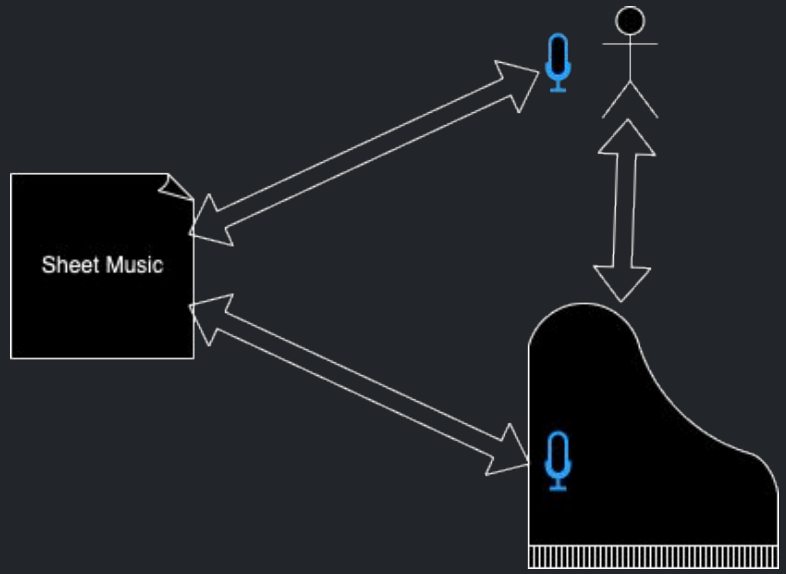

InSync
Aakash Sell, Mathias Thomas, Ben Martinez

Use Case

- Rehearsal tool to improve timing between singer with pianist (or accompaniment)
  - New Musicians
  - Vocalist + Pianist pairs
- Most existing solutions can only measure performance of one musician
- ECE Areas:
  - Signals & Systems
  - Hardware Systems
  - Software Systems

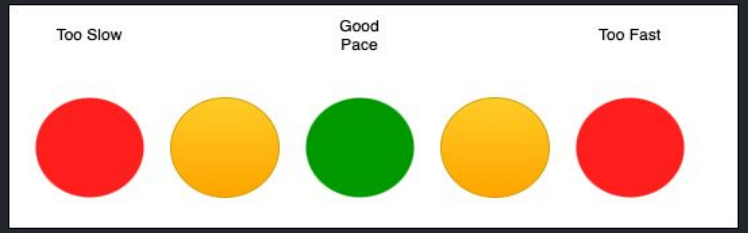


Requirement #1: Real Time feedback

Motivation:

System needs to provide feedback within a timeframe that allows musicians to adjust

- ➔ Sub Requirements:
- ➔ Provide feedback within 30 millisecond latency
- ➔ Find position within piece with an accuracy of ± 15 milliseconds
- ➔ Feedback latency synchronized within ± 15 milliseconds



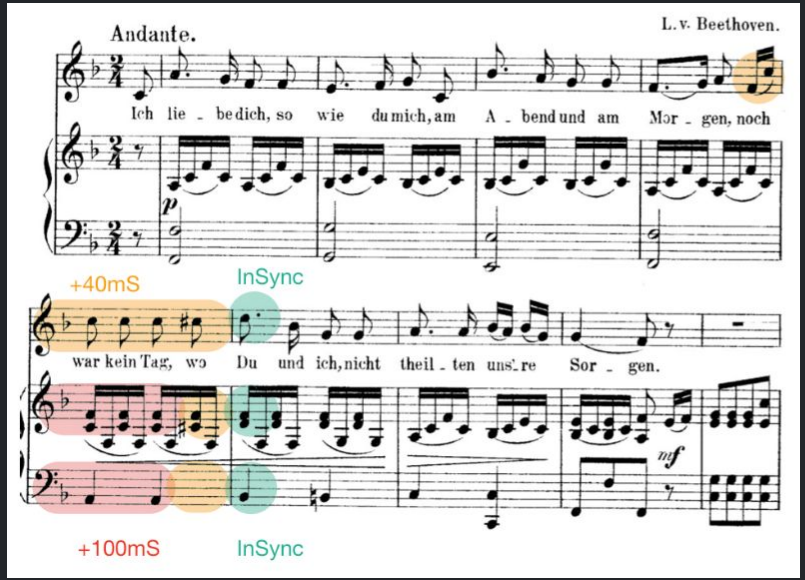
Requirement #2: Post Performance Feedback

Motivation:

Highlight trouble areas; show improvement over time

Sub Requirements:

- ➔ Highlight areas where musicians are out of sync with accuracy within ± 15 milliseconds
- ➔ Feedback should be available within 30 seconds after recording



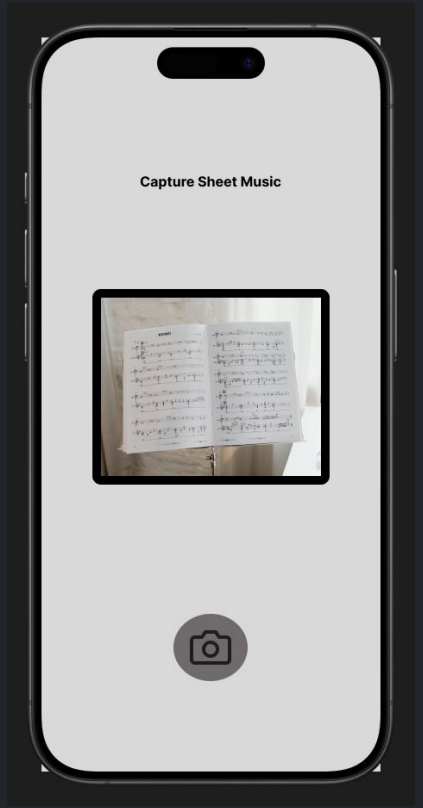
Requirement #3: Web Application

Motivation:

Users need a way to receive and store feedback after each “run”

Sub Requirements:

- ➔ Scan sheet music and convert to audio format with 96% accuracy
- ➔ Store historical data to mark improvements



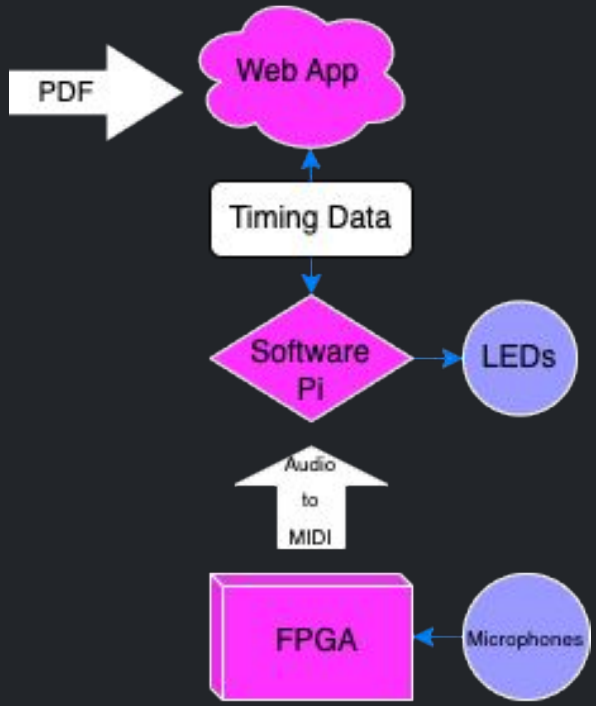
Technical Challenges

How do we…

- Accurately convert sheet music to an audio file format
- Add feedback onto sheet music
- Isolate instruments when processing mixed audio on a FPGA
- Algorithmically measure synchronicity between the pianist and singer
- What bounds do we set for margin of error between our 3 data points
- Account for musicality

Solution Approach (High Level)

- Web Application scans in sheet music which is converted to a machine readable format
- Sheet music data is sent to timing algorithm
- FPGA then sends audio data to the timing algorithm
- The timing algorithm determines synchronicity and sends it to the LEDS and Web App



Solution Approach

Software:

- Flask web application for backend
- Mozart and Werckmeister for sheet music to audio conversion
- Raspberry Pi
  - Comparing events lists
  - Generating feedback (timing) data

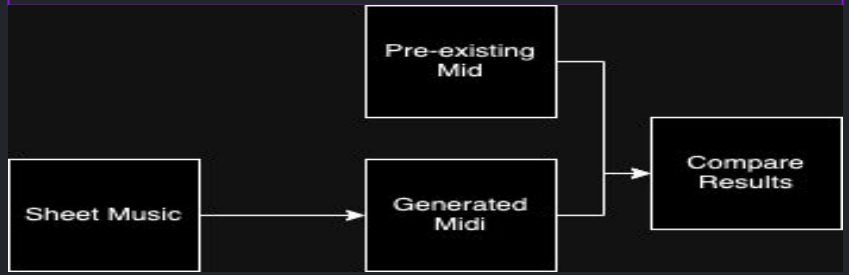
Hardware:

- FPGA for RTL audio processing
- PCB for stoplight array
- (2) Cardioid condenser microphones

Unit Testing and Verification

Web App

- Scanning sheet music
  - Music should be scanned in < 2 minutes
  - Result should be 96% similar
  - Compare to existing MIDI data
- Unit tests to test basic functionality of the frontend and backend



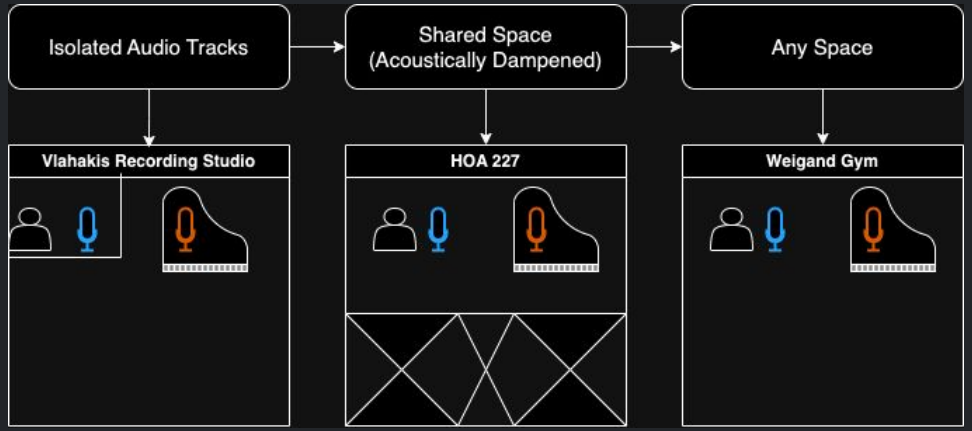
Timing Algorithm

- Calibrate with template event lists
  - Use MIDI files to simulate “perfect” timing
  - Edit recordings to delay to more than 30 mS to ensure timing detection
  - Make sure there is no data blocking if there is an error from either FPGA or sheet music with unit tests
  - Ensure that playing the same audio twice results in the same feedback output

Unit Testing and Verification (Continued)

Hardware and Audio Processing

- Calibrate with pre-recorded audio files
  - Known timing, pitch, number of notes
  - Aim for 95% transcribing accuracy
  - Compare expected timing and note quantity
  - Manually measure the delay (Audacity), verify system aligns within 15mS
- Audio Filtering
  - Start with Isolated tracks
  - Change sound isolation (location) when 95% accuracy reached



Division of Labor

Mathias	Aakash	Ben
- Music Scanner & Web Application	- Conversion Algorithm - Device Integration	- Audio Processor Hardware (FPGA) - PCB Design

Schedule (Design Stages)

Stage 1: Designing and Basic Testing

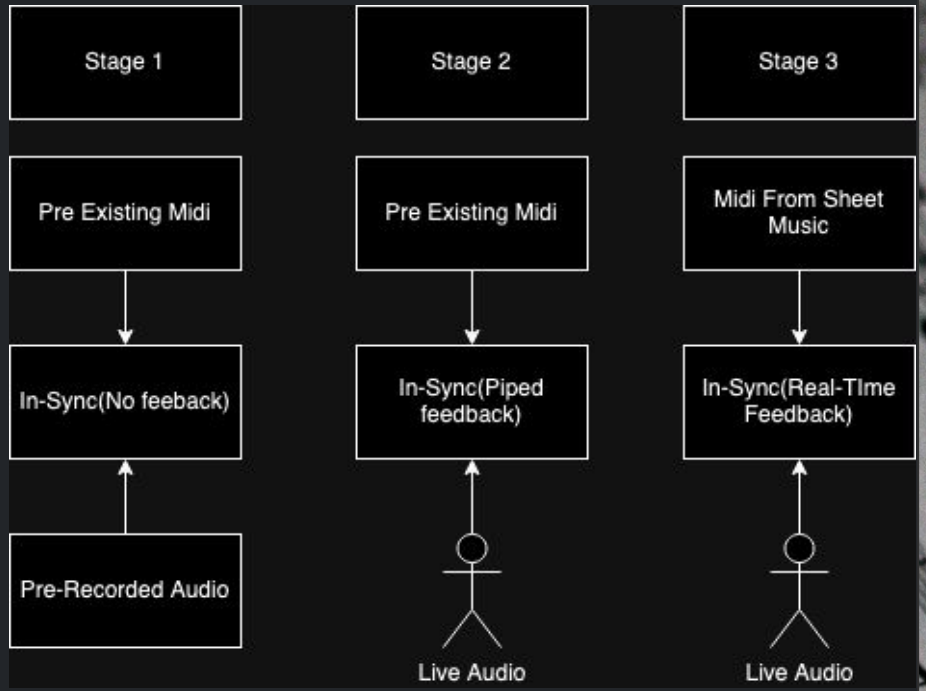
- Using pre-recorded audio to calibrate

Stage 2: RTL Audio and some Feedback

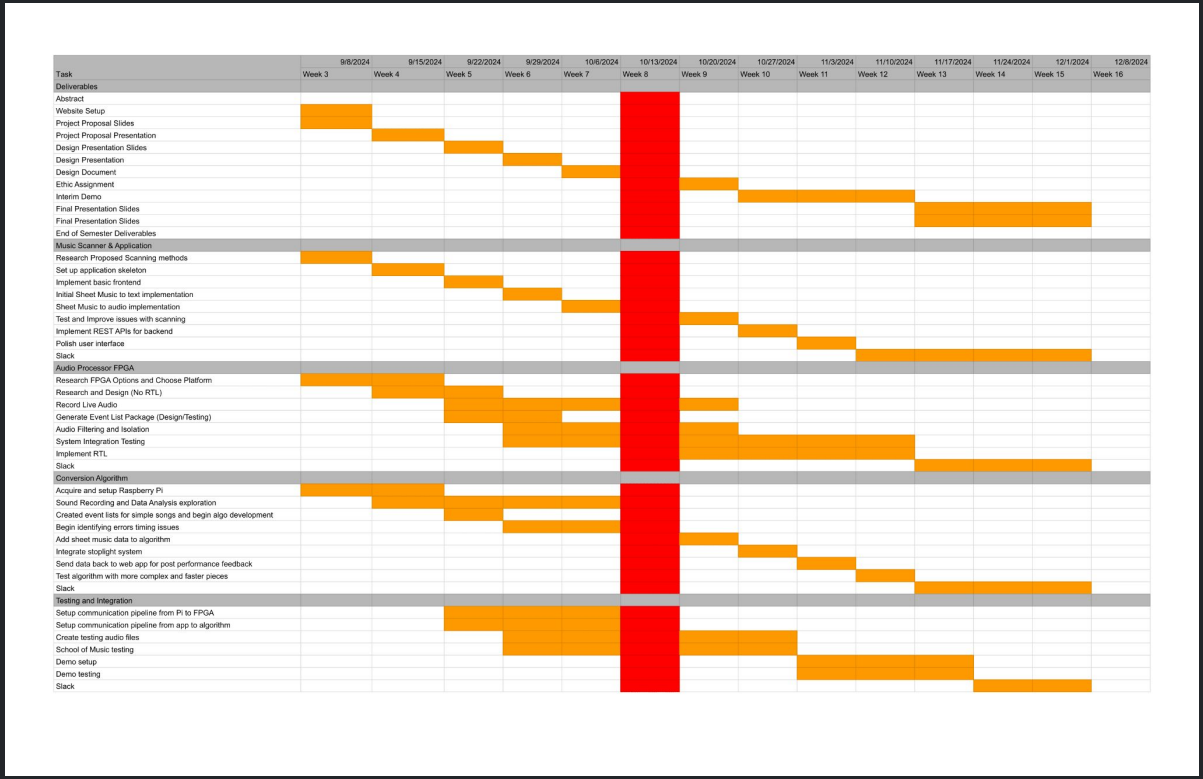
- Existing MIDI, and live audio

Stage 3: Full Web-app and feedback

- PDF Scanning, live audio (MVP)



Schedule (Gantt Chart)



Task	9/8/2024 Week 3	9/15/2024 Week 4	9/22/2024 Week 5	9/29/2024 Week 6	10/6/2024 Week 7	10/13/2024 Week 8	10/20/2024 Week 9	10/27/2024 Week 10	11/3/2024 Week 11	11/10/2024 Week 12	11/17/2024 Week 13	11/24/2024 Week 14	12/1/2024 Week 15	12/8/2024 Week 16
Deliverables														
Abstract														
Website Setup														
Project Proposal Slides														
Project Proposal Presentation														
Design Presentation Slides														
Design Presentation														
Design Document														
Ethic Assignment														
Interim Demo														
Final Presentation Slides														
Final Presentation Slides														
End of Semester Deliverables														
Music Scanner & Application														
Research Proposed Scanning methods														
Set up application skeleton														
Implement basic frontend														
Initial Sheet Music to text implementation														
Sheet Music to audio implementation														
Test and Improve issues with scanning														
Implement REST APIs for backend														
Polish user interface														
Slack														
Audio Processor FPGA														
Research FPGA Options and Choose Platform														
Research and Design (No RTL)														
Record Live Audio														
Generate Event List Package (Design/Testing)														
Audio Filtering and Isolation														
System Integration Testing														
Implement RTL														
Slack														
Conversion Algorithm														
Acquire and setup Raspberry Pi														
Sound Recording and Data Analysis exploration														
Created event lists for simple songs and begin algo development														
Begin identifying errors timing issues														
Add sheet music data to algorithm														
Integrate stoplight system														
Send data back to web app for post performance feedback														
Test algorithm with more complex and faster pieces														
Slack														
Testing and Integration														
Setup communication pipeline from Pi to FPGA														
Setup communication pipeline from app to algorithm														
Create testing audio files														
School of Music testing														
Demo setup														
Demo testing														
Slack														