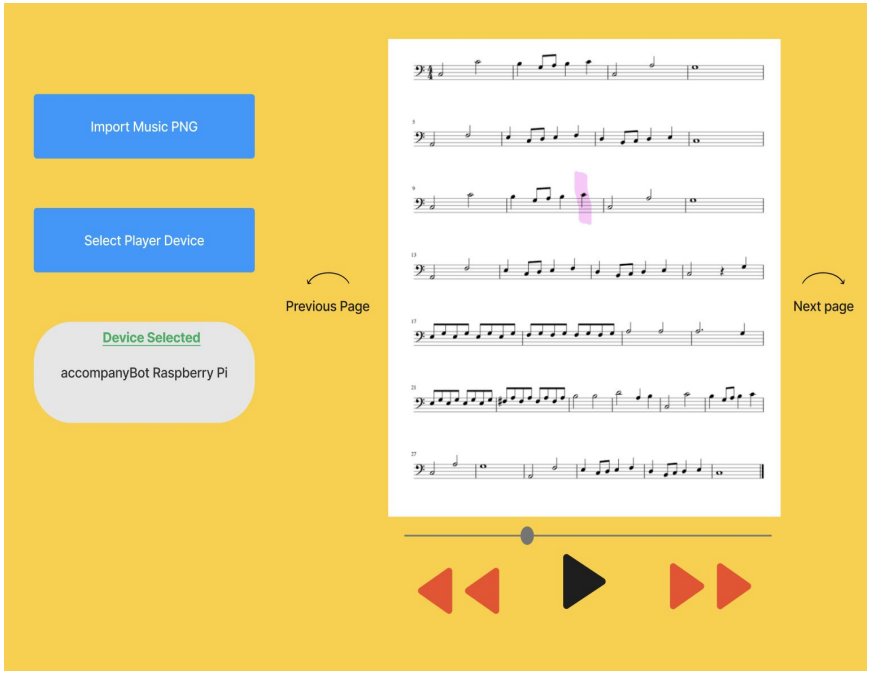
- Solenoids controlled by transistors/relays used to depress piano keys
- Device will have an adjustable mount that interfaces with the piano allowing the user to move it to the desired starting position

Block Diagram

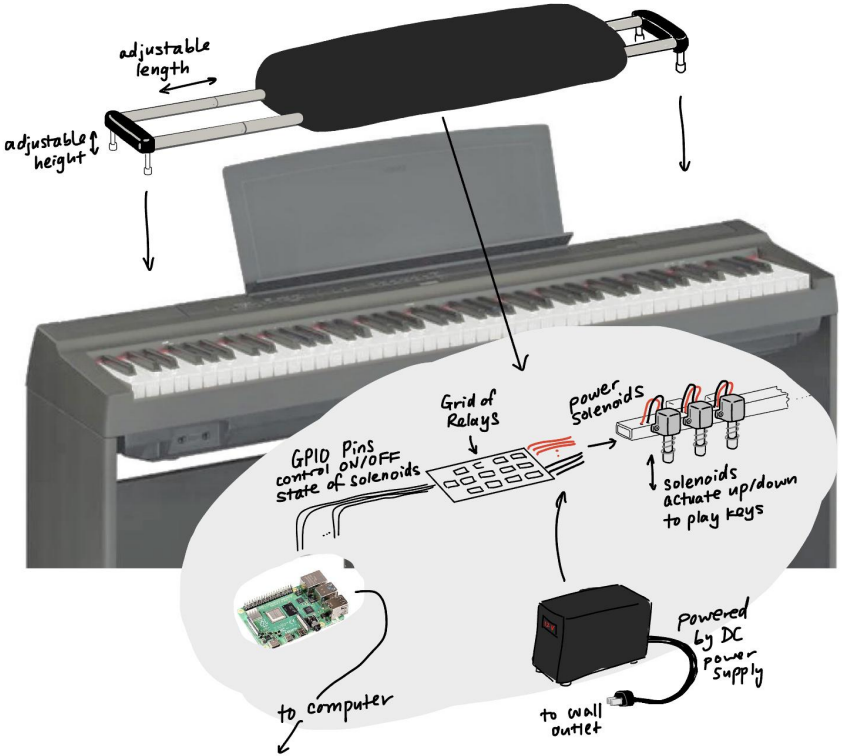


Design Mockups

Hub Application UI



Physical Implementation Model



Technical Challenges

- Guaranteeing accuracy from sheet music parser
- Integration between domains (Hardware/Software)
- Pausing and resuming our accompanyBot seamlessly
- Realizing precision of timing note presses in our note scheduler
- Powering our worst case number of solenoids at once
- Physical circuitry and control of the solenoids

Testing, Verification, and Metrics

Requirement	Testing
> 95% accuracy of note pitches and note values	Visually compare the original sheet music to the XML generated by parser
Exact BPM accuracy in tempo	Measure BPM with a metronome and compare with the tempo from sheet music
Piano player should start/stop playing within 0.5 seconds of pressing play/pause	Test the delay between pressing play/pause and the start/stop of the player with a timer
No errors played by accompanyBot	Generate MIDI file alongside the XML file and compare playback of MIDI file to the sound of the notes played
< 60W Average power	Measure voltage and current over time with oscilloscopes to calculate average power

Tasks and Division of Labor

Rahul

- Sheet music recognition through OMR
- Designing user interface to handle user inputs

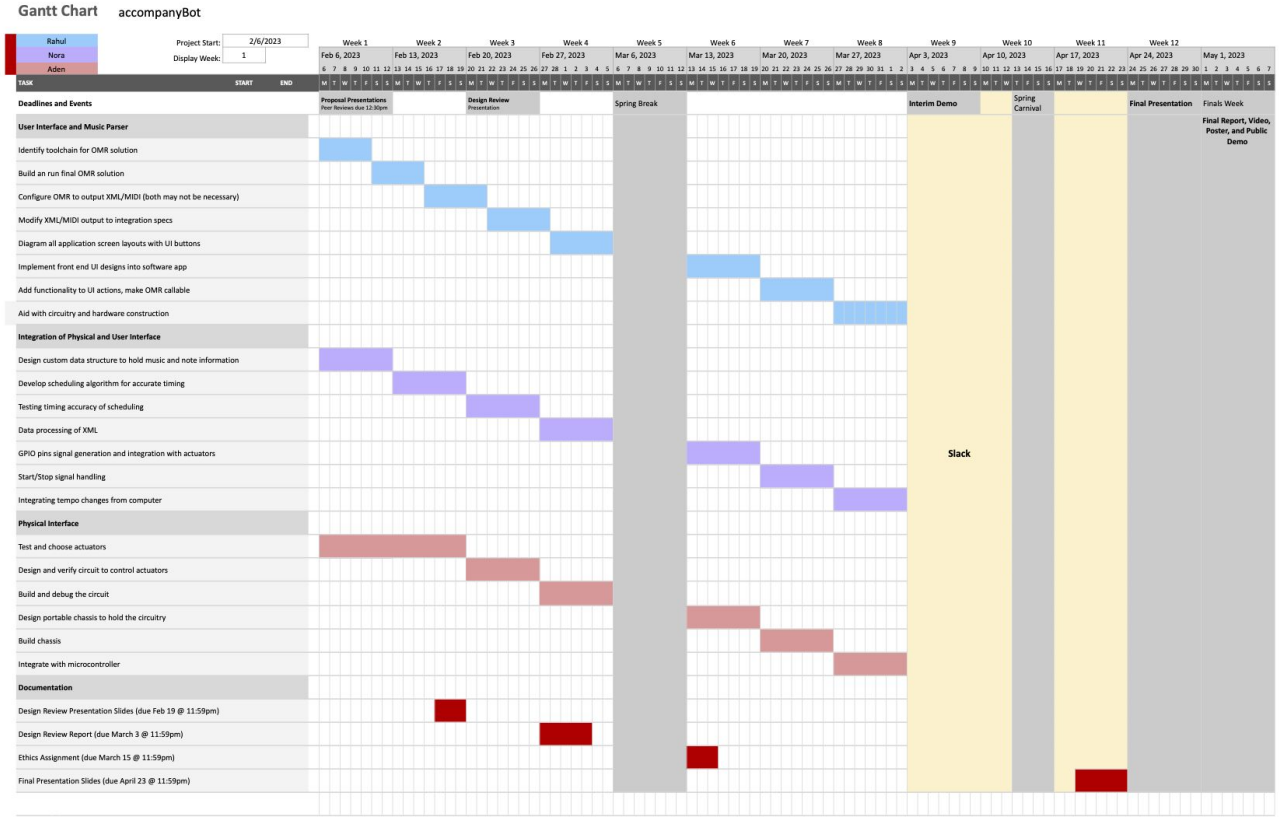
Aden

- Testing and designing the circuit to power the actuators
- Building chassis for holding up actuators

Nora

- Communication between computer and microcontroller to change the tempo and start/stop playback
- Note scheduling on the microcontroller to control actuators

Schedule


Gantt Chart
accompanyBot
Rahul
Nora
Aden
Project Start: 2/6/2023
Display Week: 1
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Week 11
Week 12
Feb 6, 2023
Feb 13, 2023
Feb 20, 2023
Feb 27, 2023
Mar 6, 2023
Mar 13, 2023
Mar 20, 2023
Mar 27, 2023
Apr 3, 2023
Apr 10, 2023
Apr 17, 2023
Apr 24, 2023
May 1, 2023
TASK
START
END
Deadlines and Events
Proposal Presentations
Peer Reviews due 12:30pm
Design Review
Presentation
Spring Break
Interim Demo
Spring
Carnival
Final Presentation
Finals Week
User Interface and Music Parser
Final Report, Video, Poster, and Public Demo
Identify toolchain for OMR solution
Build an run final OMR solution
Configure OMR to output XML/MIDI (both may not be necessary)
Modify XML/MIDI output to integration specs
Diagram all application screen layouts with UI buttons
Implement front end UI designs into software app
Add functionality to UI actions, make OMR callable
Aid with circuitry and hardware construction
Integration of Physical and User Interface
Design custom data structure to hold music and note information
Develop scheduling algorithm for accurate timing
Testing timing accuracy of scheduling
Data processing of XML
GPIO pins signal generation and integration with actuators
Slack
Start/Stop signal handling
Integrating tempo changes from computer
Physical Interface
Test and choose actuators
Design and verify circuit to control actuators
Build and debug the circuit
Design portable chassis to hold the circuitry
Build chassis
Integrate with microcontroller
Documentation
Design Review Presentation Slides (due Feb 19 @ 11:59pm)
Design Review Report (due March 3 @ 11:59pm)
Ethics Assignment (due March 15 @ 11:59pm)
Final Presentation Slides (due April 23 @ 11:59pm)

Conclusion

MVP:

- A fully functional user interface
- A working pipeline from music parsing to actuator movements
- Proper scheduling from the microcontroller
- Portable structure that holds the actuators and circuitry
    - Can be placed anywhere along an octave range on the keyboard