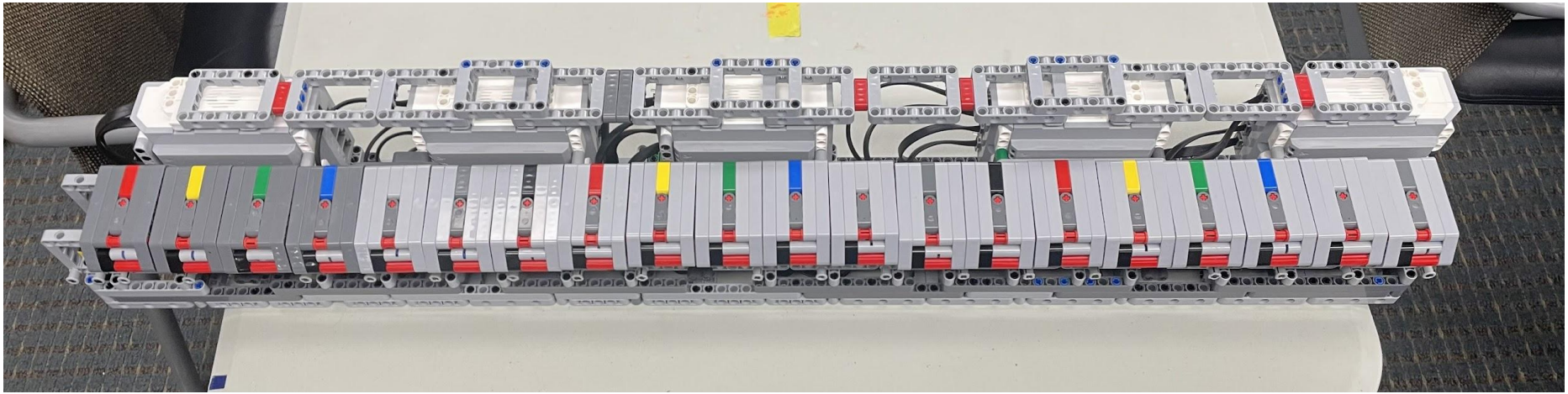
Automatic Piano

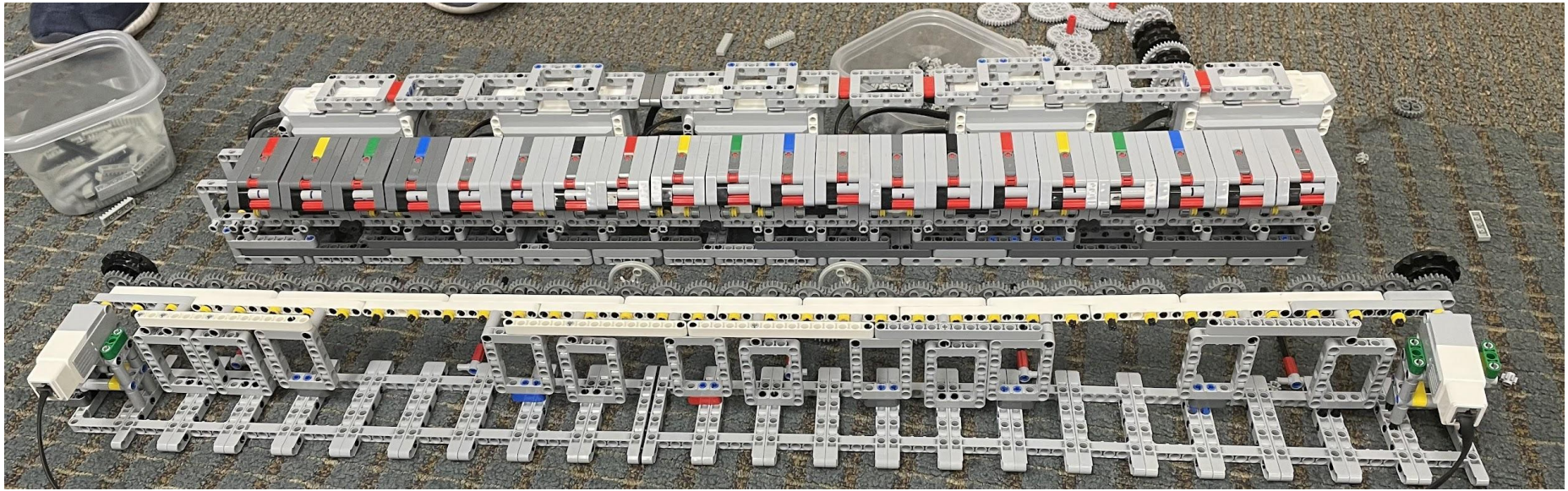
by Theo, Lucas, Dhruv

Our Design



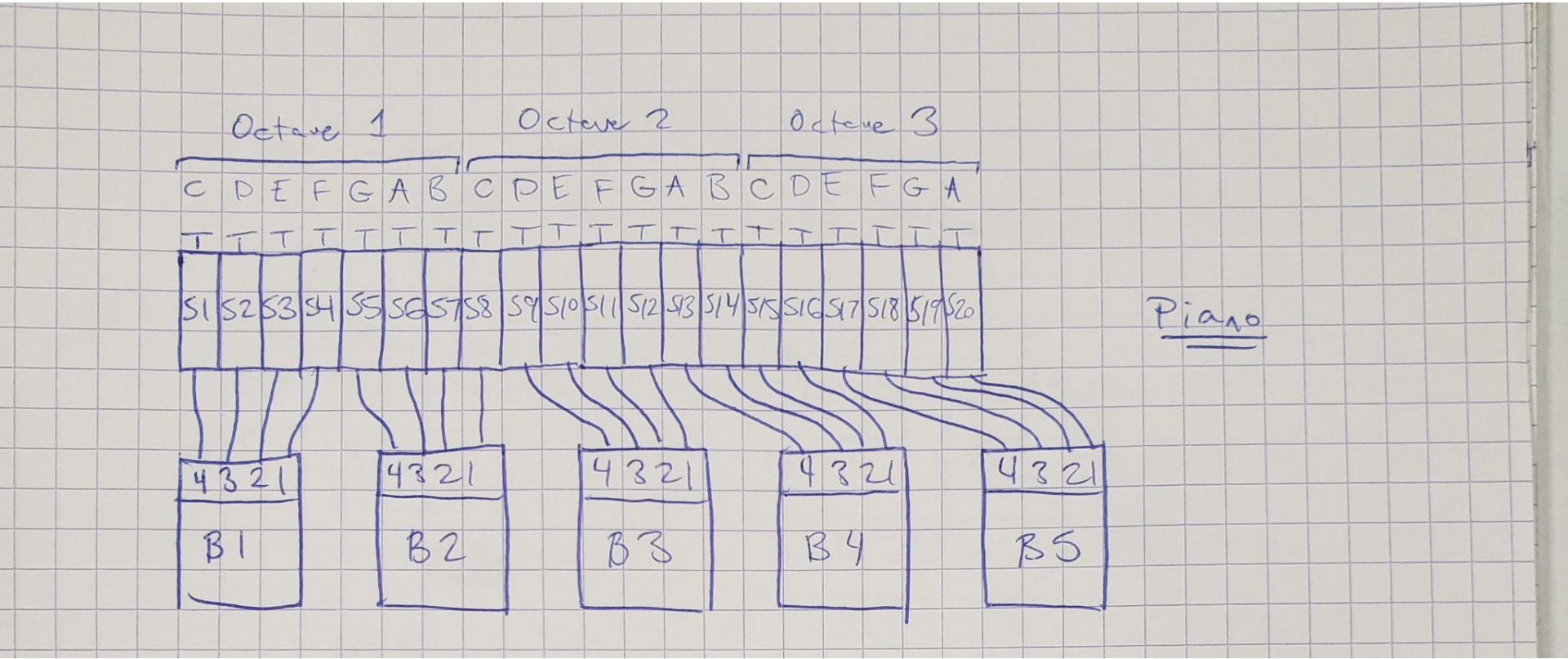
- 20 keys
- Playable by human

Our Design



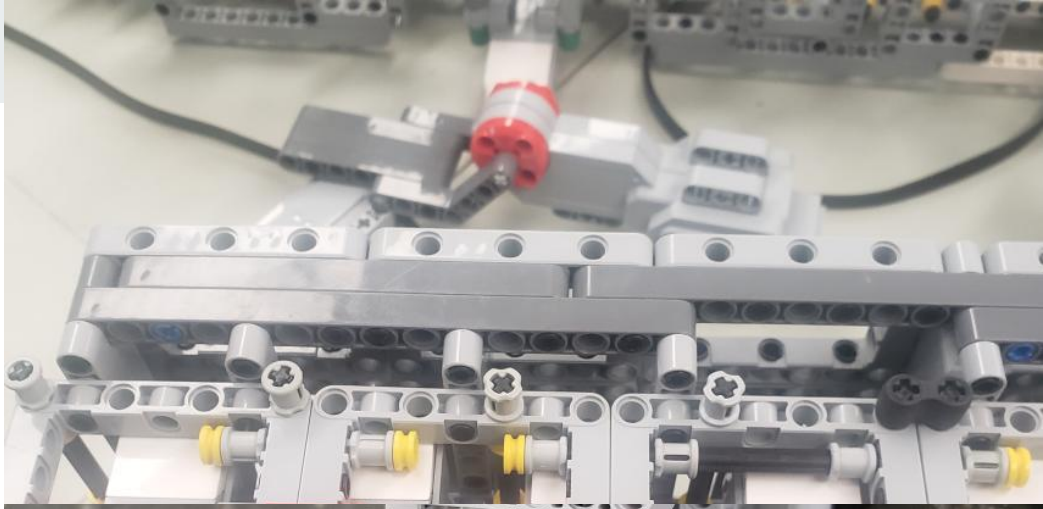
- Playable by robot

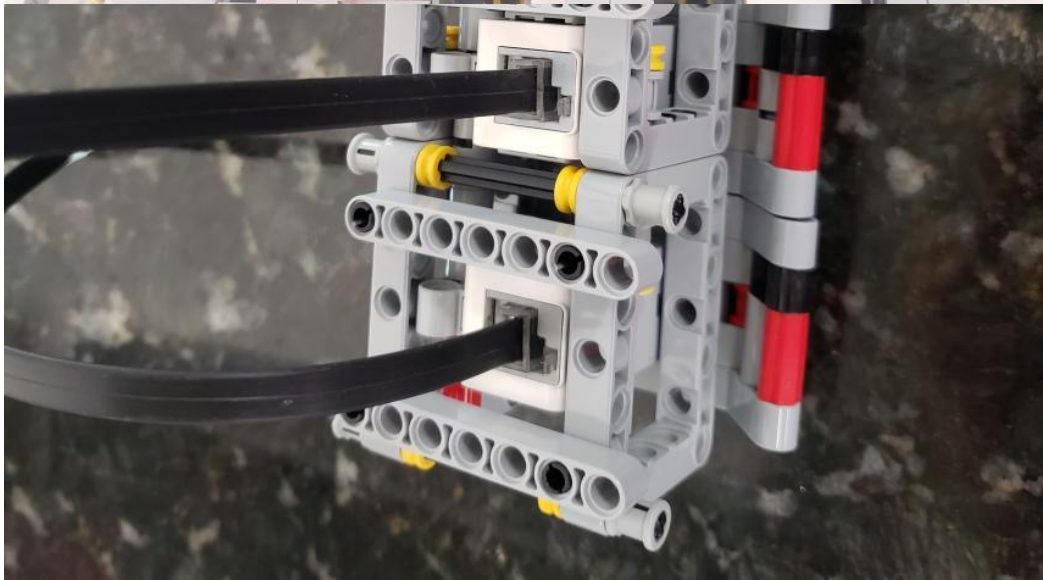
Wire Map


Octave 1
Octave 2
Octave 3
C D E F G A B C D E F G A B C D E F G A
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S20
4 3 2 1
B1
B2
B3
B4
B5
Piano

Fabrication Process

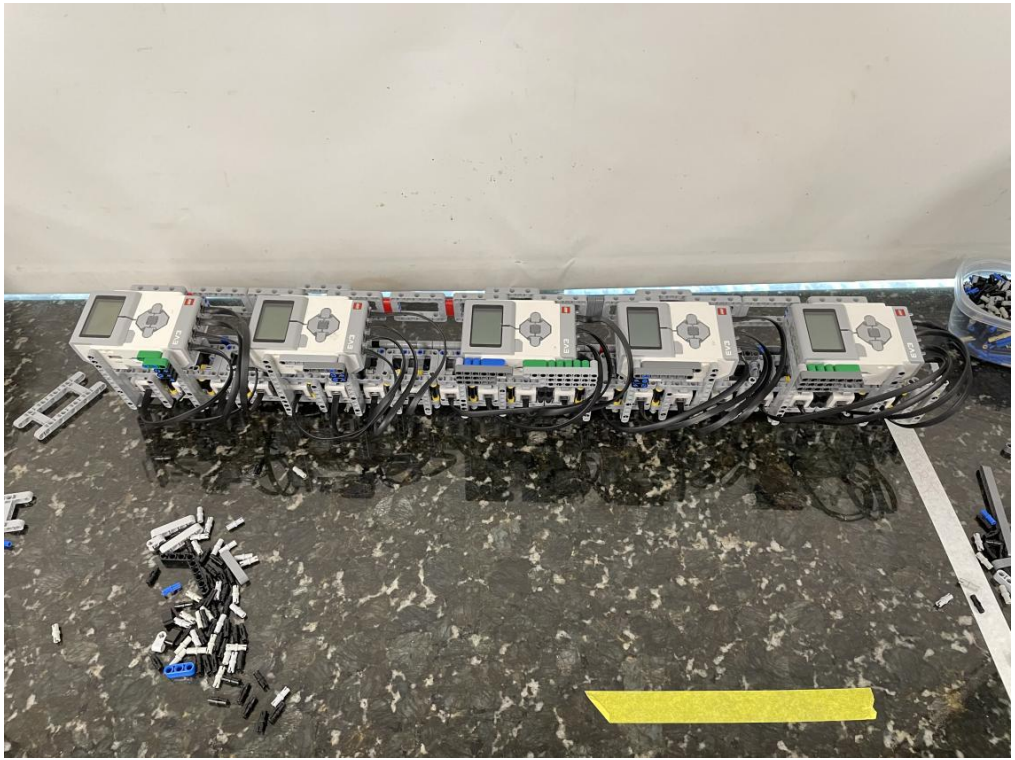
- Modular
- Color coded notes
- **High structural integrity!!!**

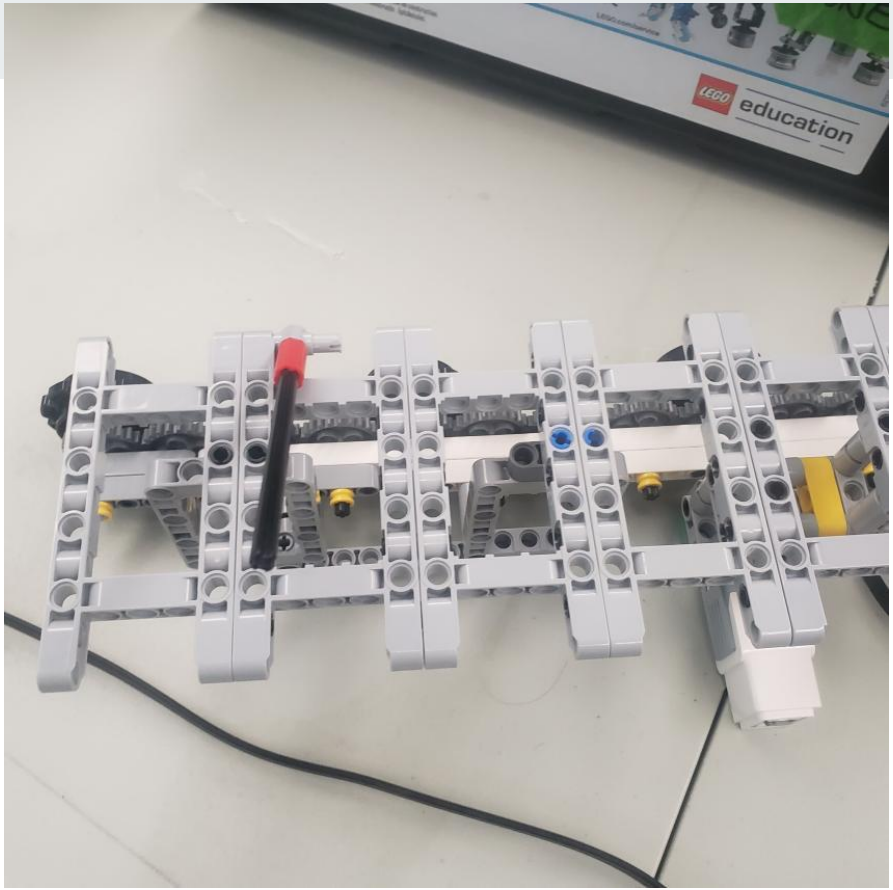


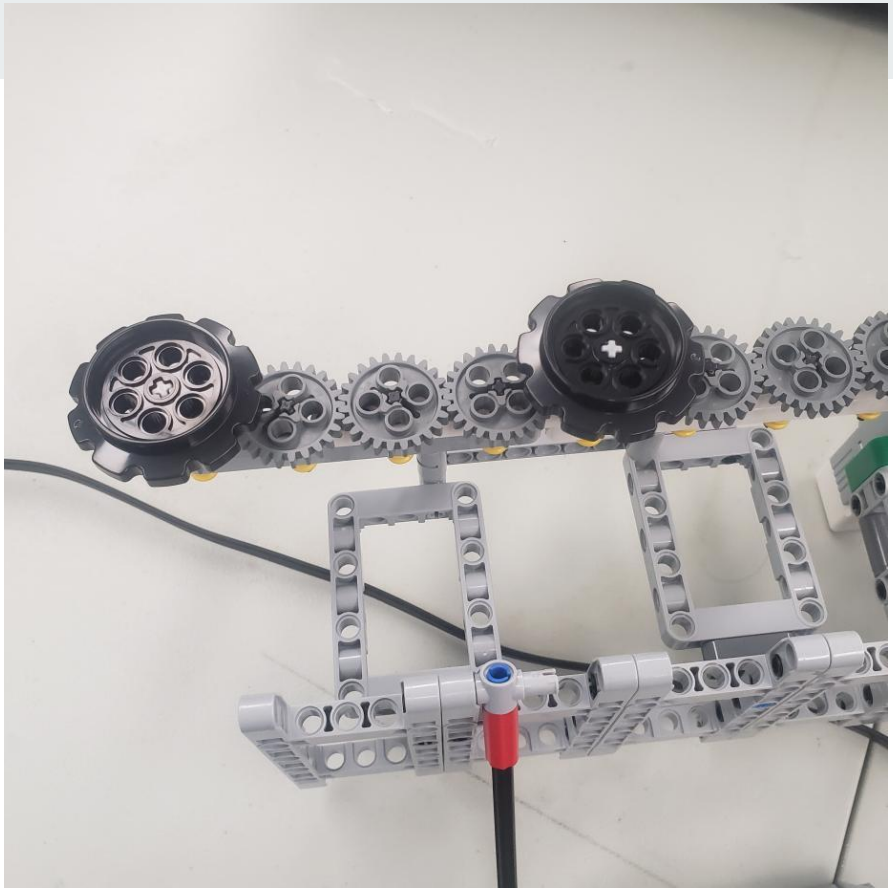


Fabrication Process

- Speakers facing up
- Wire management
- More structure





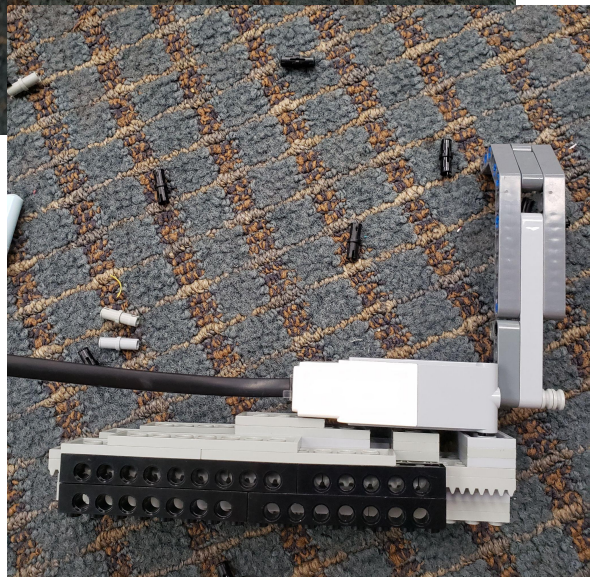


The Finger Track 1.0 - Primary Obstacle

Traction and Tension

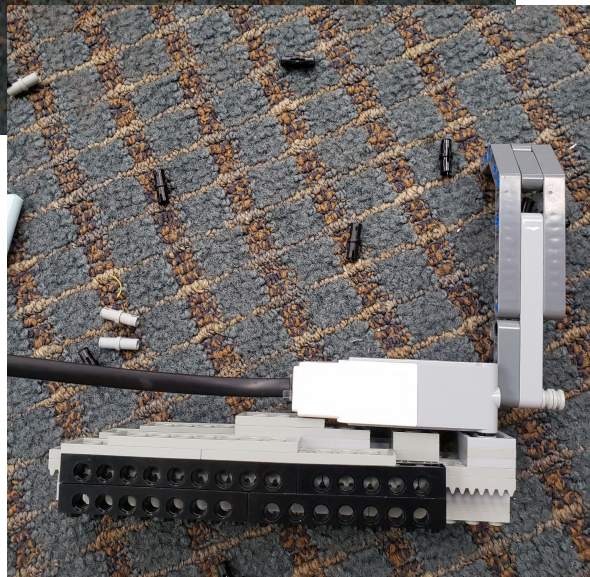
Finger Track 2.0

- Alternating gears
- Linear carriage
- More stable
- Still inconsistent



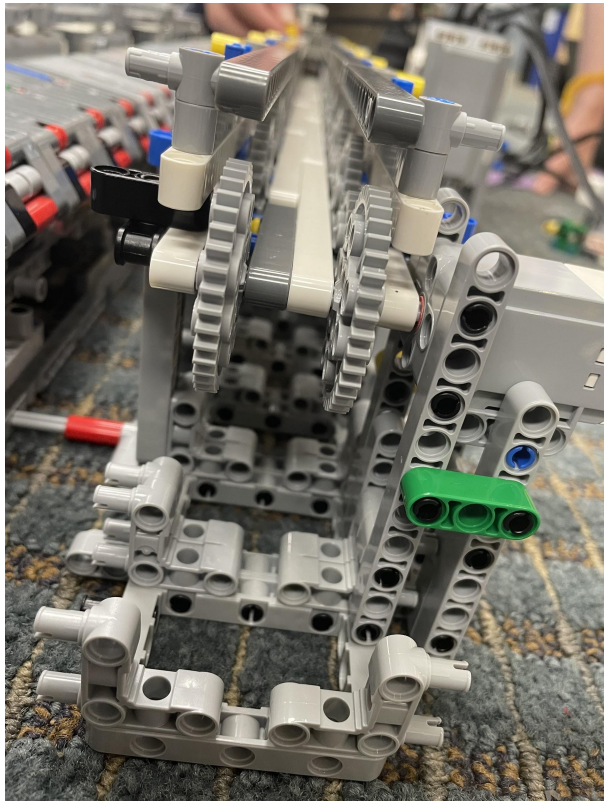
Finger Track 2.0

- Alternating gears
- Linear carriage
- More stable
- Still inconsistent



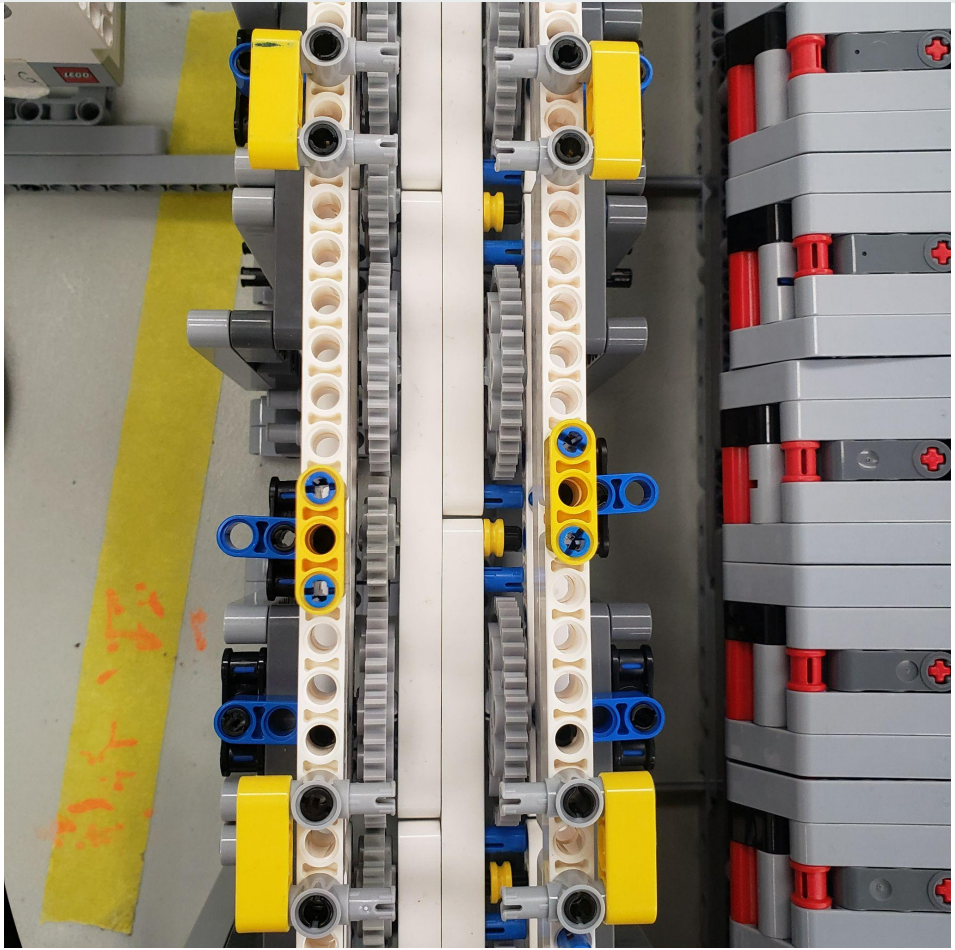
Finger Track 3.0

- “Stability walls” added to gear assembly
- Carriage gets stuck less often



Finger Track 4.0

- Replace top walls with pegs
- Carriage no longer gets stuck on walls
- More accuracy and consistency



Piano Programming: Pseudo

Store which keys are down

When key (up -> down), play sound

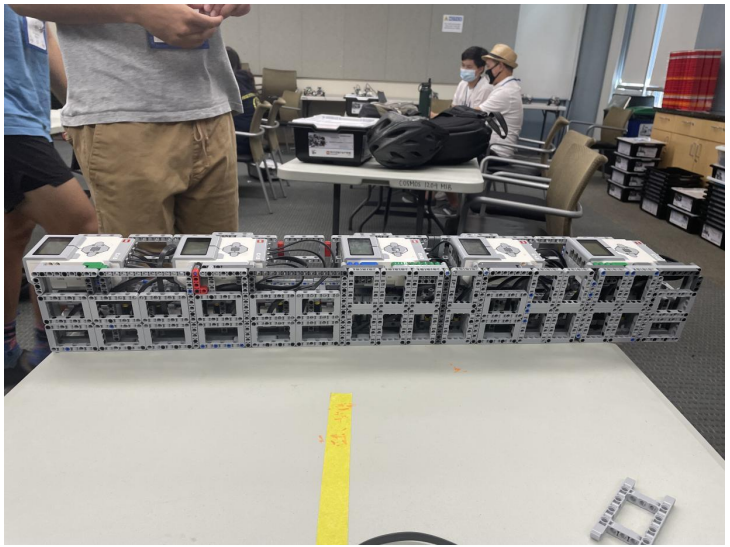
Store iterable array of pitches, one per key

```
#pragma config(Sensor, S1,     ,               sensorEV3_Touch)
#pragma config(Sensor, S2,     ,               sensorEV3_Touch)
#pragma config(Sensor, S3,     ,               sensorEV3_Touch)
#pragma config(Sensor, S4,     ,               sensorEV3_Touch)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  int numSensors = 4;
  bool pressed[4] = { false, false, false, false };
  short sensors[4] = { S1, S2, S3, S4 };
  int tones[4] = { 349, 330, 294, 262 };
  while(true) {
    for(int i = 0; i < numSensors; i++) {
      int val = SensorValue(sensors[i]);
      if(val > 0 && !pressed[i]) {
        pressed[i] = true;
        playImmediateTone(tones[i], 10);
      } else if(val < 1 && pressed[i]) {
        pressed[i] = false;
      }
    }
  }
}

```

```
1  int numPerMove = 115;
2  string key[] = { "A3", "G3", "F3", "E3", "D3", "C3", "B2", "A2", "G2",
3  int numKeys = sizeof(key) / sizeof(key[0]);
4  int speed = 100;
5  void playNote() {
6    setMotorTarget(motorD, 60, 50);
7    waitUntilMotorStop(motorD);
8    setMotorTarget(motorD, 100, 100);
9    waitUntilMotorStop(motorD);
10   setMotorTarget(motorD, 0, 100);
11   waitUntilMotorStop(motorD);
12 }
13 void moveTo(int target, bool skip = false) {
14   setMotorTarget(motorA, target * numPerMove, speed);
15   setMotorTarget(motorB, target * numPerMove, speed);
16   waitUntilMotorStop(motorA);
17   waitUntilMotorStop(motorB);
18   if(!skip) {
19     playNote();
20   }
21 }
```

State Machine

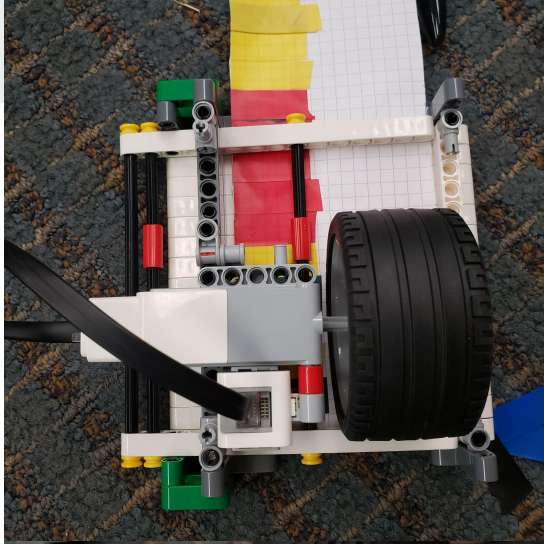
- 4 buttons -> 4 songs
  - Hot cross buns
  - Twinkle twinkle little star
  - Final Countdown
  - Lucid Dreams

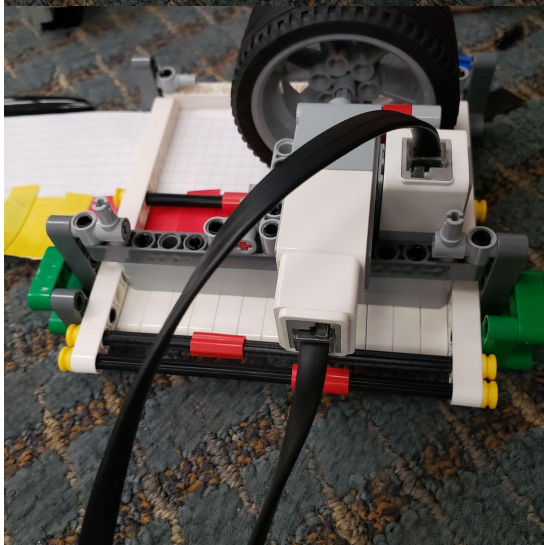


```
22  void playSong(char* input) {
23    int len = strlen(input) / 2;
24    for(int i = 0; i < len; i++) {
25      char* res = "  ";
26      res[0] = input[2 * i];
27      res[1] = input[2 * i + 1];
28      for(int j = 0; j < numKeys; j++) {
29        if(key[j] == res) {
30          moveTo(j);
31          break;
32        }
33      }
34    }
35    moveTo(0, true);
36  }
37  task main()
38  {
39    while(true) {
40      waitForButtonPress();
41      if(getButtonPress(buttonUp)) {
42        playSong("C3C3G3G3A3A3G3F3F3E3E3D3D3C3G3G3F3F3E3E3D3G3G3F3F3E3E3D3C3C3G3G3A3A3G3F3F3E3E3D3D3C3");
43      } else if(getButtonPress(buttonDown)) {
44        playSong("E3E3C3C3G3F3E3E3D3E3E3C3C3G3F3E3E3D3E3E3C3C3G3F3E3E3D3D3E3E3F3E3C3C3E3E3F3E3C3C3E3E3F3E3");
45      } else if(getButtonPress(buttonLeft)) {
46        playSong("E3D3C3E3D3C3C3C3C3C3D3D3D3D3E3D3C3");
47      } else if(getButtonPress(buttonRight)) {
48        playSong("E3D3E3A2F3E3F3E3D3F3E3F3A2D3C3D3C3B2D3C3A2");
49      }
50    }
51  }
52
```

Reading Music

- Only red, yellow, green
- Play notes one at a time on one octave
- Completely different program





```
55  void readNote(int noteNumber) {
56    //green is between 5 and 8
57    //yellow is between 77 to 81
58    //red is between 68 - 71
59    setMotorTarget(motorC, noteNumber * 20, 10);
60    waitUntilMotorStop(motorC);
61
62    int colorVal = sense();
63
64    if (colorVal <= 80 && colorVal >= 50) {
65      //Brain displays color it detects
66      displayBigTextLine(1, "yellow %d", colorVal);
67      moveTo(4);
68      } else if (colorVal <= 49 && colorVal >= 25  ) {
69      displayBigTextLine(1, "red %d", colorVal);
70      moveTo(5);
71      } else if (colorVal <= 24 && colorVal >= 0){
72      displayBigTextLine(1, "green %d", colorVal);
73      moveTo(3);
74    }
75  }
```

```
77  task main()
78  {
79    int noteNumber = 0;
80    while (true) {
81      readNote(noteNumber);
82      noteNumber++;
83    }
84  }
```

Summary

- Human player
- Robot player
- Music reader
- Detachable