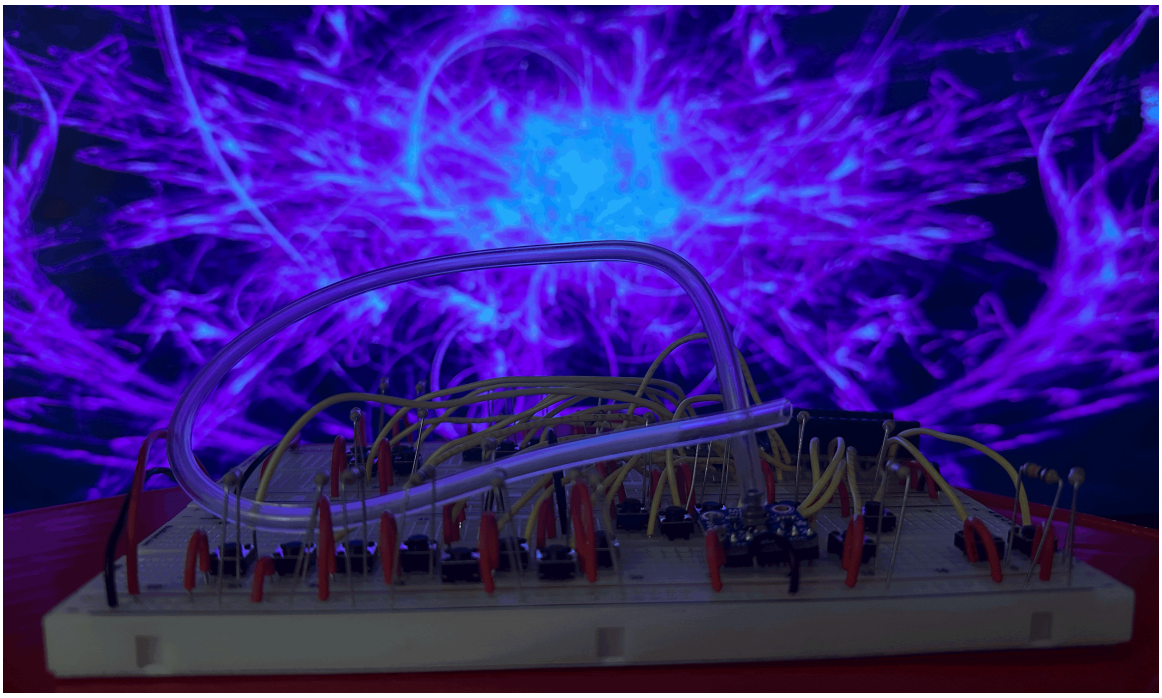
Digital Electronics Lab Fall Final Report

Braden Cantor-Goldner



12/08/2024

NYU STEINHARDT

What Is It?

For my final project I built a **MIDI wind controller** that takes in a combination of breath and fingering data to output MIDI to a DAW(Logic Pro in the case of the demo). What makes this special is that it **uses clarinet fingerings**, a feature which no commercially available wind synth or EWI(Electronic Wind Instrument) has. There are plenty of devices catering to other woodwind musicians such as flautists and saxophonists but as a clarinet player myself I have always wanted to have an electronic version of my instrument to practice and perform on.

Demo Video: https://youtu.be/-sFTWze4Yus

How It Works

Breath Control

By blowing through a piece of clear plastic tubing you can **control the volume** of the instrument with your breath. At the other end of the plastic tubing is an air pressure sensor that measures how much pressure is built up in a small space. The small space on the breakout board is extended by the plastic tubing. The best way to go about playing it is to hold your tongue over the end of the plastic tubing and start blowing. When you release your tongue it will create sound. This is the same way you'd play a real clarinet with the release of your tongue causing the reed to vibrate and start a note. You can crescendo and decrescendo by both covering and uncovering the opening with your tongue or more like how you would on a real clarinet you can change the speed of the air you're blowing. The only unique part(s) I had to purchase for this project was the **MPRLS air pressure sensor** which can be found here and the tubing to blow into said sensor that doesn't come with it.

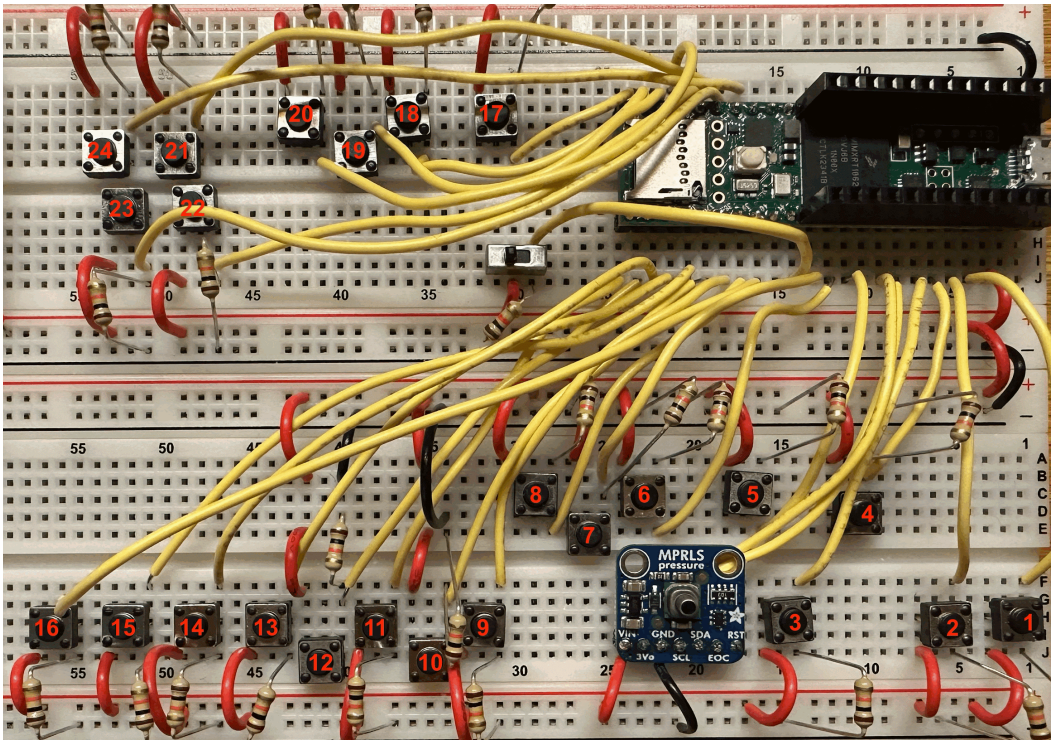
Fingerings

On the breadboard there are 24 buttons that correspond to the 24 keys of a clarinet which **control what note you're playing**. Each one is wired to its own pin on Teensy and is handled in code. The explanation of the code can be found later in this document and the diagram of what button relates to what key can be found on the next page. This **doesn't work like a keyboard** where you press one button down and get a sound. You have to be pressing all the required buttons to make a fingering to get the desired note.

Table Of Clarinet Keys/Clar-E-net Buttons

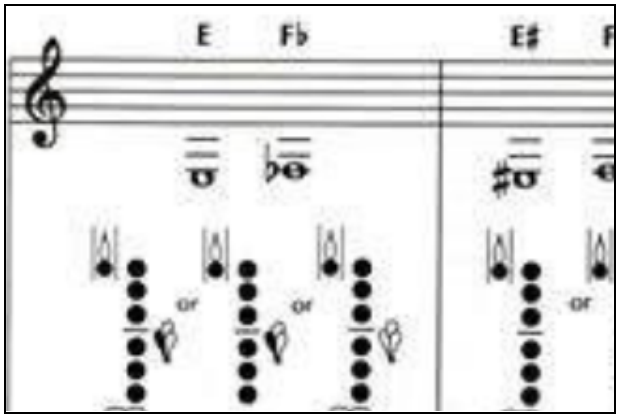
Key	#	Key	#	Key	#	Key	#
registerKey	1	bananaTopKey	7	sideKey1	13	bananaLowKey	19
thumbKey	2	threeKey	8	sideKey2	14	sixKey	20
leftSideKey	3	cSharpKey	9	sideKey3	15	ebKey	21
aKey	4	leftPinkyExt1	10	sideKey4	16	rightPinky1	22
oneKey	5	leftPinkyExt2	11	fourKey	17	rightPinky2	23
twoKey	6	leftPinkyExt3	12	fiveKey	18	rightPinky3	24

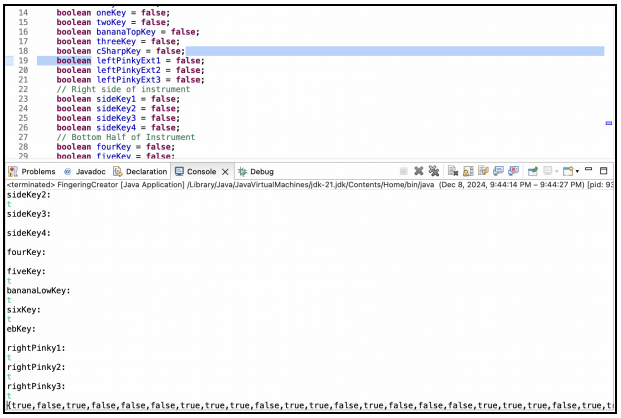
Breadboard Diagram

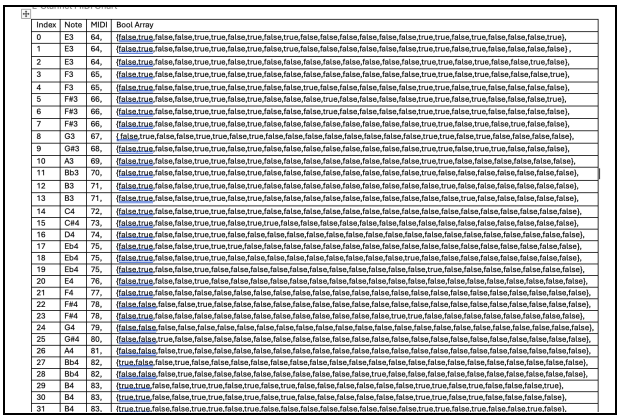


Technical Code Explanation

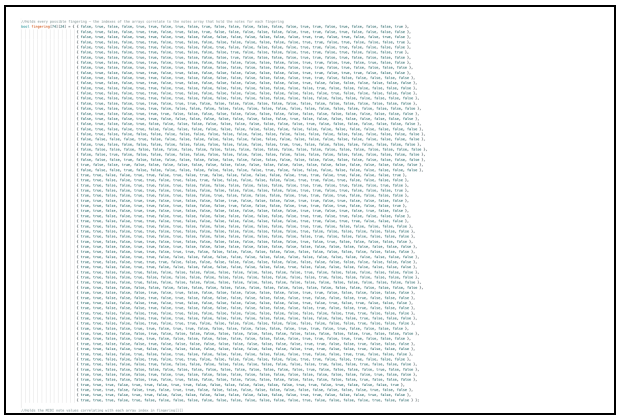
How I wrote the code to store possible fingering combinations







Index	Note	MIDI	Bool Array
0	E3	64,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,true,true,false,true,false,false,false,true},
1	E3	64,	{false,true,false,false,true,true,true,false,true,true,false,false,false,false,false,true,true,true,false,true,false,false,false} ,
2	E3	64,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,true,false,false,false,false,false},
3	F3	65,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,true,false,false,false,false,true},
4	F3	65,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,true,false,false,false,false,false},
5	F#3	66,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,true,false,false,false,false,true},
6	F#3	66,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,true,false,false,false,false,false},
7	F#3	66,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,true,false,false,false,false,false},
8	G3	67,	{ false,true,false,false,true,true,true,false,true,false,false,false,false,false,false,true,true,true,false,false,false,false},
9	G#3	68,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,true,false,false,false,false,false},
10	A3	69,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,true,false,false,false,false,false},
11	Bb3	70,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
12	B3	71,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
13	B3	71,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
14	C4	72,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
15	C#4	73,	{false,true,false,false,true,true,true,true,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
16	D4	74,	{false,true,false,false,true,true,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
17	Eb4	75,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
18	Eb4	75,	{false,true,false,false,true,true,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
19	Eb4	75,	{false,true,false,false,true,false,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
20	E4	76,	{false,true,false,false,true,false,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
21	F4	77,	{false,true,false,false,false,false,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
22	F#4	78,	{false,false,false,false,true,false,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
23	F#4	78,	{false,true,false,false,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false,false,false},
24	G4	79,	{false,false,false,false,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false,false,false},
25	G#4	80,	{false,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
26	A4	81,	{false,false,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
27	Bb4	82,	{true,false,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
28	Bb4	82,	{false,false,false,true,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false,false,false},
29	B4	83,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,true,true,false,false,false,false,false,true},
30	B4	83,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
31	B4	83,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,true,true,false,false,false,false,false,false},



Explanation: The image on the top left is an example of a clarinet fingering chart slide that I used to create the “fingering chart library” for my instrument in code. Because I knew I was going to have many long arrays of booleans to store whether or not buttons were pressed and I didn’t trust my brain to remember so much I **wrote a helper program in Java**. The Java program I wrote(top right) printed a button name to the console and if I typed ‘t’ it would mark the variable as being true and if I hit enter it would mark it as false. After all the buttons were given for a specific fingering the program then printed the formatted array that I could copy and paste into the Arduino IDE. I stored all of these array outputs I got from my program in a word doc table(bottom left) that also had the MIDI note equivalent values with the index of the array. This made it so I could select the whole columns of both the note[] and for the fingering[][] in the word doc and **copy and paste into the Arduino IDE**(bottom right) without small mistakes.

How does the device use that to know what note to play?

```
void createFingerArray() {
  for (int i = 0; i < 24; i++) {
    activeFingering[i] = digitalRead(buttonPins[i]);
  }
  delay(5);
}

int getNote() {
  for (int i = 0; i < 74; i++) {
    bool boolean = true;
    for (int j = 0; j < 24; j++) {
      if (activeFingering[j] != fingering[i][j]) {
        boolean = false;
        continue;
      }
      if (j == 23 and boolean == true) {
        return notes[i]-2;
      }
    }
  }
  return 77;
}
```

These two functions allow the code to take what the device is currently reading and compare it to the library of fingerings. The function createFingerArray() takes in all of the button readings from the 24 input pins and **creates this array called activeFingering**. The getNote() function then takes the newly created activeFingering array and compares it to all 74 possible fingerings in the library. If it finds it then **it passes along the MIDI value** for the note associated with the fingering. If the activeFingering array doesn't exist in the library it returns the number for open G meaning no buttons are pressed down on the instrument or the fingering doesn't exist. In the case that the fingering doesn't exist I still wanted it to return open G as that's what happens a decent amount of the time when actually playing clarinet.

```
void playNote() {
  prevNote = currentNote;
  currentNote = getNote();
  if (currentNote != prevNote) {
    usbMIDI.sendNoteOff(prevNote, 127, 1);
    usbMIDI.sendNoteOn(currentNote, 127, 1);
  }
}
```

The other function that allows notes to be played by instruments is the playNote() function which takes in the MIDI note value given by getNote() and **checks to see if that's the note already being played**. If it's not then it turns off the previous note and begins the new note. If it's the same note then it does nothing as to not overload the daw with unnecessary MIDI data. This is very **similar to the better button code** that we wrote in class conceptually.

Technical Code Explanation(Cont.)

How does the breath sensor code work?

```
if (!mpr.begin()) { //Breath Sensor Setup
  Serial.println("Failed to communicate with MPRLS sensor, check wiring?");
  while (1) {
    delay(10);
  }
}
delay(1000);

firstBreath = mpr.readPressure();
pressureMax = firstBreath + 22;
```

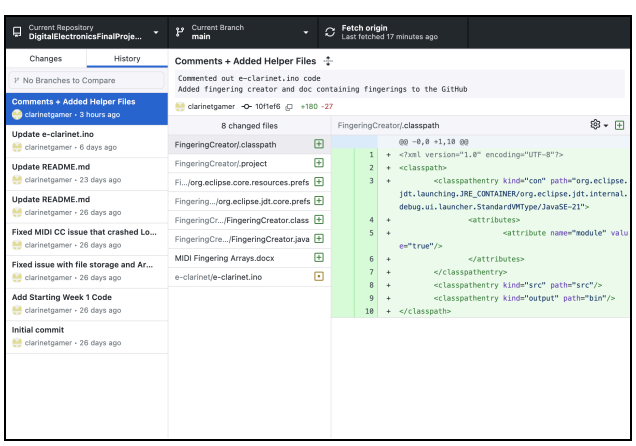
```
void getBreathVal() {
  float pressure = mpr.readPressure();


  int total = 0;
  int breath;
  for (int i = 0; i < 1000; i++) {
    breath = map(pressure, firstBreath, pressureMax, 0, 127);
    total += constrain(breath, 0, 127);
  }
  breathVal = total / 1000;
}

void controlCC() {
  usbMIDI.sendControlChange(7, breathVal, 1);
}
```

Explanation: The code on the left is what goes in the setup() function for the breath sensor. The first if statement with the mpr.begin() is what causes the breath sensor to calibrate itself and the firstBreath and pressureMax variables are how I keep track of the baseline pressure in the room so that the room doesn't affect the sensor readings. The getBreathVal() function gets the current pressure reading from the sensor **1000 times and gets the average reading**. I did this as I noticed the pressure sensor has random inconsistencies every hundred or so readings. This is also why I used the constraint function to catch random outliers. The map function is there to lock the pressure readings to the 128 possible MIDI numbers. The controlCC() function just takes the breath average from the last function and **sends that data to control the CC volume**.

Github and Other Documentation



In terms of problems I faced, **the biggest issue was that I ran out of time** to implement all the features I wanted to. Everything described above including the word doc I used to store the array and note values, my Java helper program, and my full Arduino code is included both in the appendix following this page and is available on my GitHub repo.

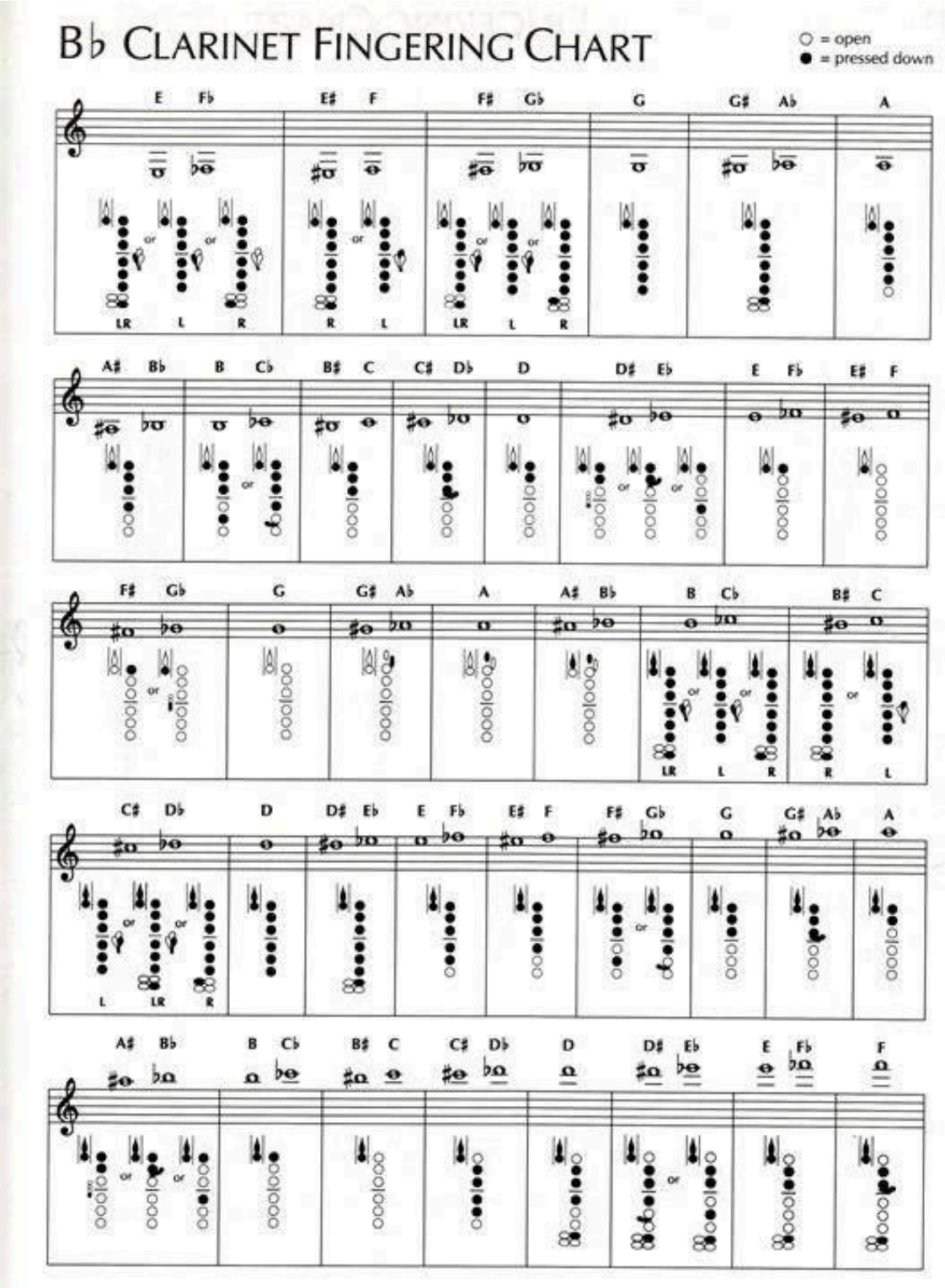
Fingering Creator Java Code

```java
package Main;
import java.util.Scanner;
public class FingeringCreator {
	// Back of instrument
	boolean registerKey = false;
	boolean thumbKey = false;
	// Left of Instrument
	boolean leftSideKey = false;
	// Top Half of Instrument
	boolean aKey = false;
	boolean oneKey = false;
	boolean twoKey = false;
	boolean bananaTopKey = false;
	boolean threeKey = false;
	boolean cSharpKey = false;
	boolean leftPinkyExt1 = false;
	boolean leftPinkyExt2 = false;
	boolean leftPinkyExt3 = false;
	// Right side of instrument
	boolean sideKey1 = false;
	boolean sideKey2 = false;
	boolean sideKey3 = false;
	boolean sideKey4 = false;
	// Bottom Half of Instrument
	boolean fourKey = false;
	boolean fiveKey = false;
	boolean bananaLowKey = false;
	boolean sixKey = false;
	boolean ebKey = false;
	boolean rightPinky1 = false;
	boolean rightPinky2 = false;
	boolean rightPinky3 = false;
	public boolean inputToBool(String print) {
		Scanner input = new Scanner(System.in);
		System.out.println(print+":");
		String s = input.nextLine();
		if (s.contains("t")) {
			return true;
		} else {
			return false;
		}
	}

	public void askUser() {
		registerKey = inputToBool("registerKey");
		thumbKey = inputToBool("thumbKey");
		// Left of Instrument
		leftSideKey = inputToBool("leftSideKey");
		// Top Half of Instrument
		aKey = inputToBool("aKey");
		oneKey = inputToBool("oneKey");
		twoKey = inputToBool("twoKey");
```

```
		bananaTopKey = inputToBool("bananaTopKey");
		threeKey = inputToBool("threeKey");
		cSharpKey = inputToBool("cSharpKey");
		leftPinkyExt1 = inputToBool("leftPinkyExt1");
		leftPinkyExt2 = inputToBool("leftPinkyExt2");
		leftPinkyExt3 = inputToBool("leftPinkyExt3");
		// Right side of instrument
		sideKey1 = inputToBool("sideKey1");
		sideKey2 = inputToBool("sideKey2");
		sideKey3 = inputToBool("sideKey3");
		sideKey4 = inputToBool("sideKey4");
		// Bottom Half of Instrument
		fourKey = inputToBool("fourKey");
		fiveKey = inputToBool("fiveKey");
		bananaLowKey = inputToBool("bananaLowKey");
		sixKey = inputToBool("sixKey");
		ebKey = inputToBool("ebKey");
		rightPinky1 = inputToBool("rightPinky1");
		rightPinky2 = inputToBool("rightPinky2");
		rightPinky3 = inputToBool("rightPinky3");
	}
	public boolean[] createFingeringArr() {
		return new boolean[] { registerKey, thumbKey, leftSideKey, aKey, oneKey, twoKey, bananaTopKey,
threeKey, cSharpKey, leftPinkyExt1, leftPinkyExt2, leftPinkyExt3, sideKey1, sideKey2, sideKey3, sideKey4, fourKey,
fiveKey, bananaLowKey, sixKey, ebKey, rightPinky1, rightPinky2, rightPinky3 };
	}
	public static void main(String[] args) {
		FingeringCreator fc = new FingeringCreator(); //Creates to FingeringCreator Object
		fc.askUser(); //Ask user input for t or f for whether button is pressed for that finger
		//Display array to be copied into doc
		System.out.print("{");
		boolean[] arr = fc.createFingeringArr();
		for(int i = 0; i < arr.length; i++) {
			System.out.print(arr[i]);
			if(i != arr.length-1) {
				System.out.print(",");
			}
		}
		System.out.print("},");
	}
}
```

Traditional Clarinet Fingering Chart



*This is the fingering chart I mostly based the Clar-E-net off of. I also added some fingerings I know of that were left off of this chart.

MIDI Fingerings Chart Doc

Index	Note	MIDI	Bool Array
0	E3	64,	{false,true,false,false,true,true,false,true,false,true,false,false,false,false,false,false,true,true,false,true,false,false,false,true},
1	E3	64,	{false,true,false,false,true,true,false,true,false,true,false,false,false,false,false,false,true,true,false,true,false,false,false,false} ,
2	E3	64,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,false,false,true,false},
3	F3	65,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,false,false,false,true},
4	F3	65,	{false,true,false,false,true,true,false,true,false,false,true,false,false,false,false,false,true,true,false,true,false,false,false,false},
5	F#3	66,	{false,true,false,false,true,true,false,true,false,false,false,true,false,false,false,false,true,true,false,true,false,false,false,true},
6	F#3	66,	{false,true,false,false,true,true,false,true,false,false,false,true,false,false,false,false,true,true,false,true,false,false,false,false},
7	F#3	66,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,false,true,false,false},
8	G3	67,	{ false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,false,false,false,false},
9	G#3	68,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,true,false,false,false},
10	A3	69,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
11	Bb3	70,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false},
12	B3	71,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false},
13	B3	71,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false},
14	C4	72,	{false,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
15	C#4	73,	{false,true,false,false,true,true,false,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
16	D4	74,	{false,true,false,false,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
17	Eb4	75,	{false,true,false,false,true,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
18	Eb4	75,	{false,true,false,false,true,true,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false},
19	Eb4	75,	{false,true,false,false,true,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false},
20	E4	76,	{false,true,false,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
21	F4	77,	{false,true,false},
22	F#4	78,	{false,false,false,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
23	F#4	78,	{false,true,false,false,false,false,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false,false,false},
24	G4	79,	{false,false},
25	G#4	80,	{false,false,true,false},
26	A4	81,	{false,false,false,true,false},
27	Bb4	82,	{true,false,false,true,false},
28	Bb4	82,	{false,false,false,true,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false,false,false},
29	B4	83,	{true,true,false,false,true,true,false,true,false,true,false,false,false,false,false,false,true,true,false,true,false,false,false,true},
30	B4	83,	{true,true,false,false,true,true,false,true,false,true,false,false,false,false,false,false,true,true,false,true,false,false,false,false},
31	B4	83,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,false,false,true,false},
32	C5	84,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,false,false,false,true},
33	C5	84,	{true,true,false,false,true,true,false,true,false,false,true,false,false,false,false,false,true,true,false,true,false,false,false,false},
34	C#5	85,	{true,true,false,false,true,true,false,true,false,false,false,true,false,false,false,false,true,true,false,true,false,false,false,false},
35	C#5	85,	{true,true,false,false,true,true,false,true,false,false,false,true,false,false,false,false,true,true,false,true,false,false,false,true},
36	C#5	85,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,false,true,false,false},
37	D5	86,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,false,false,false,false},
38	Eb5	87,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,true,true,false,false,false},

39	E5	88,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
40	F5	89,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false},
41	F#5	90,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false},
42	F#5	90,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,true,false,true,false,false,false,false,false},
43	G5	91,	{true,true,false,false,true,true,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
44	Ab5	92,	{true,true,false,false,true,true,false,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
45	A5	93,	{true,true,false,false,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
46	Bb5	94,	{true,true,false,false,true,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
47	Bb5	94,	{true,true,false,false,true,true,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false},
48	Bb5	94,	{true,true,false,false,true,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false},
49	Bb5	94,	{true,true,false,false,true,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false},
50	B5	95,	{true,true,false,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false},
51	C6	96,	{true,true,false},
52	C#6	97,	{true,true,false,false,false,true,false,true,false,false,false,false,false,false,false,false,true,true,false,false,false,false,false,false},
53	D6	98,	{true,true,false,false,false,true,false,true,false,false,false,false,false,false,false,false,true,false,false,false,true,false,false,false},
54	Eb6	99,	{true,true,false,false,false,true,false,true,false,false,false,false,false,false,false,false,true,false,true,false,true,false,false,false},
55	Eb6	99,	{true,true,false,false,false,true,false,true,false,false,false,false,false,false,false,false,true,false,false,true,false,false,false,false},
56	Eb6	99,	{true,true,false,false,false,true,false,true,false,false,false,false,false,false,false,false,false,false,true,true,false,false,false,false},
57	E6	100,	{true,true,false,false,false,true,false,true,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false},
58	F6	101,	{true,true,false,false,false,true,false,true,true,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false},
59	F6	101,	{true,true,false,false,true,true,false,true,true,false,false,false,false,false,false,false,true,true,false,true,false,false,false,false},
60	F#6	102,	{true,true,false,false,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false},
61	F#6	102,	{true,true,false,false,true,true,false,false,false,false,false,false,false,false,false,false,true,true,false,true,true,false,false,false},
62	G6	103,	{true,true,false,false,false,true,false,false,false,false,false,false,false,false,false,false,true,true,false,false,true,false,false,false},
63	G6	103,	{true,true,false,false,true,false,false,false,false,false,false,false,false,false,false,false,true,true,false,false,true,false,false,false},
64	G6	103,	{true,true,false,false,true,false,false,true,false,false,false,false,false,false,false,false,true,false,false,true,true,false,false,false},
65	G6	103,	{true,true,false,false,false,true,false,true,true,false,false,false,false,false,false,false,true,true,false,false,true,false,false,false},
66	Ab6	104,	{true,true,false,false,false,true,false,false,false,false,false,false,false,false,false,false,false,true,false,false,true,false,false,false},
67	Ab6	104,	{true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,true,false,false},
68	A6	105,	{true,true,false,false,false,true,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false},
69	A6	105,	{true,true,false,false,false,true,false,true,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false},
70	Bb6	106,	{true,true,true,false,true,true,false,true,true,false,false,false,false,false,false,false,true,true,false,true,false,false,false,true},
71	Bb6	106,	{true,true,true,false,false,true,false,true,true,false,false,false,false,false,false,false,false,false,false,false,true,false,false},
72	B6	107,	{true,true,true,false,true,true,false,false,false,false,false,false,false,false,false,false,true,true,false,false,false,true,false,false},
73	C7	108	{true,true,true,false,true,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,true,false,false}

Clar-E-net Arduino Code

```
//All the necessary things to import from breath sensor demo code
#include <Wire.h>
#include "Adafruit_MPRLS.h"

#define RESET_PIN -1
#define EOC_PIN -1
Adafruit_MPRLS mpr = Adafruit_MPRLS(RESET_PIN, EOC_PIN);
//The following array is the key on the clarinet represented by each button pin:
//{registerKey,thumbKey,leftSideKey,aKey,oneKey,twoKey,bananaTopKey,threeKey,cSharpKey,leftPinkyExt1,leftPinkyExt2,leftPinkyExt3,sideKey1,sideKey2,side
Key3,sideKey4,fourKey,fiveKey,bananaLowKey,sixKey,ebKey,rightPinky1,rightPinky2,rightPinky3}
int buttonPins[24] = { 23, 22, 20, 21, 16, 15, 14, 40, 39, 38, 33, 34, 36, 35, 37, 41, 32, 31, 30, 29, 28, 27, 25, 26 };

//This array holds the data of which keys are actively being pressed
bool activeFingering[24] = { false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false,
false };

//Holds every possible fingering - the indexes of the arrays correlate to the notes array that hold the notes for each fingering
bool fingering[74][24] = { { false, true, false, false, true, true, false, true, false, true, false, false, false, false, false, false, true, true, false, true, false, false, false, true },
        { false, true, false, false, true, true, false, true, false, true, false, false, false, false, false, false, true, true, false, true, false, false, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, false, false, true, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, false, false, false, true },
        { false, true, false, false, true, true, false, true, false, false, true, false, false, false, false, false, true, true, false, true, false, false, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, true, false, false, false, false, true, true, false, true, false, false, false, true },
        { false, true, false, false, true, true, false, true, false, false, false, true, false, false, false, false, true, true, false, true, false, false, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, false, true, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, false, false, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, true, false, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, false, false, false, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false },
        { false, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, true, false, false, true, true, false, true, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, true, false, false, true, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, true, false, false, true, true, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, true, false, false, true, true, false, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false, false, false },
        { false, true, false, false, true, false, false, false, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false, false },
        { false, true, false, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, false, false, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, true, false, false, false, false, false, false, false, false, false, false, false, true, true, false, false, false, false, false, false, false, false, false },
        { false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, false, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { true, false, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
        { false, false, false, true, false, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false, false, false, false, false },
        { true, true, false, false, true, true, false, true, false, true, false, false, false, false, false, false, true, true, false, true, false, false, false, true },
        { true, true, false, false, true, true, false, true, false, true, false, false, false, false, false, false, true, true, false, true, false, false, false, false },
        { true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, false, false, true, false },
        { true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, false, false, false, true },
        { true, true, false, false, true, true, false, true, false, false, true, false, false, false, false, false, true, true, false, true, false, false, false, false },
        { true, true, false, false, true, true, false, true, false, false, false, true, false, false, false, false, true, true, false, true, false, false, false, false },
        { true, true, false, false, true, true, false, true, false, false, false, true, false, false, false, false, true, true, false, true, false, false, false, true },
        { true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, false, true, false, false },
        { true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, false, false, false, false },
        { true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, true, true, false, false, false },
        { true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, true, false, false, false, false, false, false },
        { true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false, false },
```

```
{ true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false },
{ true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, false, true, false, true, false, false, false, false, false },
{ true, true, false, false, true, true, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
{ true, true, false, false, true, true, false, true, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
{ true, true, false, false, true, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
{ true, true, false, false, true, true, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false },
{ true, true, false, false, true, true, false, false, false, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false, false },
{ true, true, false, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false, false },
{ true, true, false, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, true, false, false, false, false, false, false, false },
{ true, true, false, false, true, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false
```

```
void loop() {
 createFingerArray(); //This updates the activeFingering[]
 playNote(); //Plays note based on current fingering
 getBreathVal(); //Reads average value from pressure sensor from 1000 different readings
 if (millis() % 50 == 0) {
  controlCC(); //Sends pressure sensor data to control CC Volume
 }
}

void getBreathVal() {
 float pressure = mpr.readPressure(); //Get breath data

 //Average and map breath data to MIDI values
 int total = 0;
 int breath;
 for (int i = 0; i < 1000; i++) {
  breath = map(pressure, firstBreath, pressureMax, 0, 127);
  total += constrain(breath, 0, 127);
 }
 breathVal = total / 1000; //Set breathVal variable to be used elsewhere
}

void controlCC() {
 usbMIDI.sendControlChange(7, breathVal, 1);  //Controls MIDI Volume based on breath readings
}

void playNote() {
 prevNote = currentNote;
 currentNote = getNote(); // Check to see if note has changed
 if (currentNote != prevNote) {   //If note has changed, turn off previous note and start new note. This only sends the MIDI data when there is a change instead
of constantly sending data
  usbMIDI.sendNoteOff(prevNote, 127, 1);
  usbMIDI.sendNoteOn(currentNote, 127, 1);
 }
}

void createFingerArray() {
 for (int i = 0; i < 24; i++) {
  activeFingering[i] = digitalRead(buttonPins[i]); //Updates activeFingering[] based on current button readings
 }
 delay(5);
}

int getNote() {
 for (int i = 0; i < 74; i++) { //Checks all 74 fingering arrays to compare
  bool boolean = true;
  for (int j = 0; j < 24; j++) { //Checks every position to see if activeFingering matches that array
   if (activeFingering[j] != fingering[i][j]) {
    boolean = false;
    continue;// If activeFingering isn't the fingering continue
   }
   if (j == 23 and boolean == true) {
    return notes[i]-2; //If fingerings match and a previous button has not been marked a false return the note associated with that array
   }
  }
 }
 return 77;  //open G if fingering doesn't exist
}
```