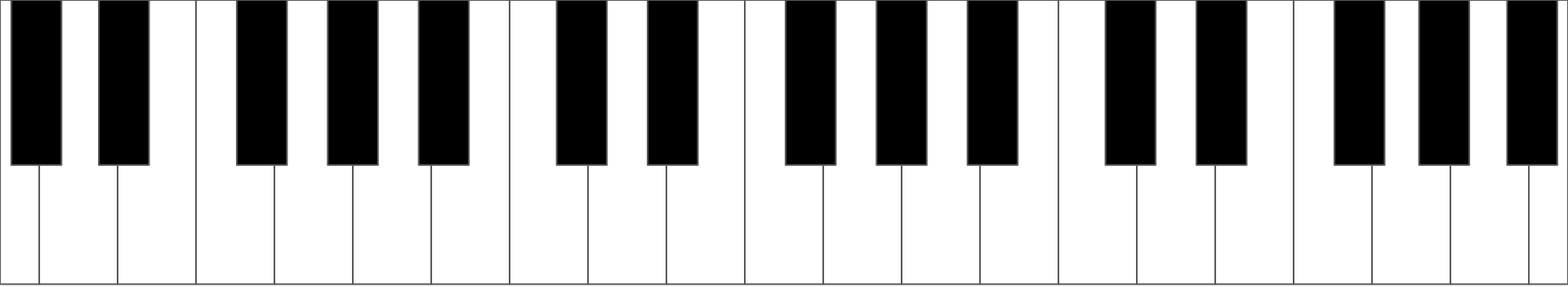


ac·com·pa·ny·Bot

/əˈkəmp(ə)nē bät/ *noun*

A piano playing robot that scans and parses sheet music and reproduces the notes by pressing keys on a piano

Made by Aden Fiol, Rahul Khandelwal, and Nora Wan

Use-Case Requirements

- **Parsing sheet music to XML accuracy:** > 95% accuracy of note pitches and note values
- **Tempo Limit:** Limit tempo and smallest note values such that the time between any two notes is greater than 100 ms
- **Tempo variability:** Ability to speed up/slow down playback tempo with exact BPM accuracy
- **Low latency between UI and piano player:** start/stop playing within 150 ms of pressing the start/stop button
- **Key press frequency:** Limited to 6 key presses per second
- **Reasonable power consumption:** < 60W average power

Solution Approach

- Local application on user's computer handles file input and UI to play/pause device
- Raspberry Pi embedded in the device accepts data through serial communication from computer and parses text to scheduled times for when to turn mapped GPIO pins high or low (for pressed or not pressed keys)

Recent Discoveries:

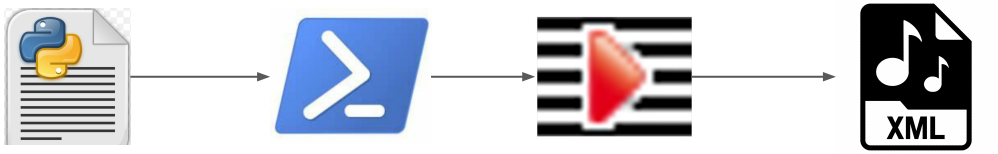
- MusicXML is readable yet lengthy and complex
- Further Conversion through music21 for python (handy for note scheduling)

Safety considerations:

- Solenoids accelerate rather quickly, without dampeners can be hazardous and noisy
- Keep open wires enclosed

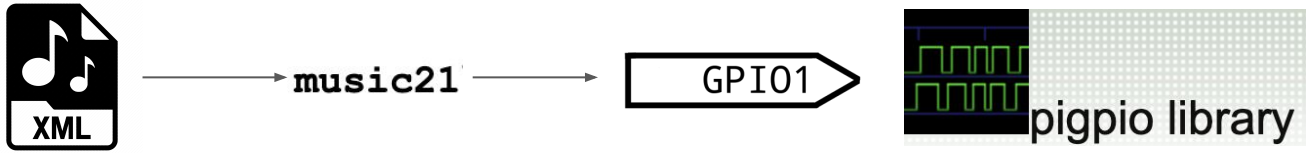
System Specification

Translation Path:

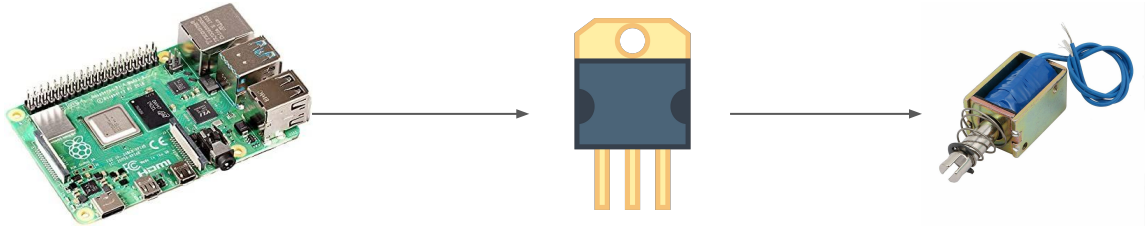


Audiveris OMR

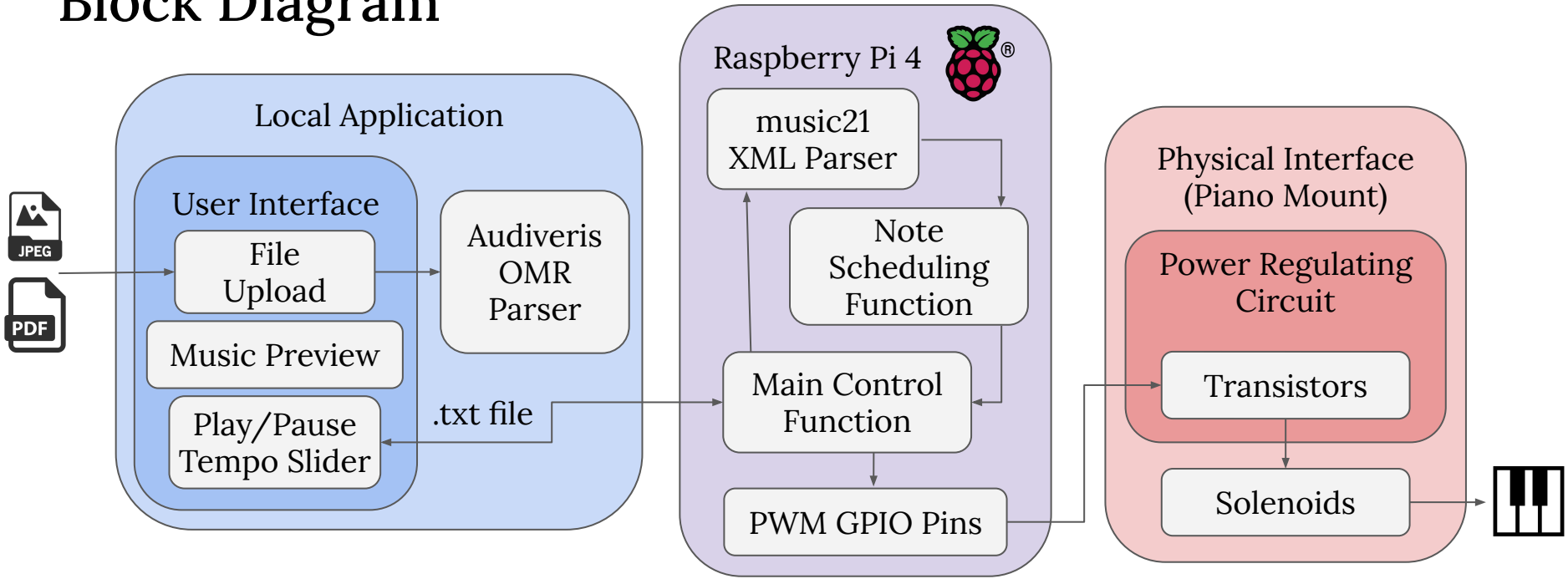
Scheduling Path:



Execution Path:

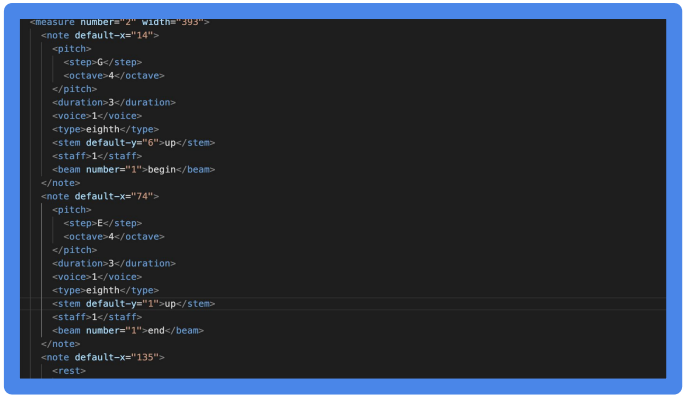


Block Diagram



Implementation Plan (Software)

- Build a Python application for the user interface
- Optical Music Recognition through Audiveris (open source OMR engine) that converts sheet music into a MusicXML file
- Data transmission and file coordination with Raspberry Pi.
- Introduce communication signals for commanding the execution of the Raspberry Pi.



Implementation Plan (Hardware)

Raspberry Pi 4 processes MusicXML data and schedules using steps below:

- music21 parse function converts XML into a Stream object
- Iterate through the list of notes and set bit masks that correspond to which GPIO pins are high or low at each time instance

pigpio library: can read and write to a bank of GPIO pins simultaneously and has hardware timed PWM on all GPIO pins

Solenoids assembled on a chassis and hooked up with transistors to a shared power supply. Gates of transistors are attached to GPIO pin outputs from RPi

Test, Verification and Validation

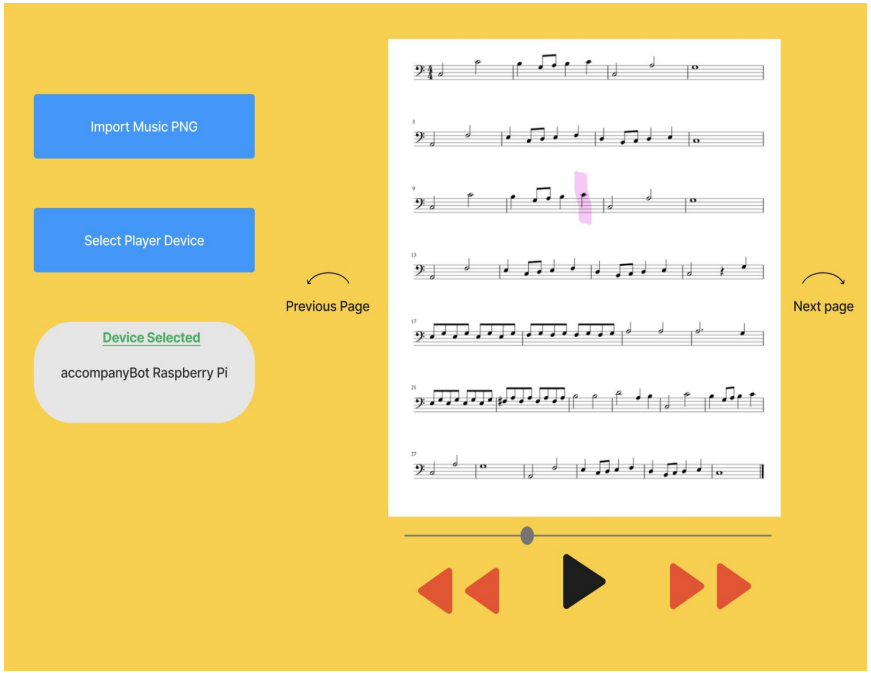
- **OMR Accuracy** → Reconvert back to pdf and compare visually
- **Note pressing accuracy** → Compare to music21 MIDI conversion
- **Other tests** → Direct timing and measurement of process

Setting Preconditions:

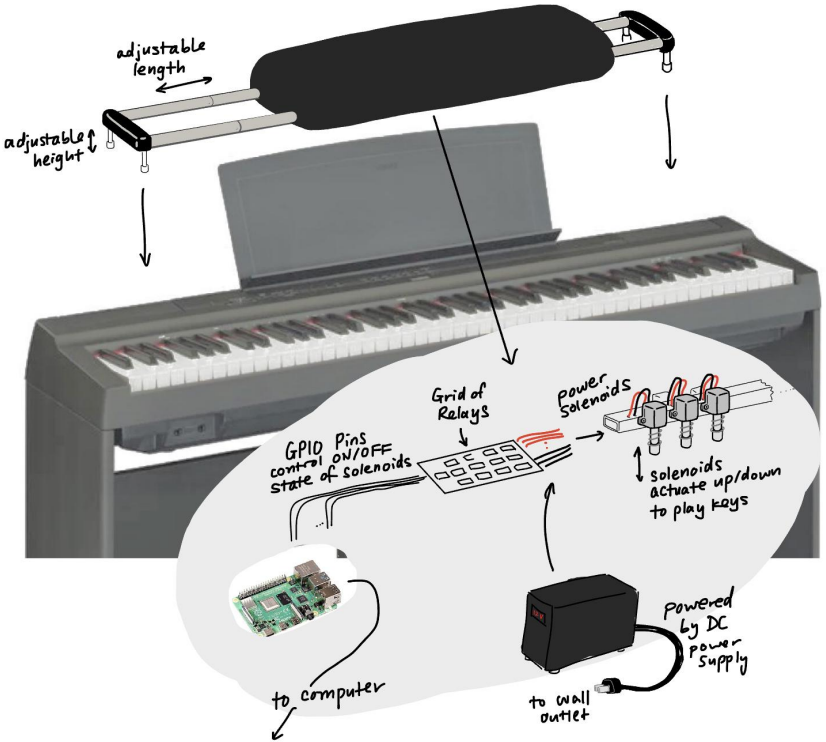
- **OMR -** Keep pdf scores as black/white files (minimal grayscale variation)
- **Note Presser** - Calculate range of notes needed, adjust start position accordingly
- **Tempo Limiter** - Physical solenoids should not be instructed to play faster than movement threshold

Design Mockups

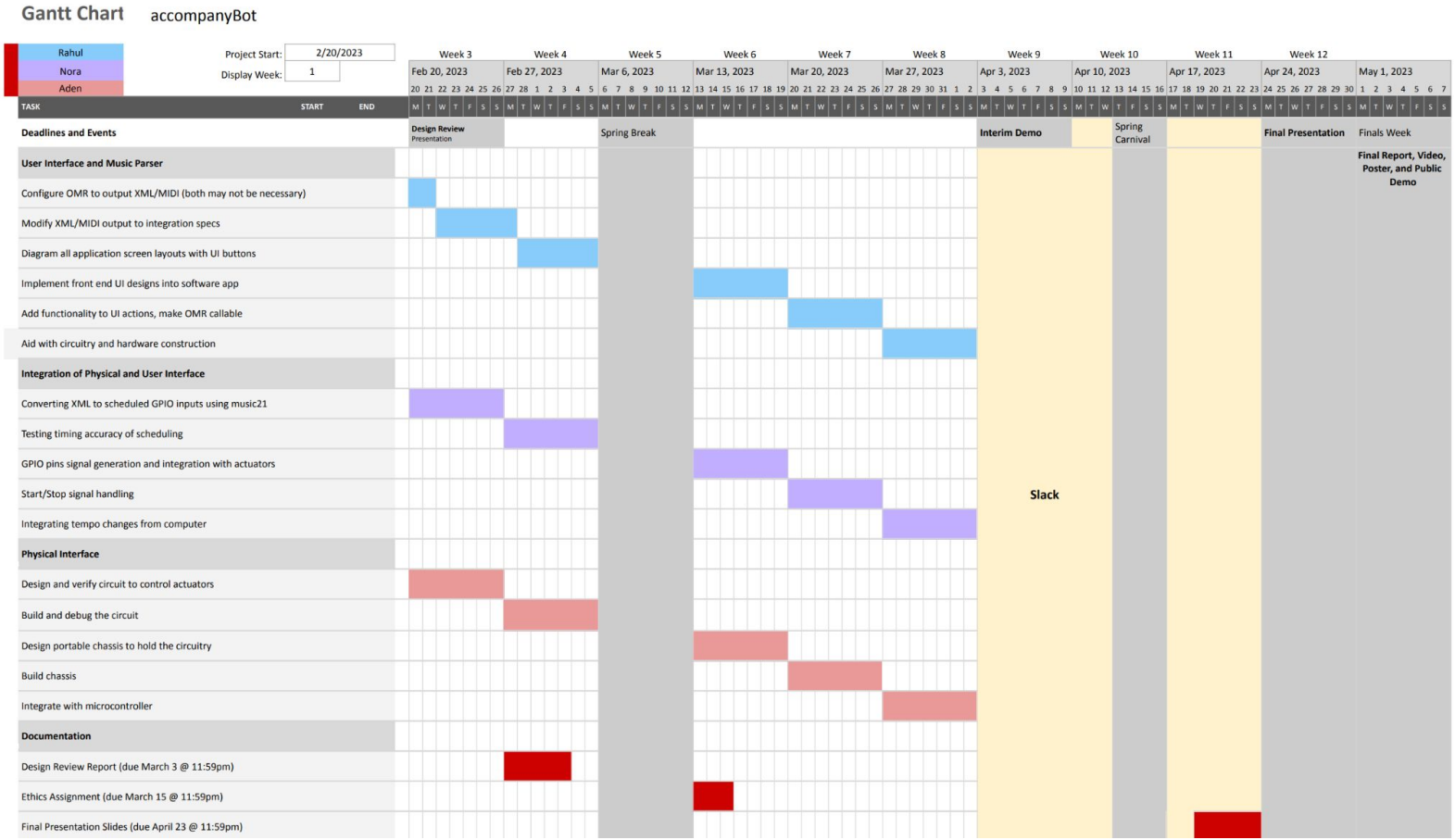
Hub Application UI



Physical Implementation Model



Schedule


Gantt Chart
accompanyBot
Rahul
Nora
Aden
Project Start: 2/20/2023
Display Week: 1
TASK
START
END
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Week 11
Week 12
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
Deadlines and Events
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
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
Converting XML to scheduled GPIO inputs using music21
Testing timing accuracy of scheduling
GPIO pins signal generation and integration with actuators
Start/Stop signal handling
Integrating tempo changes from computer
Slack
Physical Interface
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 Report (due March 3 @ 11:59pm)
Ethics Assignment (due March 15 @ 11:59pm)
Final Presentation Slides (due April 23 @ 11:59pm)