**Dylan Ou** :

Net ID: dou5

E-mail ID: [dou5@uic.edu](mailto:dou5@uic.edu)

**Rohan Verma** :

Net ID: rverma26

E-mail ID: [rverma26@uic.edu](mailto:rverma26@uic.edu)

**Project Idea:** Arduino MIDI Piano and Controller

**Abstract:**

Carrying around an actual music instrument is impractical and not always possible. This project essentially makes recording and playing music on-the-go easier. The piano and controller are one MIDI device that can be plugged into a computer to be used to produce and play music in any music producing software. MIDI (Musical Instrument Digital Interface) is a way to connect devices that make and control sound so that they can communicate with other devices. With this device, a musician can send MIDI information to a computer in order to play whatever the user wants.

**Overall Description:**

Our project is a MIDI piano and controller. MIDI devices are used to connect musical instruments to any device that can read MIDI information. Such devices are used to record and perform music into a digital audio workstation (DAW) such as Logic, Ableton, Pro Tools, Fl Studio, etc. Our project consisted of buttons to play notes on a piano, four potentiometers, and a joystick to control different parameters within a DAW the user chooses to use. Every button, potentiometer and joystick sends MIDI information to the computer using the serial monitor. One of the initial problems with our project was that the Arduino Uno and Mega do not have native USB ports, meaning that our Arduinos cannot be seen as external devices like a mouse and keyboard.

To bypass this problem, there are several programs we needed to download and run alongside our MIDI device. The first program is HairlessMIDI, which converts information from the serial monitor and converts it into MIDI. The second program is loopMIDI, which creates a virtual MIDI device that uses the MIDI information from the first program so any DAW can see a device is connected and ready to use for recording. All of the buttons, potentiometers, and joystick are connected to the Arduino Mega. It is communicating with an Arduino Uno that is using an LCD screen to display the note the player is using.

**Project Design:**

The MIDI piano and controller mainly has two outputs, which are the USB port to connect to a computer and the LCD screen to display a note that the user is playing. Proper MIDI devices have a USB output and a proper MIDI output, which is a connector with five different pins. Both receive information from a serial monitor and that is read by a DAW to use. For the sake of this project, we just used the USB port. Regarding the LCD screen, the two Arduinos are communicating via serial communication. Addressing the problem of needing multiple serial ports and digital pins for this project, an Arduino Mega needed to be used. An Arduino Mega has a total of 16 analog pins, 54 I/O pins, and 10 communication pins, allowing up to five different serial monitors. The Arduino UNO is then used to display the notes via an LCD screen.

The rest of the project consists of inputs. The twelve buttons are used to play notes, but twelve notes are not enough to play an instrument normally. To make up for this, the user can change whether they want lower notes or higher notes. Therefore, the piano can play a total of 72 notes. The potentiometers can change four different potentiometers that can be assigned by the user. The joystick allows XY-modulation of two different parameters.

**Timeline (20th FEBRUARY - 12th APRIL):**

**Week 1:** Getting individual parts

**Week 2:** Assembling the LCD and piano circuit.

**Week 3:** Working on arduino code for the piano circuit and writing code for the LCD circuit.

**Week 4:** Getting the rotating knobs for the circuit. Working on arduino code for the piano circuit.

**Week 5:** Using the 2 buttons to change the scale of the piano. Setting up the 2nd arduino for the knobs and the LCD display.

**Week 6:** working on the code for the LCD and the joystick and the 4 knobs.

**Week 7:** Code for the LCD, joystick and the 4 knobs. Code for making the arduinos communicate.

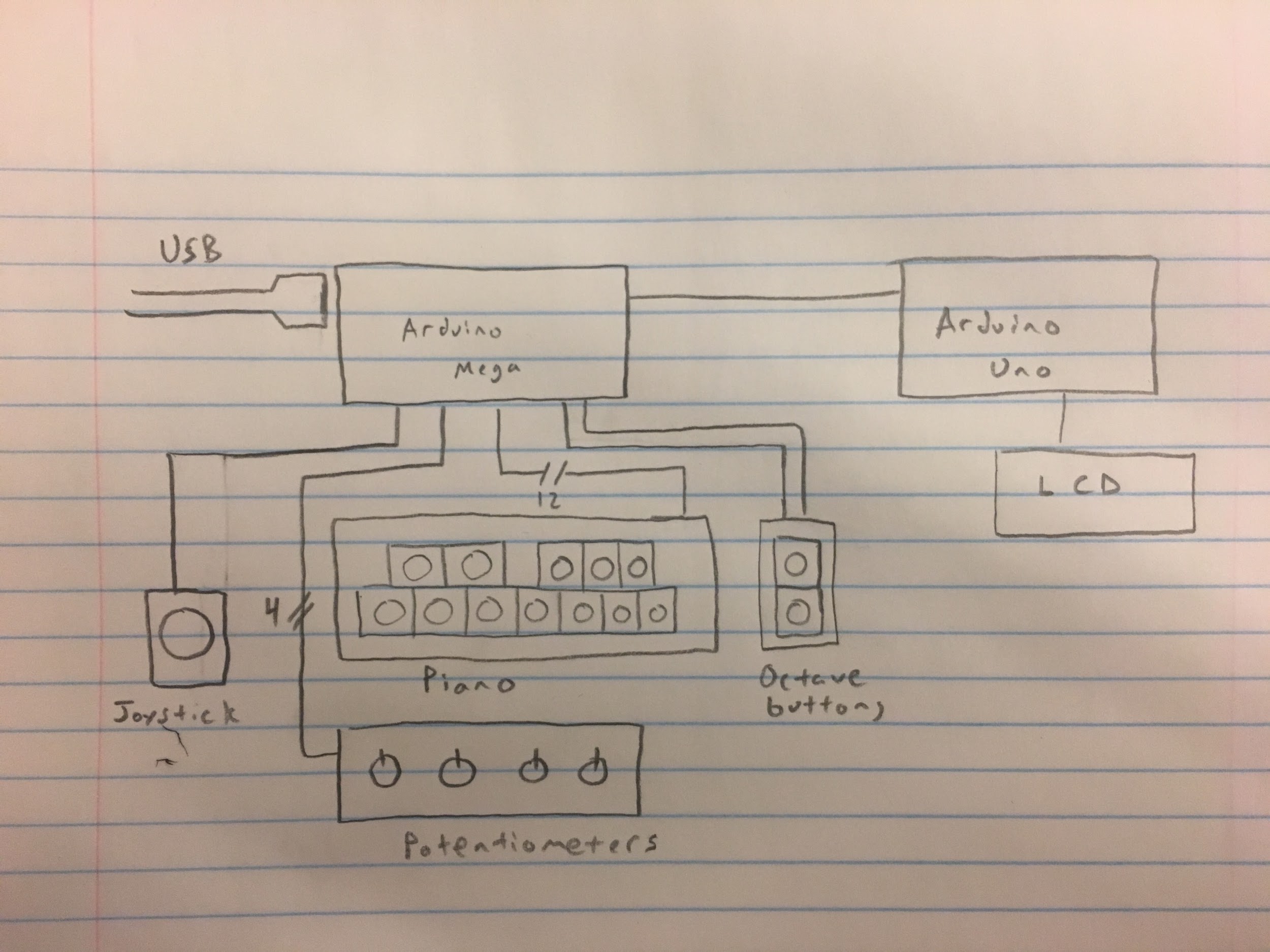
**Week 8:**  Testing the setup and demonstrating if completed.

**Week 9:** Demonstration.

**List Of References:**

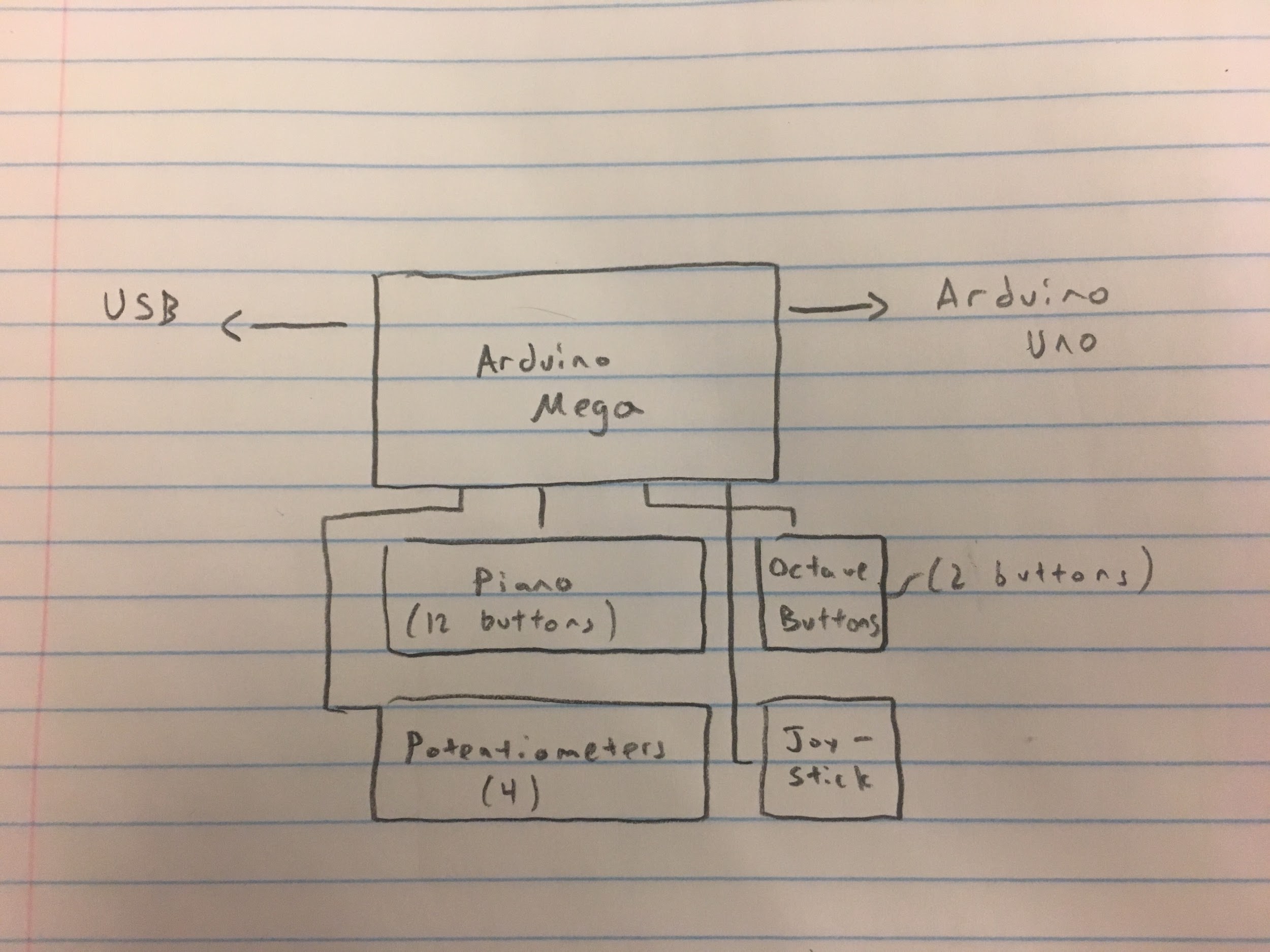
1. Youtube link from where we got our Idea: <https://www.youtube.com/watch?v=4sopfrr1830>
2. A brief description of the MIDI controller: <https://www.instructables.com/id/Arduino-MIDI-Controller/>
3. Link for the list of materials needed to build the controller: <https://www.musiconerd.com/single-post/Build-this-MIDI-controller-Fliper-Dj>
4. Link for potential tutorial videos which can be used: <https://www.makeuseof.com/tag/arduino-midi-controllers/>
5. Another tutorial link: <https://www.makeuseof.com/tag/make-midi-controller-arduino/>
6. An official Arduino tutorial: <https://www.arduino.cc/en/Tutorial/MidiDevice>
7. A tutorial on Github: <https://github.com/tttapa/MIDI_controller/blob/master/README.md>
8. Hairless MIDI (used to read the serial inputs from the Arduinos to convert into MIDI information): <http://projectgus.github.io/hairless-midiserial/>
9. loopMIDI (used to convert the MIDI information from Hairless MIDI to be used in a Digital Audio Workstation): <https://www.tobias-erichsen.de/software/loopmidi.html>
10. Arduino MIDI library (reference for the user to understand the MIDI functions used in the code): <http://fortyseveneffects.github.io/arduino_midi_library/index.html>

**Sketch:**

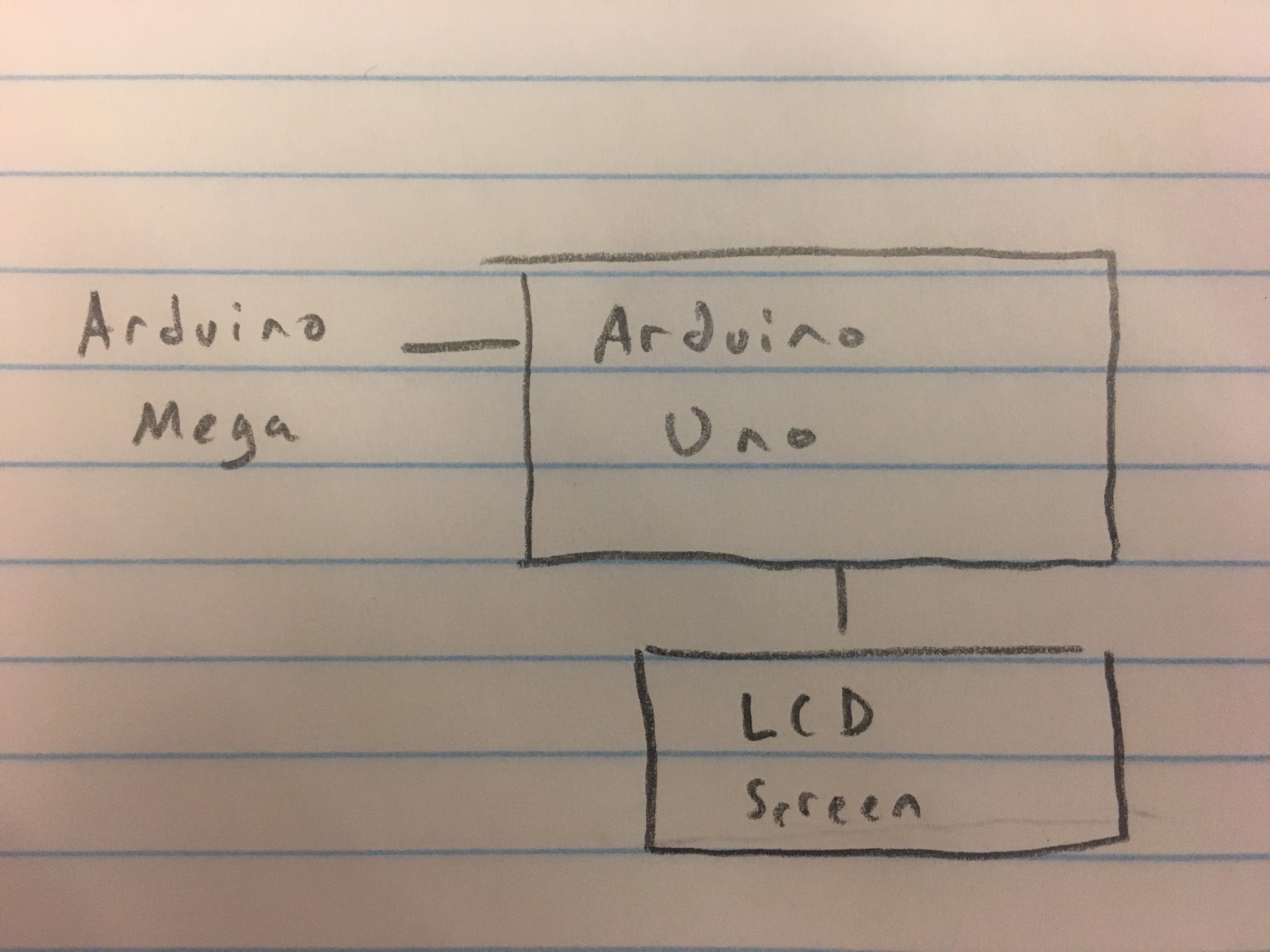
****

**Individual Sketches of the Arduinos:**

**Arduino Mega**

****

**Arduino Uno**

****

**List Of Materials Needed:**

* Arduino Uno
* Arduino Mega
* 4 10k potentiometers
* 15 x 220 ohm resistors
* 2 x 10k ohm resistors
* 14 x momentary switches (12 for piano keys, 2 for changing scales)
* Wires
* 4 breadboards
* Cable to connect to the computer
* LCD screen
* Joystick
* Hairless MIDI (program)
* loopMIDI (program)

**Original Work:**

We connected an LCD and joystick with our device. The LCD displays the note that the user is using. The joystick allows the user to customize XY-control of two different parameters in a DAW

**Code:**

**Code for Arduino Mega (main instrument):**

// Group 53

//

// Dylan Ou - dou5

// Rohan Verma - rverma26

//

// Project: Arduino MIDI piano and controller

//

// Abstract:

// Our project is a piano that can be plugged into a computer and be able to play

// notes in music making software. The piano includes several knobs and a joystick to

// control effects. The LCD screen will display the value of the controller being affected.

// This device can be used for live performances, recording, etc.

#include <MIDI.h>

#include "pitches.h"

#define NUM\_BUTTONS 12

#define THRESHOLD 5

#define NOTE\_LENGTH 50

#define VEL\_SENSE\_VAL 2

const int buttonPin1 = 52; // the number of the pushbutton pin

const int buttonPin2 = 51; // the number of the pushbutton pin

const int buttonPin3 = 48; // the number of the pushbutton pin

const int buttonPin4 = 46; // the number of the pushbutton pin

const int buttonPin5 = 44; // the number of the pushbutton pin

const int buttonPin6 = 42; // the number of the pushbutton pin

const int buttonPin7 = 40; // the number of the pushbutton pin

const int buttonPin8 = 38; // the number of the pushbutton pin

const int buttonPin9 = 36; // the number of the pushbutton pin

const int buttonPin10 = 34; // the number of the pushbutton pin

const int buttonPin11 = 32; // the number of the pushbutton pin

const int buttonPin12 = 30;

int val = 0;

int lastVal = 0;

int val2 = 0;

int lastVal2 = 0;

int val3 = 0;

int lastVal3 = 0;

int val4 = 0;

int lastVal4 = 0;

int val5 = 0;

int lastVal5 = 0;

int val6 = 0;

int lastVal6 = 0;

const int buttonDown = 8;

const int buttonUp = 9;

const int buttons[NUM\_BUTTONS] = {buttonPin1, buttonPin2, buttonPin3,

buttonPin4, buttonPin5, buttonPin6, buttonPin7, buttonPin8, buttonPin9,

buttonPin10, buttonPin11, buttonPin12

};

int buttonRead[NUM\_BUTTONS];

int notes[] = {

NOTE\_C2, NOTE\_CS2, NOTE\_D2, NOTE\_DS2, NOTE\_E2, NOTE\_F2, NOTE\_FS2, NOTE\_G2, NOTE\_GS2, NOTE\_A2, NOTE\_AS2, NOTE\_B2,

NOTE\_C3, NOTE\_CS3, NOTE\_D3, NOTE\_DS3, NOTE\_E3, NOTE\_F3, NOTE\_FS3, NOTE\_G3, NOTE\_GS3, NOTE\_A3, NOTE\_AS3, NOTE\_B3,

NOTE\_C4, NOTE\_CS4, NOTE\_D4, NOTE\_DS4, NOTE\_E4, NOTE\_F4, NOTE\_FS4, NOTE\_G4, NOTE\_GS4, NOTE\_A4, NOTE\_AS4, NOTE\_B4,

NOTE\_C5, NOTE\_CS5, NOTE\_D5, NOTE\_DS5, NOTE\_E5, NOTE\_F5, NOTE\_FS5, NOTE\_G5, NOTE\_GS5, NOTE\_A5, NOTE\_AS5, NOTE\_B5,

NOTE\_C6, NOTE\_CS6, NOTE\_D6, NOTE\_DS6, NOTE\_E6, NOTE\_F6, NOTE\_FS6, NOTE\_G6, NOTE\_GS6, NOTE\_A6, NOTE\_AS6, NOTE\_B6,

NOTE\_C7, NOTE\_CS7, NOTE\_D7, NOTE\_DS7, NOTE\_E7, NOTE\_F7, NOTE\_FS7, NOTE\_G7, NOTE\_GS7, NOTE\_A7, NOTE\_AS7, NOTE\_B7

};

String noteNames[] = {

"C ", "C#", "D ", "D#", "E ", "F ", "F#", "G ", "G#", "A ", "A#", "B "

};

int keyStart = 24;

struct MySettings : public midi::DefaultSettings

{

static const bool UseRunningStatus = false;

static const long BaudRate = 115200;

};

MIDI\_CREATE\_CUSTOM\_INSTANCE(HardwareSerial, Serial, MIDI, MySettings);

const int velocity = 127; //Max Velocity (range is 0-127)

const int channel = 1; //MIDI Channel 1 (out of 16)

bool noteIsOn[NUM\_BUTTONS] = {false};

int midiNoteTime[NUM\_BUTTONS] = {0};

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

Serial1.begin(9600);

for (int i = 0; i < NUM\_BUTTONS; i++)

pinMode(buttons[i], INPUT\_PULLUP);

pinMode(buttonUp, INPUT\_PULLUP);

pinMode(buttonDown, INPUT\_PULLUP);

MIDI.begin();

}

void loop() {

// put your main code here, to run repeatedly:

int buttonStateUp = digitalRead(buttonUp);

int buttonStateDown = digitalRead(buttonDown);

val = analogRead(0) / 8; // Divide by 8 to get range of 0-127 for midi

if (val != lastVal) // If the value does not = the last value the following command is made. This is because the pot has been turned. Otherwise the pot remains the same and no midi message is output.

{

MIDI.sendControlChange(16, val, 1);

} // 176 = CC command (channel 1 control change), 1 = Which Control, val = value read from Potentionmeter 1 NOTE THIS SAYS VAL not VA1 (lowercase of course)

lastVal = val;

val2 = analogRead(2) / 8; // Divide by 8 to get range of 0-127 for midi

if (val2 != lastVal2)

{

MIDI.sendControlChange(17, val2, 2);

} // 176 = CC command, 2 = Which Control, val = value read from Potentionmeter 2

lastVal2 = val2;

val3 = analogRead(4) / 8; // Divide by 8 to get range of 0-127 for midi

if (val3 != lastVal3)

{

MIDI.sendControlChange(18, val3, 3);

} // 176 = CC command, 3 = Which Control, val = value read from Potentionmeter 3

lastVal3 = val3;

val4 = analogRead(6) / 8; // Divide by 8 to get range of 0-127 for midi

if (val4 != lastVal4)

{

MIDI.sendControlChange(19, val4, 4);

} // 176 = CC command, 3 = Which Control, val = value read from Potentionmeter 3

lastVal4 = val4;

val5 = analogRead(8) / 8; // Divide by 8 to get range of 0-127 for midi

if (val5 != lastVal5)

{

MIDI.sendControlChange(20, val5, 5);

} // 176 = CC command, 3 = Which Control, val = value read from Potentionmeter 3

lastVal5 = val5;

val6 = analogRead(10) / 8; // Divide by 8 to get range of 0-127 for midi

if (val6 != lastVal6)

{

MIDI.sendControlChange(21, val6, 6);

} // 176 = CC command, 3 = Which Control, val = value read from Potentionmeter 3

lastVal6 = val6;

//delay(15); //here we add a short delay to help prevent slight fluctuations, knocks on the pots etc. Adding this helped to prevent my pots from jumpin up or down a value when slightly touched or knocked.

if (buttonStateUp == HIGH) {

keyStart = keyStart + 12;

if (keyStart > 60) {

keyStart = 60;

}

delay(250);

}

if (buttonStateDown == HIGH) {

keyStart = keyStart - 12;

if (keyStart < 0) {

keyStart = 0;

}

delay(250);

}

for (int i = 0; i < NUM\_BUTTONS; i++) {

buttonRead[i] = digitalRead(buttons[i]);

if (buttonRead[i] == HIGH && noteIsOn[i] == false) {

MIDI.sendNoteOn(notes[keyStart + i], velocity, channel);

noteIsOn[i] = true;

midiNoteTime[i] = NOTE\_LENGTH;

}

if (noteIsOn[i] == true) {

Serial1.print(noteNames[i]);

Serial1.print("\n");

if (buttonRead[i] == LOW) {

MIDI.sendNoteOff(notes[keyStart + i], 0, channel);

noteIsOn[i] = false;

}

}

}

//pause the loop

delay(5);

}

**Code for Arduino Uno (LCD screen):**

#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

String a;

char mystr[2]; //Initialized variable to store recieved data

void setup() {

// Begin the Serial at 9600 Baud

Serial.begin(9600);

lcd.begin(16,2);

lcd.clear();

}

void loop() {

//Print data on Serial Monitor

if (Serial.available() > 0) {

Serial.readBytesUntil('\n', mystr, 2);

}

lcd.clear();

lcd.setCursor(0,0);

lcd.print(mystr[0]);

lcd.setCursor(1,0);

lcd.print(mystr[1]);

delay(5);

}

**pitches.h (defines MIDI number of note being played):**

#ifndef pitches

#define pitches

#define NOTE\_C2 36

#define NOTE\_CS2 37

#define NOTE\_D2 38

#define NOTE\_DS2 39

#define NOTE\_E2 40

#define NOTE\_F2 41

#define NOTE\_FS2 42

#define NOTE\_G2 43

#define NOTE\_GS2 44

#define NOTE\_A2 45

#define NOTE\_AS2 46

#define NOTE\_B2 47

#define NOTE\_C3 48

#define NOTE\_CS3 49

#define NOTE\_D3 50

#define NOTE\_DS3 51

#define NOTE\_E3 52

#define NOTE\_F3 53

#define NOTE\_FS3 54

#define NOTE\_G3 55

#define NOTE\_GS3 56

#define NOTE\_A3 57

#define NOTE\_AS3 58

#define NOTE\_B3 59

#define NOTE\_C4 60

#define NOTE\_CS4 61

#define NOTE\_D4 62

#define NOTE\_DS4 63

#define NOTE\_E4 64

#define NOTE\_F4 65

#define NOTE\_FS4 66

#define NOTE\_G4 67

#define NOTE\_GS4 68

#define NOTE\_A4 69

#define NOTE\_AS4 70

#define NOTE\_B4 71

#define NOTE\_C5 72

#define NOTE\_CS5 73

#define NOTE\_D5 74

#define NOTE\_DS5 75

#define NOTE\_E5 76

#define NOTE\_F5 77

#define NOTE\_FS5 78

#define NOTE\_G5 79

#define NOTE\_GS5 80

#define NOTE\_A5 81

#define NOTE\_AS5 82

#define NOTE\_B5 83

#define NOTE\_C6 84

#define NOTE\_CS6 85

#define NOTE\_D6 86

#define NOTE\_DS6 87

#define NOTE\_E6 88

#define NOTE\_F6 89

#define NOTE\_FS6 90

#define NOTE\_G6 91

#define NOTE\_GS6 92

#define NOTE\_A6 93

#define NOTE\_AS6 94

#define NOTE\_B6 95

#define NOTE\_C7 96

#define NOTE\_CS7 97

#define NOTE\_D7 98

#define NOTE\_DS7 99

#define NOTE\_E7 100

#define NOTE\_F7 101

#define NOTE\_FS7 102

#define NOTE\_G7 103

#define NOTE\_GS7 104

#define NOTE\_A7 105

#define NOTE\_AS7 106

#define NOTE\_B7 107

#endif