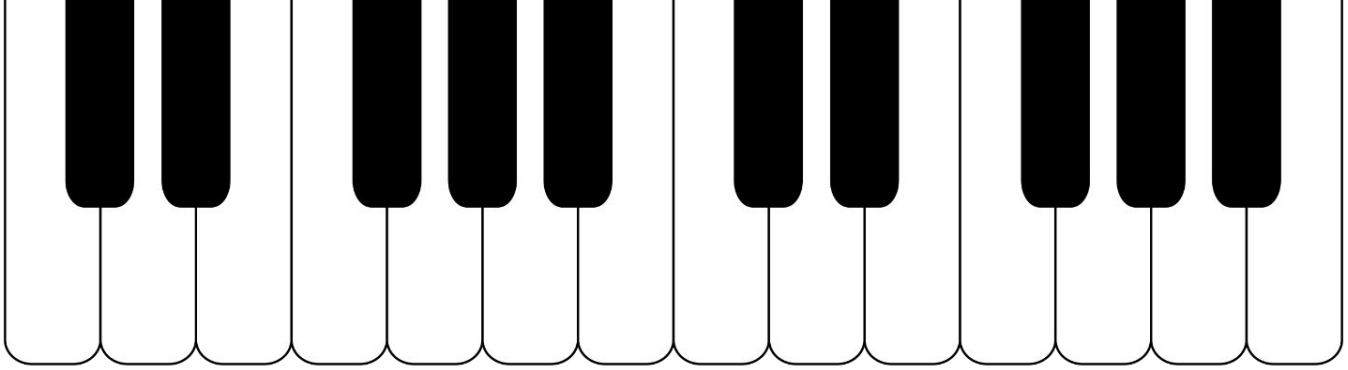
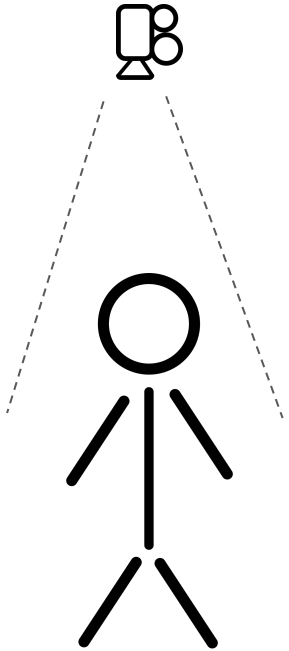


Team D2: Keynetic

Lance Yarlott, Katherine Dettmer, Sun A Cho

The Device: Current State

- Mechanically actuated keyboard that is managed by a microcontroller and computer
- Limited to simple notes and chords (as expected)
- Playing range: 14 keys on keyboard, can be extended via MIDI or through further hardware expansions
- Can store up to 8 measures of generated music
- Can also play on the fly
- Provides a solution for the lack of options to play music hands-free

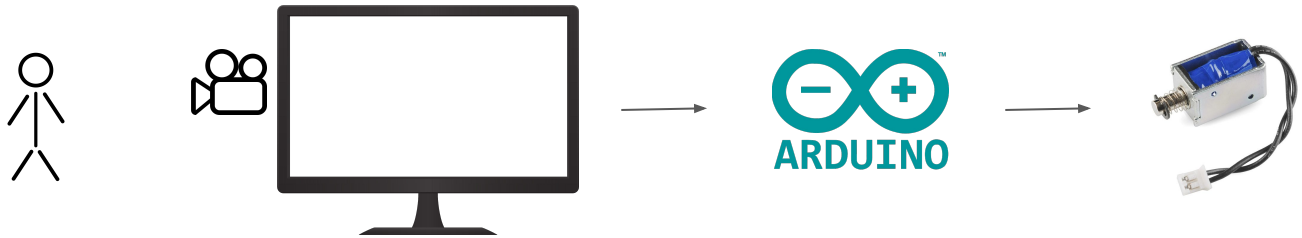


Requirements

- Hardware
  - Design and build an electrical system to actuate solenoids and play a keyboard - Complete
    - 7+ solenoids, up to 4 active at once
- Software
  - Detect hand pos >90% of the time - Complete
  - Reduce latency below 1s - Complete
- Music
  - On the downbeat, notes should *generally* fall on a chord tone - Complete
  - Generate musical phrases of diatonic notes in the key of C major - Complete

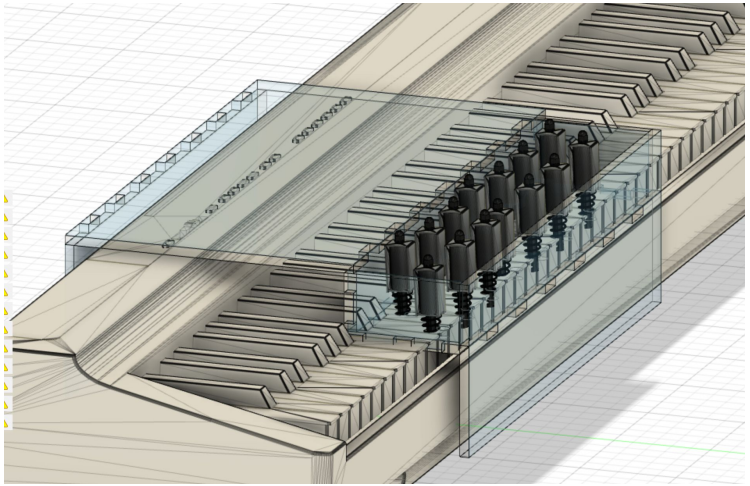
Solution Approach

- Power only the needed number of actuators
  - Avoids the use of unnecessary power
  - Current limit is 4 at once
- Use OpenCV to track hand position in screen space
  - Initially, we can divide the screen into grids, then detect what grid the hand is a part of
  - Detection done using a Haas Cascade
  - Then, we can track the hand for some period, figure out which sequence of boxes they went through, and decide on a generative pattern based on that



Solution Approach (contd.)

- Mounting System
  - Created in Fusion360, but first iteration (using 3mm acrylic) was too weak to hold 10 solenoids and the glue wasn't strong enough
  - Redesigned to use 6mm acrylic, and found glue that was strong enough to bond the acrylic pieces
  - The first iteration didn't have a way to secure the solenoids, so the second iteration was redesigned to be low enough in the solenoid area to have a piece secured over them.



The 3D model of the system. Note: keyboard is not accurate to the one we are using, our model fits accurately on ours.

Solution Approach (contd.)

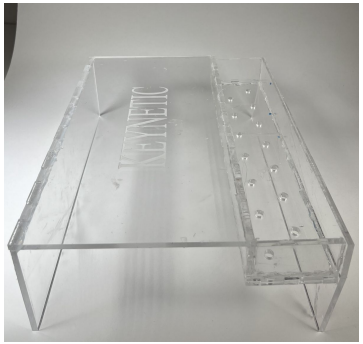
- Music Generation
  - Structs generally organized to behave like you would expect from sheet music
  - Cleaned up output
  - Whole 8 measures
    - Can do less

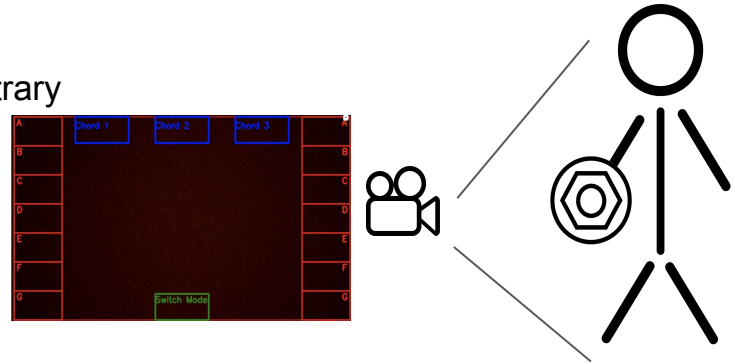
```
32  4 ------- 4 ------- 8 --- 32  4 ------- 16 -
16 - 16 - 16 - 8 --- 16 - 8 --- 32  16 - 8 --- 16 - 16 - 32  8 ---
16 - 16 - 4 ------- 8 --- 8 --- 16 - 4 ------- 32  32
8 --- 16 - 16 - 16 - 8 --- 32  8 --- 4 ------- 8 --- 32
8 --- 32  16 - 16 - 16 - 16 - 4 ------- 8 --- 16 - 16 - 32  16 -
2 --------------- 16 - 8 --- 16 - 16 - 8 --- 32  32
8 --- 8 --- 32  8 --- 32  8 --- 8 --- 16 - 8 --- 8 ---
8 --- 8 --- 16 - 8 --- 16 - 8 --- 16 - 16 - 8 --- 16 - 32  32
[['C', 'E', 'G'], ['F', 'A', 'C'], ['G', 'B', 'D'], ['F', 'A', 'C']]
[['C', 'E', 'G'], ['F', 'A', 'C'], ['G', 'B', 'D'], ['F', 'A', 'C']]
CG-------D-------D---DD-------E-
C-A-C-G---A-A---AG-G---G-G-GC---
C-C-B-------D---B---D-B-------FF
C---A-D-C-G---GG---G-------G---G
G---EG-F-F-F-G-------E---G-D-DE-
A---------------A-F---F-F-D---DD
D---G---BD---GE---E---E-D---D---
F---C---F-E---E-E---E-F-C---F-DD
```

- Arduino Sequencer
  - Takes in above data
    - Only 1 measure
  - Adds to phrase struct or adds to player struct
    - Depends on mode

Complete Solution (Software + Mounting)

- Mounting system designed in Fusion360 to withhold 10 solenoids on top of the piano keys
  - Clear acrylic, as shown on left, so the system can still be seen but no danger to the user as the wires cannot be accessed
- Software (Computer Vision)
  - Use 2 symbols that can be detected by the camera as a marker for where the ‘hands’ are
  - Triggers notes and processes patterns
- Software (Music)
  - Can either play notes from CV directly, or will take in arbitrary input and generate music based on that
  - Takes visual input and outputs to serial
- HW: All of the solenoids are soldered and heat-shrinked to prevent any safety issues



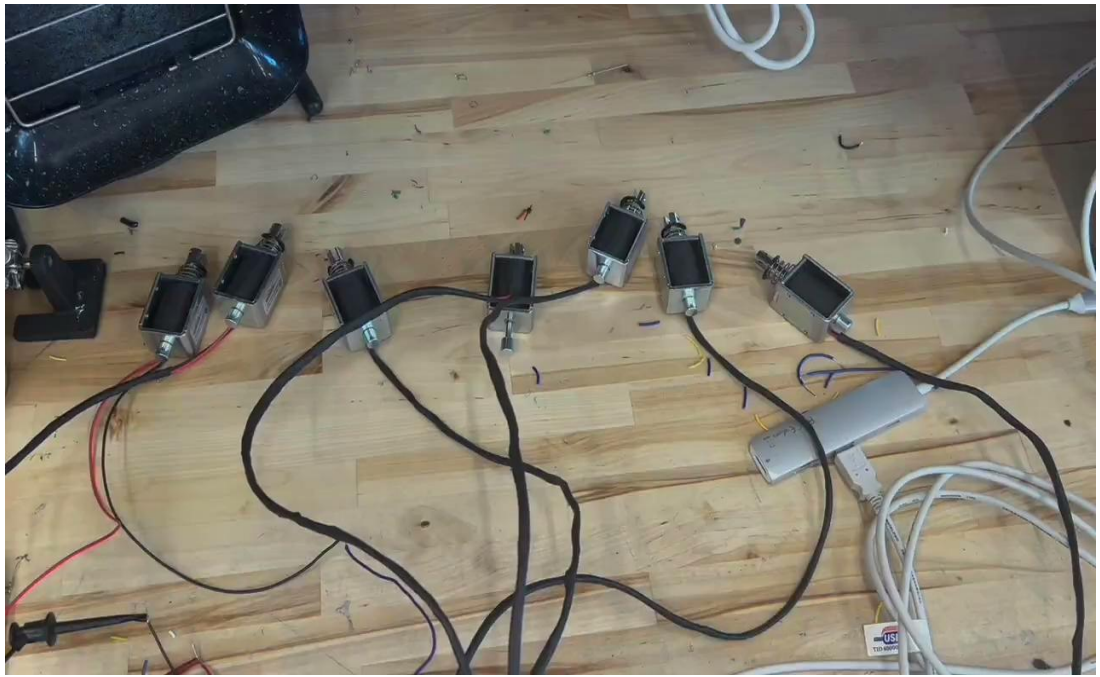


Implementation Challenges: Our Experience

- Solenoid powering, burning, and aging
  - Solenoids can be very sensitive and age with use
- Designing a comprehensive computer graphics system to detect a movement/position
  - Each position will correspond to a key or a chord
  - Currently testing with symbol detection
  - Backup Plan: using colored gloves on each hand for easy color detection
- Serial interfacing with Python and Arduino
  - “Slow”
  - Prone to errors due to the nature of handling single bytes
  - No GPIO

Testing, Verification and Metrics

- Hardware: playing random keys and chords



Testing, Verification and Metrics Cont.

- Computer Vision/ SW: measure accuracy of recognizing hands and their positions on the screen - Met
- Measure response time from when user makes motion to when the key is played (goal: < 1 sec) - Met
- Music Generation: Correctly pass notes to hardware at correct tempo (at least 60 bpm) - Met

Description	Goal	Measured
Hand Recognition Accuracy (using color)	> 90%	92%
Latency (from when a hand is placed in the box to when software recognizes)	< 1 second	0.6 seconds
Latency (from SW to when key is played)	< 1 second	0.3 seconds
Power to run the actuator system	Less than 30 V / 3 A	20V / 2A

Testing, Verification and Metrics Cont.

- Music Software can sequence notes and play them it time
  - Very accurate up to at least 120 bpm
  - Loops back around after 8 measures
  - Video taken w/ 15 bpm as reference
- Music generates as expected
  - Uses normal distributions to calculate probabilities of generating notes and rhythms
  - This is based on user input from CV
- Note sending to solenoid activation is under our requirement threshold
- Serial pipeline is robust and works >90% of the time
  - Errors include cable jostling, unreceived data, etc.

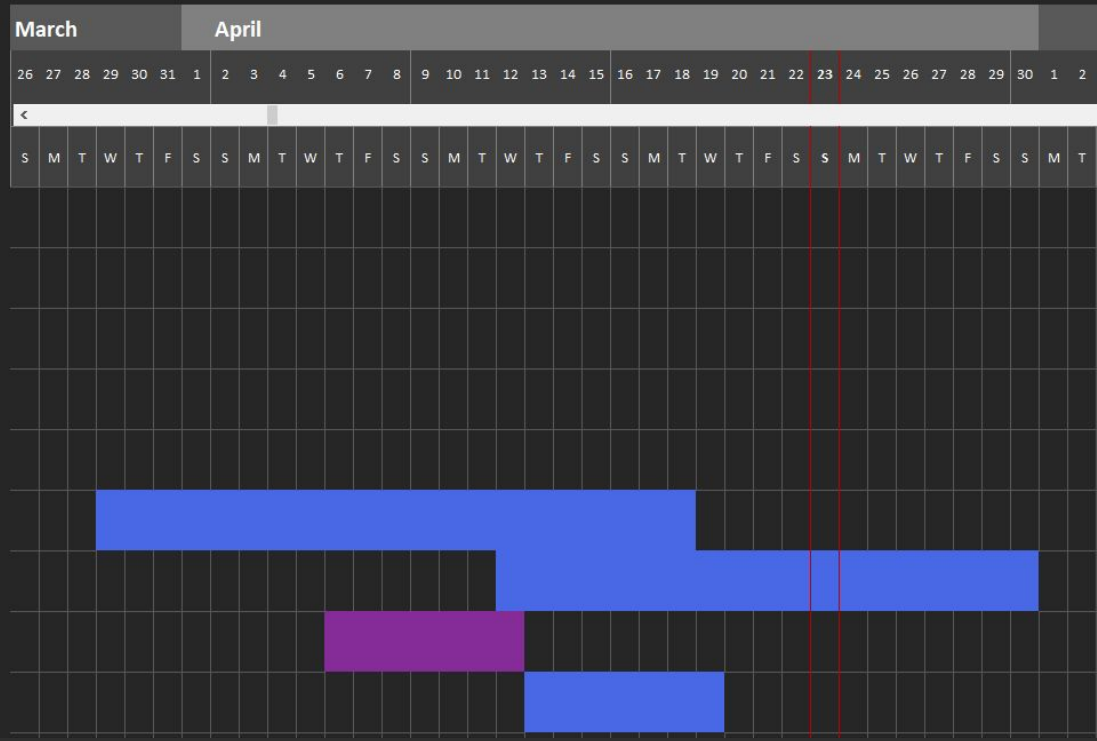
```
ready
0 : 1
0 : 2
0 : 3
0 : 4
0 : 5
0 : 6
0 : 7
0 : 8
0 : 9
0 : 10
0 : 11
0 : 12
0 : 13
0 : 14
```

```
tone_norm = get_truncated_normal(mean=tone_mood*2, sd=1, low=1, upp=10)
rhythm_norm = get_truncated_normal(mean=rhythm_mood*1.2, sd=1, low=1, upp=6)
```

Project Start Date: 1/30/2023

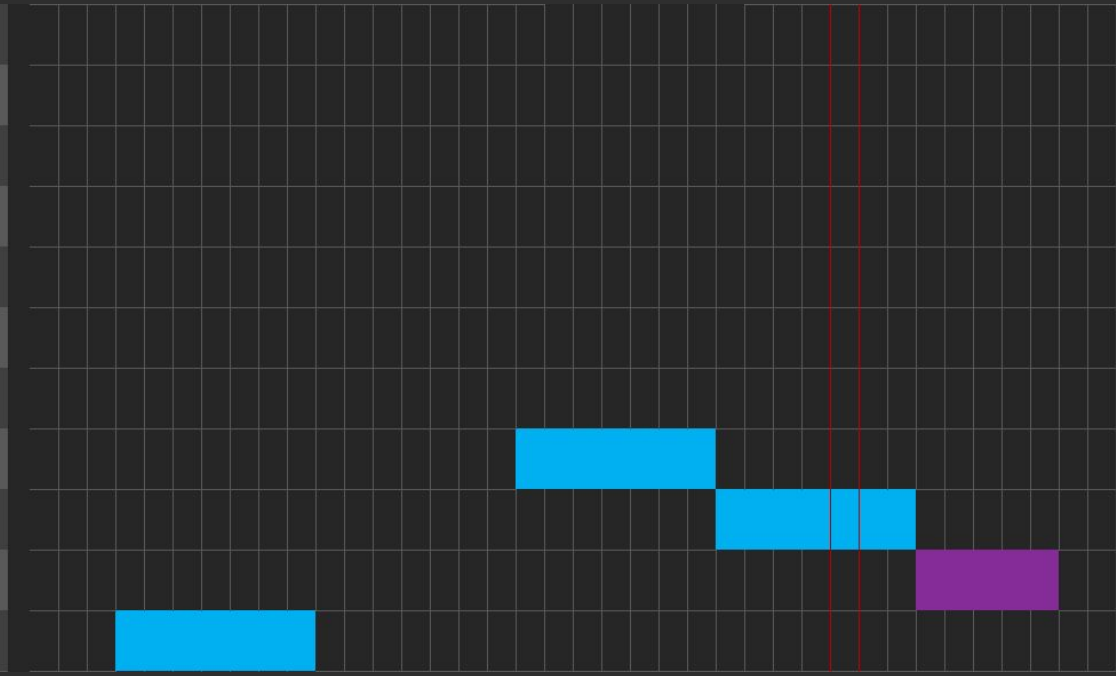
Scrolling Increment: 55

Milestone description	Category	Assigned to	Progress	Start	Days
Keyboard Hardware					
MIDI Keyboard Procurement	Milestone	Sun A	100%	2/13/2023	1
Frame Creation	Low Risk	Sun A	90%	2/14/2023	7
Actuator Selection	Med Risk	Sun A	100%	2/6/2023	7
Test System	Low Risk	Sun A	100%	2/20/2023	7
7-Solenoids System	Med Risk	Sun A	60%	3/29/2023	21
Complete System	Med Risk	Sun A	60%	4/12/2023	19
Actuator Integration	High Risk	Sun A	80%	4/6/2023	7
Software Integration	Med Risk	Lance, Katherine	100%	4/13/2023	7

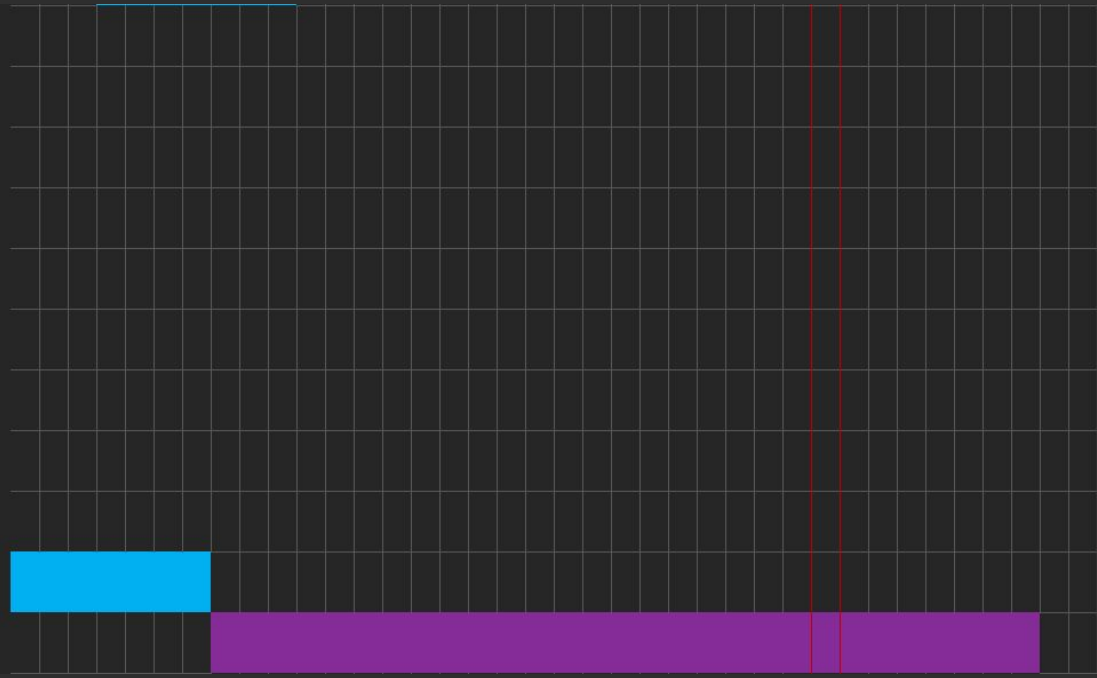


cv

Detection Method Selection	Low Risk	Katherine	100%	1/30/2023	7
Color Detection	High Risk	Katherine	100%	2/6/2023	8
Create Video Overlays	Med Risk	Katherine	100%	2/14/2023	7
Generate Notes for Note Generation Mode	Med Risk	Katherine	100%	2/21/2023	7
Generative Mode Mapping to Notes	Med Risk	Katherine	100%	2/28/2023	15
Provide Visual Feedback to User	Low Risk	Katherine	100%	3/15/2023	7
Distance Feedback	Low Risk	Katherine	0%	4/12/2023	7
Overall Integration	Low Risk	Katherine	80%	4/19/2023	7
Symbol Detection	High Risk	Katherine	50%	4/26/2023	5
Music Software Integration	Low Risk	Katherine, Lance	100%	3/29/2023	7



Music Synthesis					
Note Generation Algorithm	On Track	Lance	100%	3/6/2023	7
Note Sequencing (In-time Quarter Notes)	High Risk	Lance	100%	2/13/2023	6
Pitch Selection	High Risk	Lance	100%	2/19/2023	6
Chord Sequencing	Med Risk	Lance	100%	2/25/2023	7
Advanced Music Synthesis					
Subdivions	Low Risk	Lance	100%	2/19/2023	7
Syncopation	Low Risk	Lance	100%	2/26/2023	7
Chords (Non-Triads)	Low Risk	Lance	100%	3/5/2023	7
Code Cleanup and Optimization	Low Risk	Lance	95%	3/12/2023	21
Final Debugging and Serial Integration	High Risk	Lance	80%	4/2/2023	29



Verification and Validation					
Unit Testing	Med Risk	All	95%	4/1/2023	7
Hardware Testing	Low Risk	Sun A	100%	4/8/2023	7
Software Testing	Low Risk	Katherine, Lance	60%	4/8/2023	14
Music Synthesis Testing (Part of Unit Testing)	Med Risk	Lance	100%	4/1/2023	7
Integration Testing	High Risk	All	75%	4/8/2023	23
Slack Creation	Goal	All	100%	2/5/2023	1