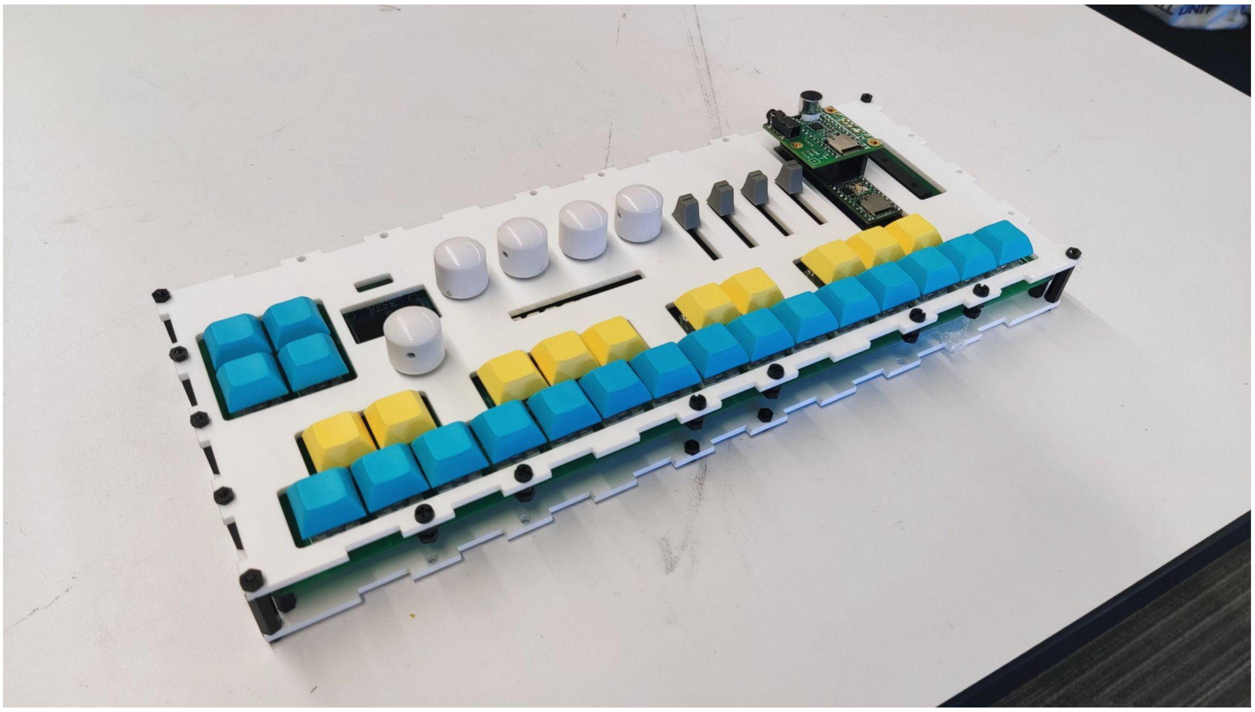
TeensyLab

– a reproduction of Arturis's portable midi controller MiniLab –



Overview:

The whole idea is to recreate Arturia's minilab with Teensy.
It is always a joyful thing to build something by myself, so does this.

There are a few reasons behind doing this project:

1. I want to have something I can use beyond the class and possibly for future academic/personal use. So it would be better to place the electronics in a better container rather than just breadboards.
   Plus, I want to compensate my analog project which did not achieve this goal. Thanks for Steve's instructions and his Fall 2022 Synth template project, I was able to build a large project (39 out of 42 pins on teensy 4.1 were used), and went through the product design process.
2. I bought MiniLab-2 when needing a controller to use in EMP during September Fall 2022. Unfortunately, after a few month, miniLab-3 came out. I was a little sad to see Arturia have lots of new features added to 3 that 2 did not have.
   So, why not I try to recreate miniLab-3 for digital project?

3. I am going to Prague for the next Fall, so there is no chance that I can take product design in New York.

Project Cost:(sadly it cost a lot…)
Links to special parts are linked below
$52.65 Digital Material (Teensy + audio shield + electronics from Steve)
$44.89 PCB:
    $19 for manufacture + $30 express world-wide shipping - $5 first order discount
    PCB Prototype & PCB Fabrication Manufacturer - JLCPCB for ordering
$33.00 Acrylic sheet:
    Bought at Canal Plastic. I bought for different colors, so maybe maybe $20 will be enough. My box is slightly wider than 12 inch, so I have to buy 18 inch size board.
    Find Acrylic Panels And Plexiglass In NYC – Canal Plastics Center
$66.88 KeySwitch + KeyCaps + 35mm push fader pot from Adafruit
    Slide Potentiometer with Plastic Knob - 35mm Long - 10KΩ : ID 4271 : $1.95 : Adafruit Industries, Unique & fun DIY electronics and kits
    Kailh Mechanical Key Switches - Clicky White - 10 pack [Cherry MX White Compatible] : ID 4955 : $6.95 : Adafruit Industries, Unique & fun DIY electronics and kits
    Black DSA Keycaps for MX Compatible Switches - 10 pack : ID 4997 : $5.95 : Adafruit Industries, Unique & fun DIY electronics and kits

Around $200 in total

Video Demostration:
https://youtu.be/NRasdBVcbtc

See All Files on Github:
ZYFXzz/Final_Project_miniLab: minilab recreate (github.com)
Includes the Arduino File, Kicad PCB file, illustrator laser-cut file

Explanation & Breakdown:

New Arduino IDE 2.0 came out during the middle of this semester, I switch to the new version for final project development.

Part A, Electronics and Program

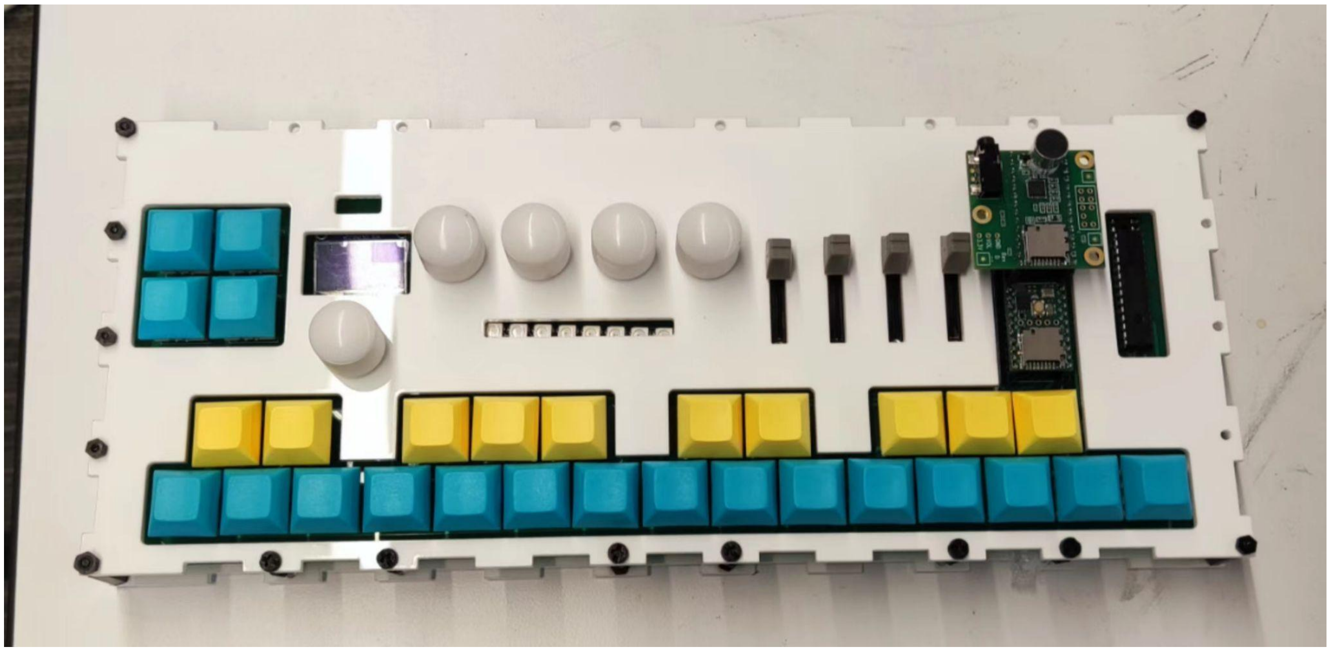
- Circuit and Electronics
- Mode 1: midi keyboard
- Mode 2: drum sequencer

Part B, Product Design

- PCB design/assembly
- Laser-cut box design/assembly

A:

Component Overview:



Compare to MiniLab from Arturia, the TeensyLab lacks the following:

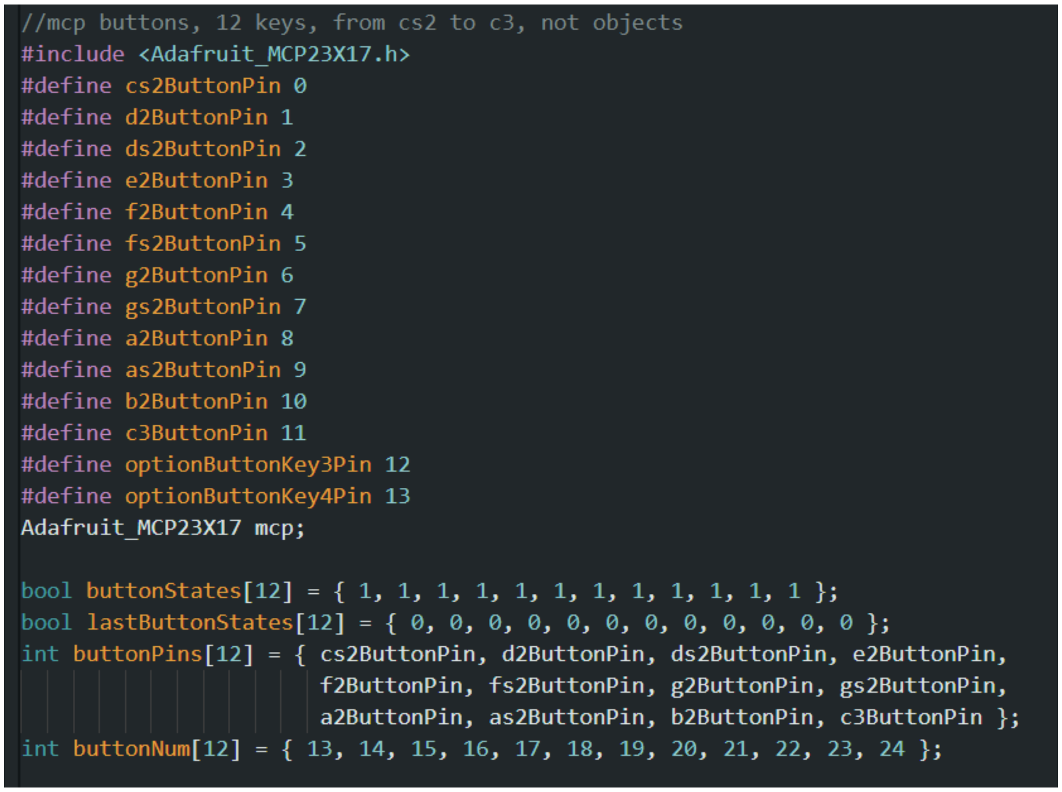
- Force-sensitive drum pads
- Touch-sensitive vibrato and mod control, I wish to do this but did not have enough time.
  Check out the SoftPot Potentiometer at Sparkfun:
  Soft Potentiometers - SparkFun Electronics 200mm, 50mm, 500mm,
  Sparkfun does not have 100mm softpot, look for it at somewhere else
  And Awesome Tutorial I looked at:
  Let's Create a Ribbon Synth! (Arduino + Max/MSP Or Pure Data) - YouTube

It is the same as the normal pots, but instead of turning the top, you turn by sliding your fingers on it.

- And some cool feature in software side:
  Like custom script for better communication with DAWs
  Like arppegiator mode in new MiniLab3, hope I can have some time later to figure the arp mode and chord mode.
  Etc, in a software perspective.

Else, the Teensy Lab has almost the same layout as the MiniLabs:

- 4 optional keyswitches, 2 button Objects, 2 mcp normal button
- 25 key switches: (do not expect have actual piano keys, most manufacturers has their own custom production line for their own keybed, so we use buttons/switches instead)
  - 13 switches, created as objects using Button Class, C1 -> C2
  - 12 switches: from Cs2 to C3
     hard coded, because it is hard to use MCP23017 class inside Button Class. not objects,
    Created in the way like early this semester when OOP was not introduced



```
//mcp buttons, 12 keys, from cs2 to c3, not objects
#include <Adafruit_MCP23X17.h>
#define cs2ButtonPin 0
#define d2ButtonPin 1
#define ds2ButtonPin 2
#define e2ButtonPin 3
#define f2ButtonPin 4
#define fs2ButtonPin 5
#define g2ButtonPin 6
#define gs2ButtonPin 7
#define a2ButtonPin 8
#define as2ButtonPin 9
#define b2ButtonPin 10
#define c3ButtonPin 11
#define optionButtonKey3Pin 12
#define optionButtonKey4Pin 13
Adafruit_MCP23X17 mcp;

bool buttonStates[12] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
bool lastButtonStates[12] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
int buttonPins[12] = { cs2ButtonPin, d2ButtonPin, ds2ButtonPin, e2ButtonPin,
                       f2ButtonPin, fs2ButtonPin, g2ButtonPin, gs2ButtonPin,
                       a2ButtonPin, as2ButtonPin, b2ButtonPin, c3ButtonPin };
int buttonNum[12] = { 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24 };
```

  - MCP23017: Steve gave me this IC, which he intended to give someone in the product design class, but seems no one used it.
    I was able to make use of this IC and its class. It is not hard to set up, you can easily find the datasheet and code example. But this do require some

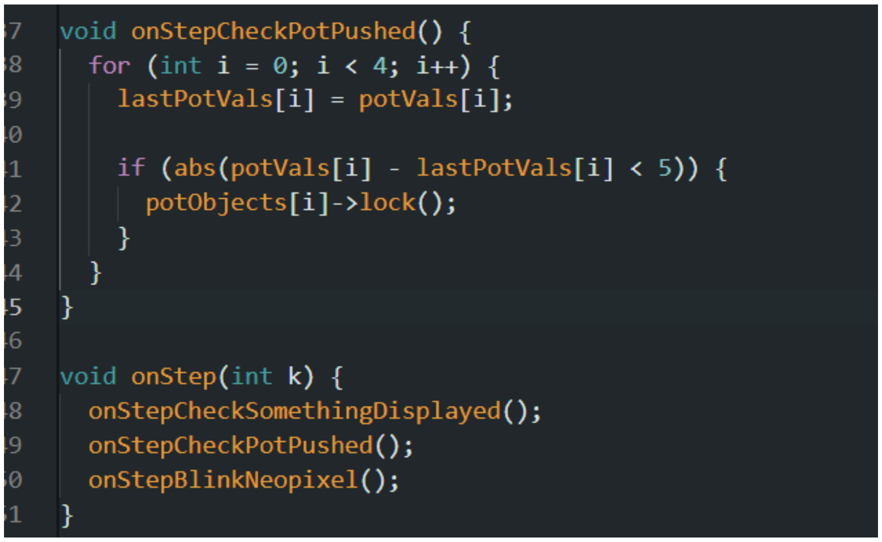
time to get things working as you want. You need to read through the code example, and the datasheet for the IC.
Need resistor connecting to power, on both SCL/SDA bus, see in video on product page of 23017 chip on Adafruit
MCP23017 - i2c 16 input/output port expander : ID 732 : $6.95 : Adafruit Industries, Unique & fun DIY electronics and kits
MCP has only digital Ports, so if you want to have like 25 keys and some analog pots like me, you need to plan where you want to place them.
Only teensy board are able to read analog voltage, so you definitely want to save the specific analog ports on teensy for potentiometers.
It can be hard to create objects on MCP, it is more advanced coding stuff which we are not covering in our class.
You probably can only use MCP for button input and light up extra LED groups.

- 5 Rotary Encoder, 1 Master + 4 Control Encoder:
  The encoder left/right action, was created as object, encoder(leftPin,rightPin)
  The encoder Button, was created as object, using the Button Class
  The encoders can take up 3 ports, I have 5, so that is why it ends up using 39 ports on teensy.
- Faders(pots): same as 10k blue b pot, but pushing instead of turning.
  Still, created using pot Class from Steve
  Slide Potentiometer with Plastic Knob - 35mm Long - 10KΩ : ID 4271 : $1.95 : Adafruit Industries, Unique & fun DIY electronics and kits
- SSD1306 display, standard display used in class, created using Class from Adafruit Library, got this from Steve
- Adafruit Neopixel Stick(*8 pixel): same as Neopixel used in Lecture Assignment, easy to set up once you understand about it, also a cool thing from Steve's stock

Notes:

- Encoders, buttons, and faders, are put into group of their individual objects types, Later in the loop(), the process() for each object is called using pointers, a more advanced stuff in C++. We can simply put it as a symbol pointing to something.
  I basically followed Steve's example, and did little modifications to make it fit my project.
  To create all these objects, all you need to do is take some time to read through the header files of these classes, just like what we were doing in the OOP assignment. And understand what the pointers are doing.
- As we know, the readings of pot is not stable as it is reading analog electricity/voltage, instead of On/Off in digital

If you use Serial.println() for displaying the analogRead() for a pot pin, it is likely the prints will takes up all your serial monitor, flashing 1021, 1022,1023 on the screen, even after you stop moving it.
In daw, when you map the fader hardware to Daw fader, the daw fader will be shaky.
The way I fixed it, is by using the Stepper class, for every 250 milliseconds, on

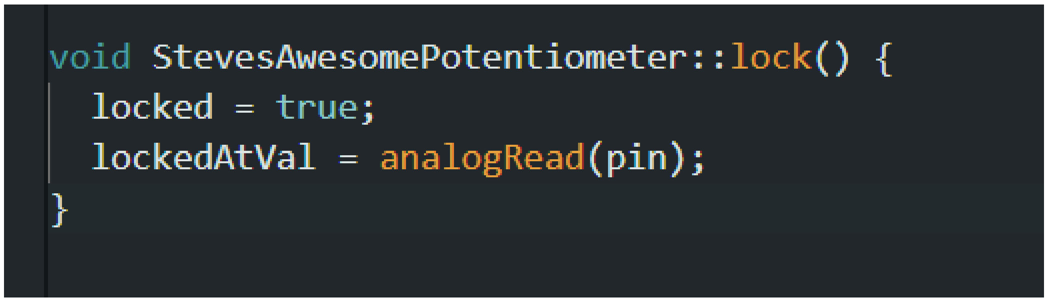


```
void onStepCheckPotPushed() {
  for (int i = 0; i < 4; i++) {
    lastPotVals[i] = potVals[i];

    if (abs(potVals[i] - lastPotVals[i] < 5)) {
      potObjects[i]->lock();
    }
  }
}

void onStep(int k) {
  onStepCheckSomethingDisplayed();
  onStepCheckPotPushed();
  onStepBlinkNeopixel();
}
```

step it will check if the last Pot value has changed only a little compared to the current value. The window of 0.25 seconds is good enough to provide a smooth feeling when using the faders. The readings will stop real quick after you take your hands away from it.
It is similar to checking the buttonStates and lastButtonStates for Buttons.
The lock():

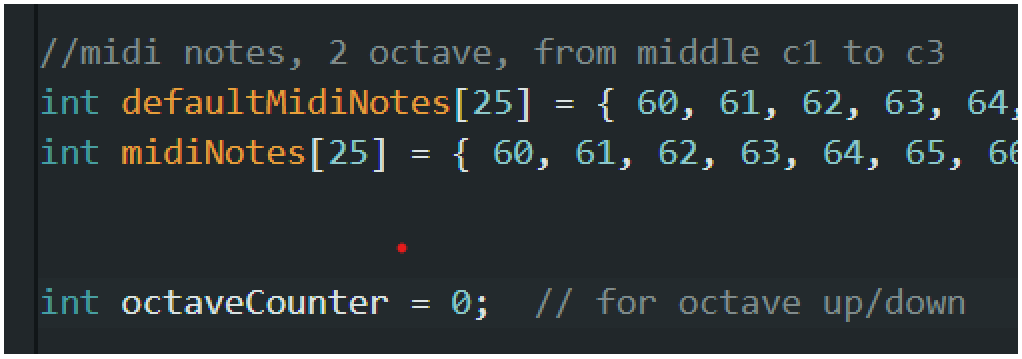


```
void StevesAwesomePotentiometer::lock() {
  locked = true;
  lockedAtVal = analogRead(pin);
}
```

It will fix the pot reading with lockedAtVal.

The Piano Keyboard Mode:

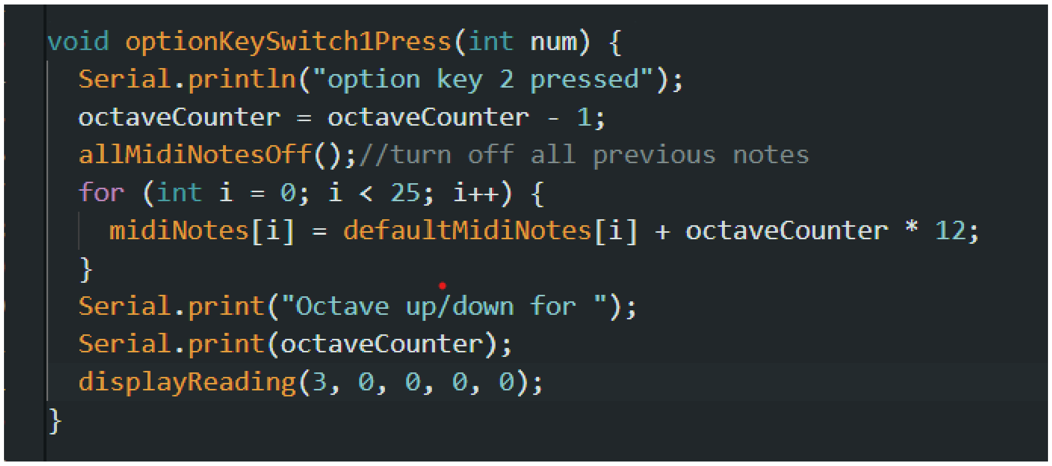
- It is fairly simple like the week3 lab, but with more buttons.
  It can switch to different octave you want:



```
//midi notes, 2 octave, from middle c1 to c3
int defaultMidiNotes[25] = { 60, 61, 62, 63, 64,
int midiNotes[25] = { 60, 61, 62, 63, 64, 65, 66


int octaveCounter = 0;  // for octave up/down
```

Below is how I modify the notes array:

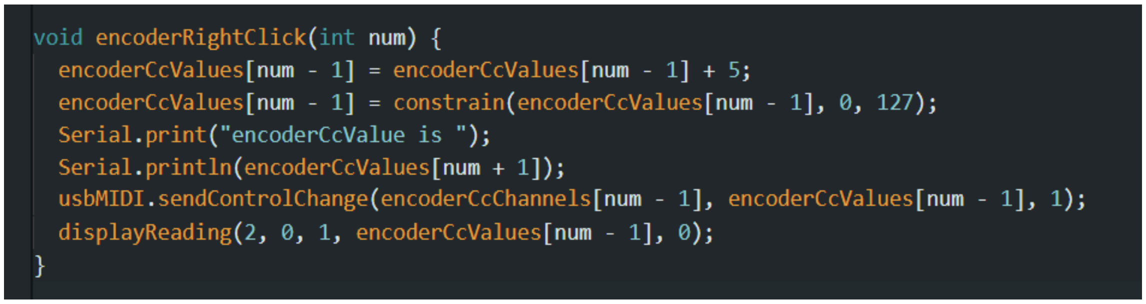


```
void optionKeySwitch1Press(int num) {
  Serial.println("option key 2 pressed");
  octaveCounter = octaveCounter - 1;
  allMidiNotesOff();//turn off all previous notes
  for (int i = 0; i < 25; i++) {
    midiNotes[i] = defaultMidiNotes[i] + octaveCounter * 12;
  }
  Serial.print("Octave up/down for ");
  Serial.print(octaveCounter);
  displayReading(3, 0, 0, 0, 0);
}
```

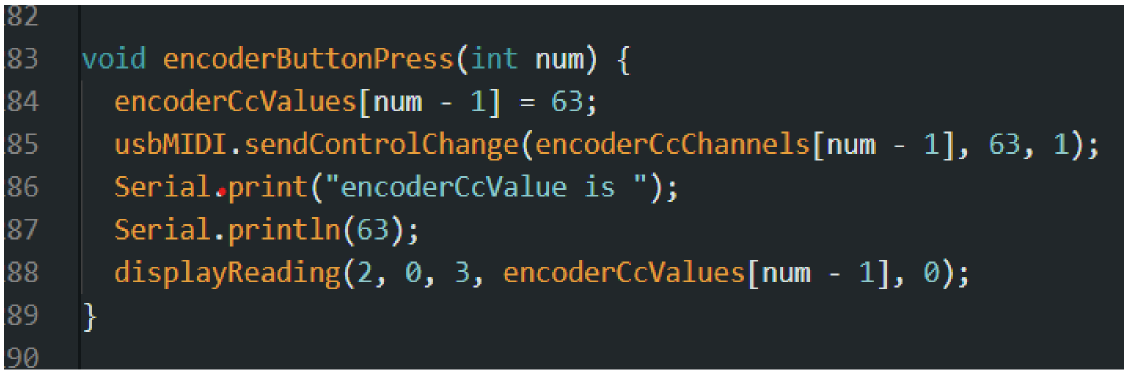
Once the switch is pressed down, the function will be triggered, we will have a new notes array with shifted midi notes.
With the octave counter, we can shift our current register to anywhere we want from between midi note 0-127.
The note pressing function is very simple, using usbMIDI.sendNoteOn().

The encoders and faders:

There are also 4 encoders you can freely map to any knobs you want in the DAW. I do not have a good idea for how to use the master encoder.
If we want to use the master encoder for switching through of channels on the DAW mixer, we need write custom script for this, which require extra work on DAW's specific guides/documentation.
Currently, during testing, I am assigning the 4 pot and 4 faders to 4 channels in FL studio.
FL studio will auto detect the message, and you can select the DAW component you want to map. FL studio will link the hardware and the component.



```
void encoderRightClick(int num) {
  encoderCcValues[num - 1] = encoderCcValues[num - 1] + 5;
  encoderCcValues[num - 1] = constrain(encoderCcValues[num - 1], 0, 127);
  Serial.print("encoderCcValue is ");
  Serial.println(encoderCcValues[num + 1]);
  usbMIDI.sendControlChange(encoderCcChannels[num - 1], encoderCcValues[num - 1], 1);
  displayReading(2, 0, 1, encoderCcValues[num - 1], 0);
}
```

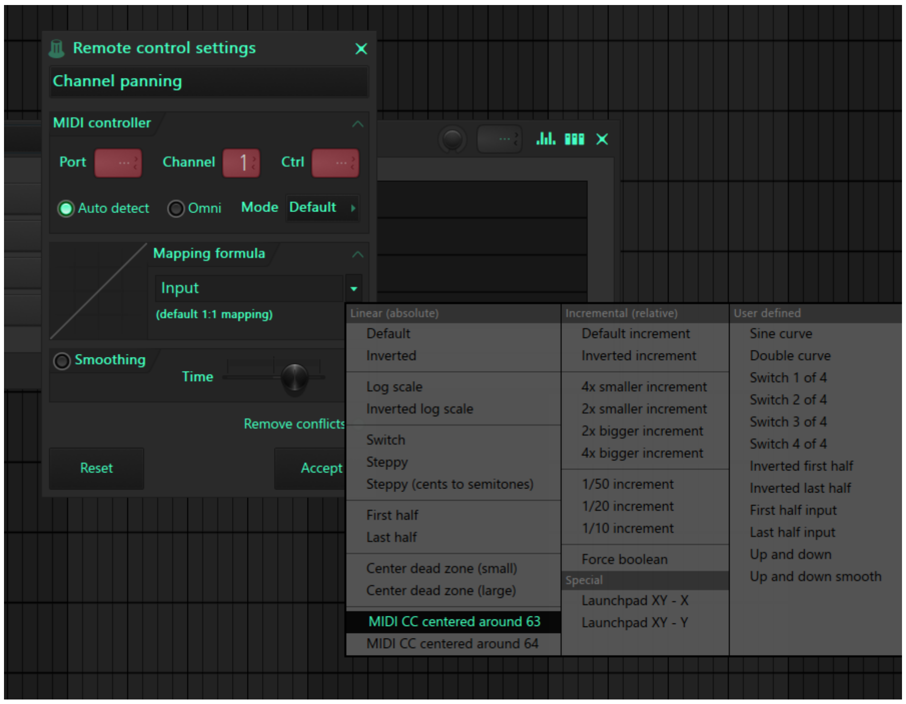


```

void encoderButtonPress(int num) {
  encoderCcValues[num - 1] = 63;
  usbMIDI.sendControlChange(encoderCcChannels[num - 1], 63, 1);
  Serial.print("encoderCcValue is ");
  Serial.println(63);
  displayReading(2, 0, 3, encoderCcValues[num - 1], 0);
}

```

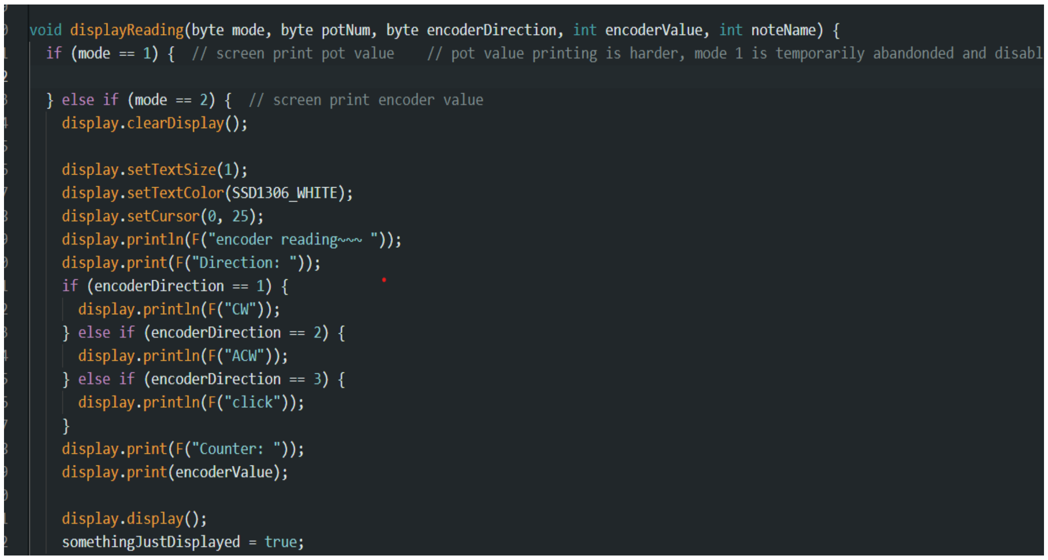
The cc value and channel number are both ranging from 0 to 127.
Keep in mind that some channel are usually mapped to a specific parameter, like channel 64 for piano pedal. I just used 1,2,3,4 channel for now.
For reset to center, there is something we need to know.
The cc value is 0-127, there is 128 levels, but midi values are only integers.
Ideally we need a value 63.5, if we want pan back to center?.
But .5 does not exist in the midi world, how?



So for FL studio, in the mapping menu, there is different mapping formulas.
To achieve our goal, we will just select MIDI CC centered around 63.
Then, we will just send 63 when we want reset to center, then the DAW will map 63 to the “center” value.
Different DAW may have different procedures. Check specific guides.

Display

I know it seems a little stupid and inconvenient when looking at my code of the display.
But, there it is:



```
void displayReading(byte mode, byte potNum, byte encoderDirection, int encoderValue, int noteName) {
  if (mode == 1) {  // screen print pot value    // pot value printing is harder, mode 1 is temporarily abandonded and disabl

  } else if (mode == 2) {  // screen print encoder value
    display.clearDisplay();

    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 25);
    display.println(F("encoder reading~~~ "));
    display.print(F("Direction: "));
    if (encoderDirection == 1) {
      display.println(F("CW"));
    } else if (encoderDirection == 2) {
      display.println(F("ACW"));
    } else if (encoderDirection == 3) {
      display.println(F("click"));
    }
    display.print(F("Counter: "));
    display.print(encoderValue);

    display.display();
    somethingJustDisplayed = true;
```

Each block’s function is pretty similar, because I am just printing different information on the screen for different action. Check full codes in the paste or on Github.
There are 5 blocks, trigger by groups of if statements:
If mode =1, then code for mode 1…
If mode =2, then code for mode 2…

The Drum Sequencer Mode:

I just migrate the week9 lab to the final project platform.
The migration process is very simple, but took some time. Because it is replacing the LED commands to neopixel commands, and command neopixel* 8 requires more efforts than controlling LEDs.
It is a separate .ino file. If we want switch two modes(each has large blocks of codes), we need more advanced theory and tools(vs code for example) to achieve this.

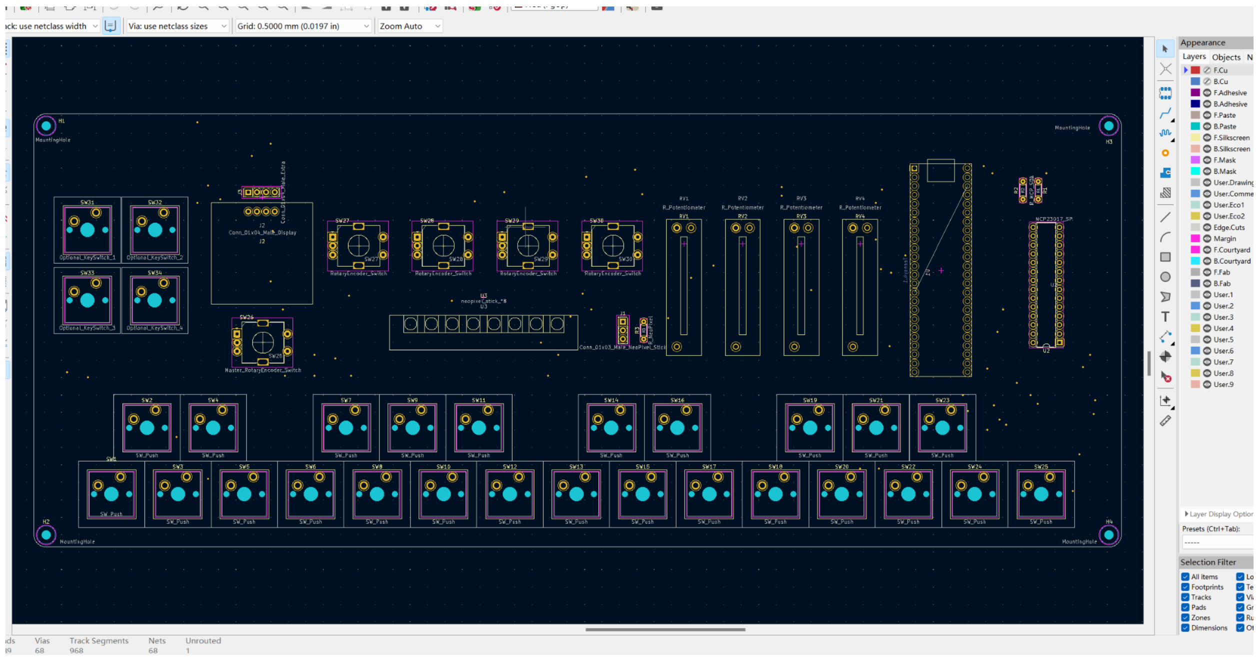
I just add more steps to it, now it's 8 steps. And you can switch patterns from 4 to 8. The current mode in my program is that:

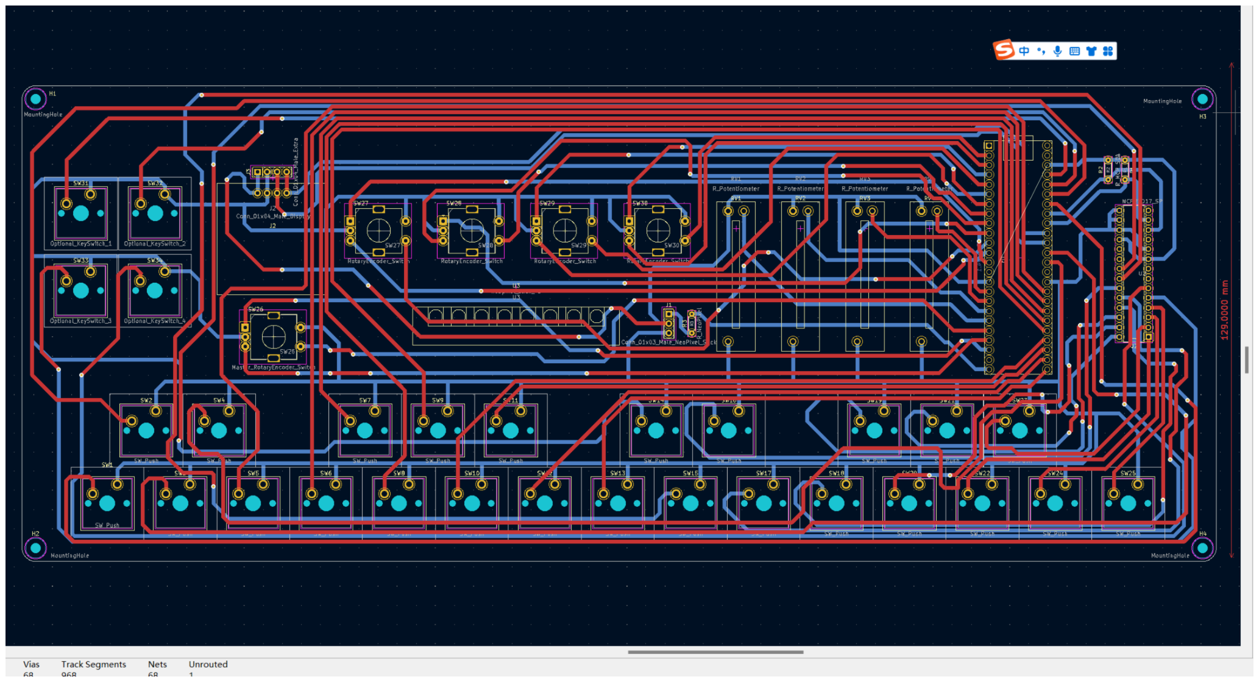
> On all 3 channels, maximum step is 4, the pattern is 4 steps.
> If one channels has steps more than 4, we will have an 8 steps pattern.
> Not very perfect to the ideal situation, but I am tired, maybe come back later.

There is the stepper class underneath controlling the time and steps, 8 in total.
So it is like running smaller patterns in each channel:
2 steps/per group, 4 groups for 8 steps
4 steps/per group, 2 groups for 8 steps

PCB:

I did everything in a day, started working on it during a lab at 11am, then continue working until 5 pm. I took a one hour lunch break in the middle.
Before you start actual working on your PCB design, I strongly recommend that you follow Steve's video doing everything he is doing before working on your actual stuff. Start doing something that does not have too much components on it. Because for mine project, there is a lot things going on, it is a mess after you start drawing the traces. Before I start on my design, I have did a simple op-amp circuit PCB with steve together during analog time. And I follow steve's video and did a basic teensy + a few button + 1 encoder small design.
The schematic part is very simple after you understand the video.
There is almost nothing to worry about schematics. The MCP has built in schematics and footprint.
I draw the symbol and footprint for Neopixel stick and faders, it is not hard. Reading the datasheet for the specific component and follow Steve's instructions about custom parts in Kicad, you will get there.
The hardest part, is when you go to the PCB design. Placement of all these thing is hard. Even though I follow the layout of MiniLab, but it is still hard for my own PCB design. The placement of the Teensy is crucial, I changed some of the pins on my schematics when drawing the traces. Everything will eventually have a link to the teensy, so the teensy is the center for all the traces. So the area to place teensy must be somewhere on the board that every hardware can access to it.

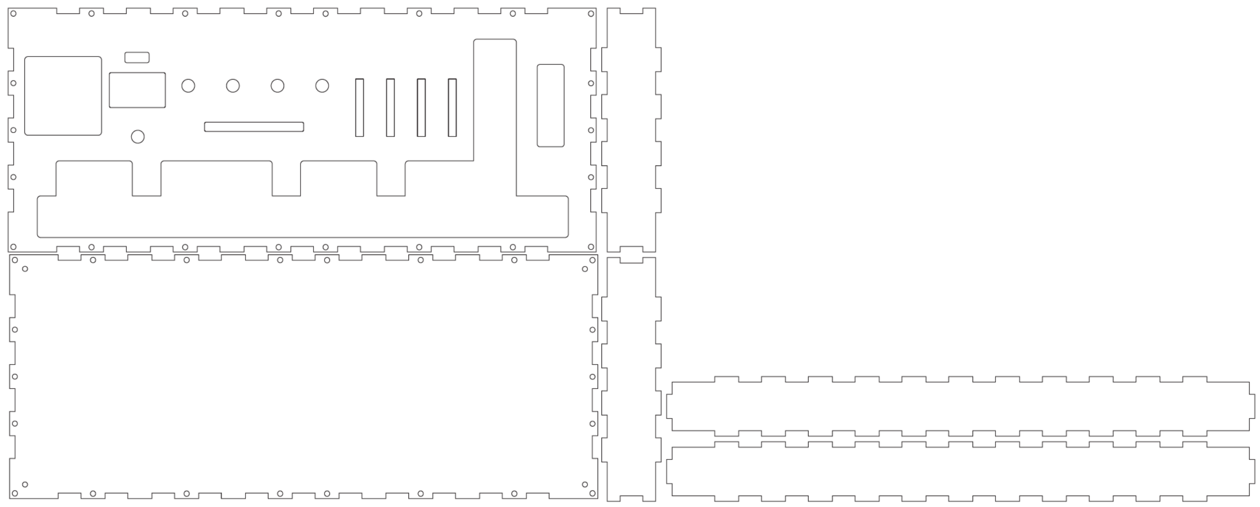




So in the finished design, the traces for the keyswitches is very hard to place. I did not draw a draft on paper before that, I just working on the PCB window. I think it will be a better experience if you put different components in groups, and plan the traces of each groups, before you actually draw anything on the PCB design.
For example, for 25 keys, there is traces going from the upper part of the board, and traces going from the lower part.

I did not realize until I got my board on hand that I just place 4 mouting holes on the corner but nowhere else. I later has to glu some hardware and place air bags underneath it for support.
Otherwise, there may be a chance will pushing my encoder buttons, my board will just break into half. Maybe putting some mounting in the empty area around the board, instead of just 4 corners.
There is also sacrifice later for the box design, because for the first time doing this, we ideally should use sockets. Socket + ICs will have a different height than other components.

Laser Cutting Box:



This one is a little tricky. It only comes to assembly that I realize my design was wrong, but there is no chance this semester for me to do a new design for it. So sadly that is it.
I only used the front and back, 2 faces out of 6 faces for this box.
two reasons for not using the other 4:

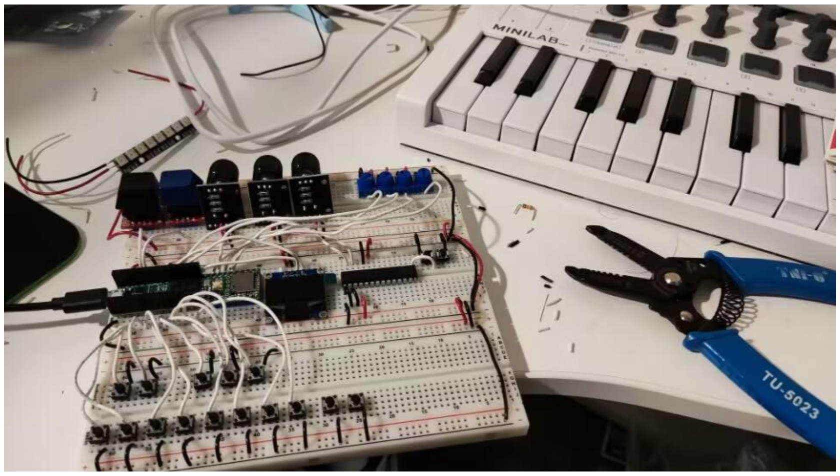
1. There is something wrong with the kerf size, I print with standard kerf size for acrylic, like 0.20mm, but the print piece would not fit it.
2. I add extra mounting holes, thought about the distance on the 2 dimensional surface, but did not think about the spatial distance on 3 dimensional space, and did not really considered the size of the mouting screws. Therefore, after I installed the all the screws, I found that even I can polish the 4 faces, there is no way to installed the 4 faces when having the screws on.

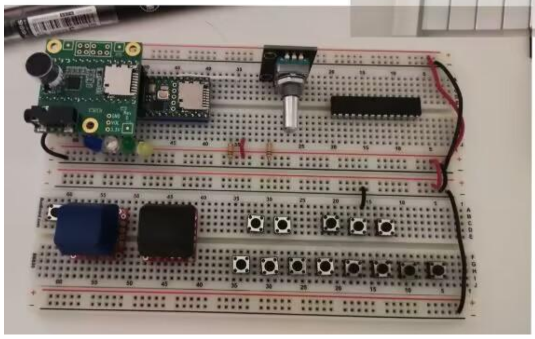
Well, maybe print a small box to give it a try before making the big box for the project. I do not have too much time as we are approaching the finals, so it is a bit of a rush.

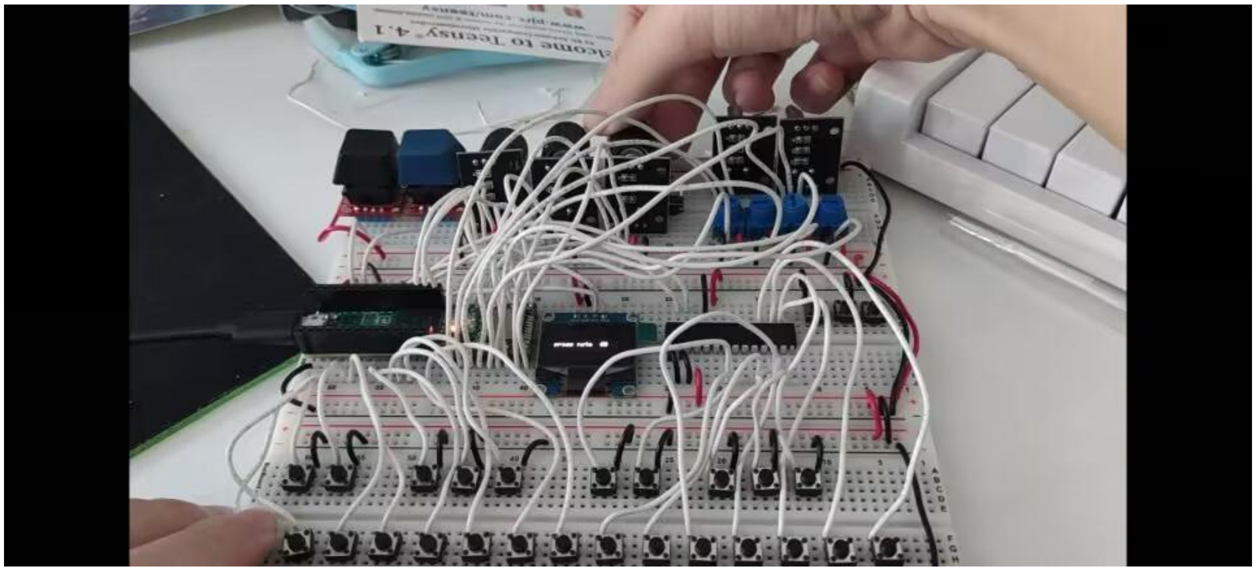
Code: Too long to paste in here, please see on Github

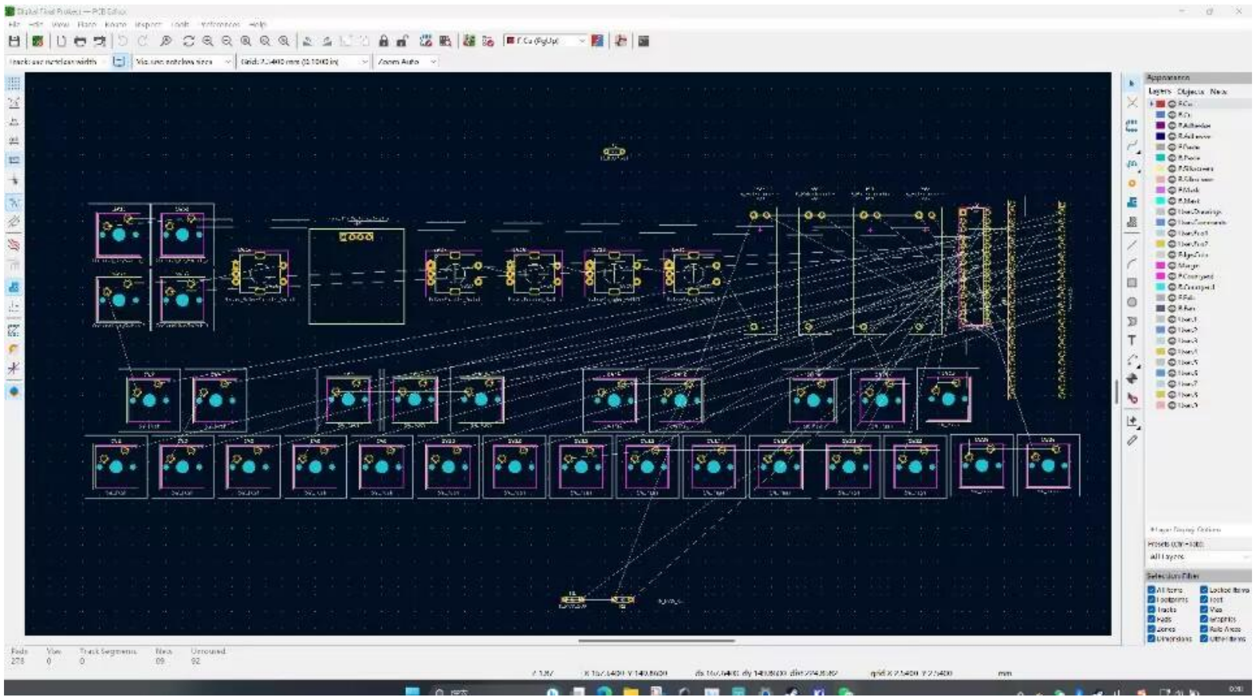
I may update it in the future, but I will save a current version for final project, with note “Spring 2023 final”

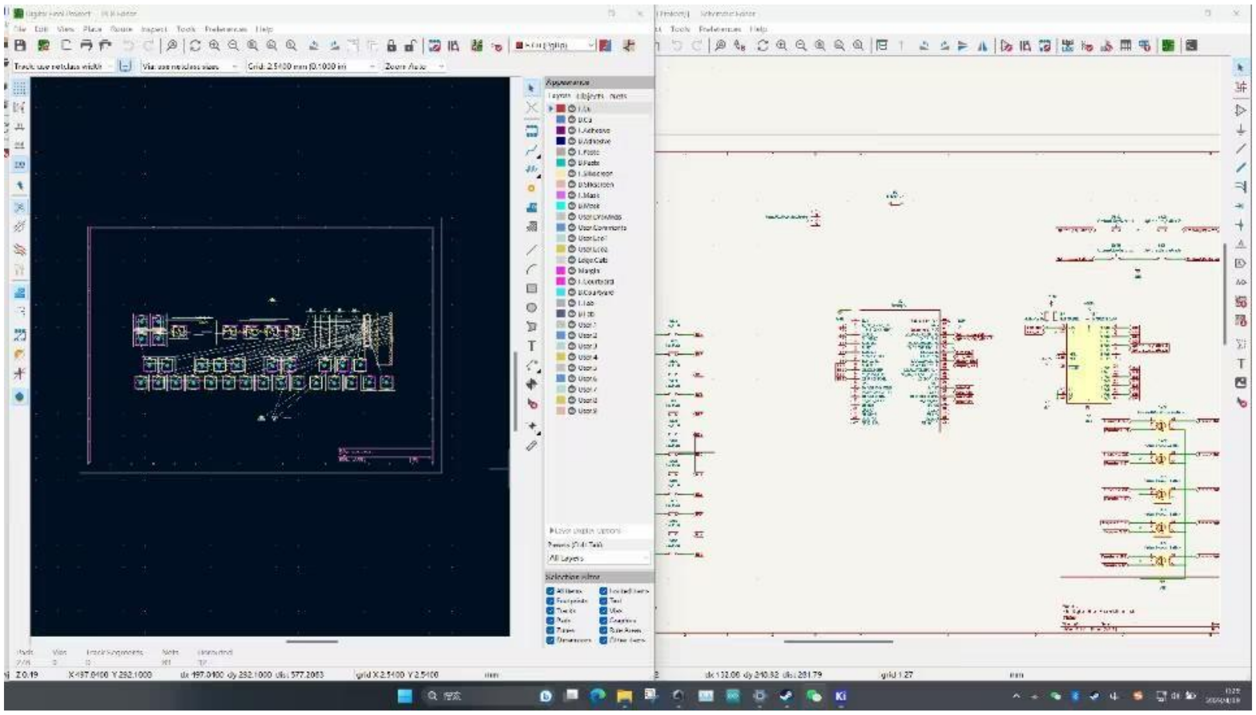
Other photos:
Starting from the beginning of April and till the end.

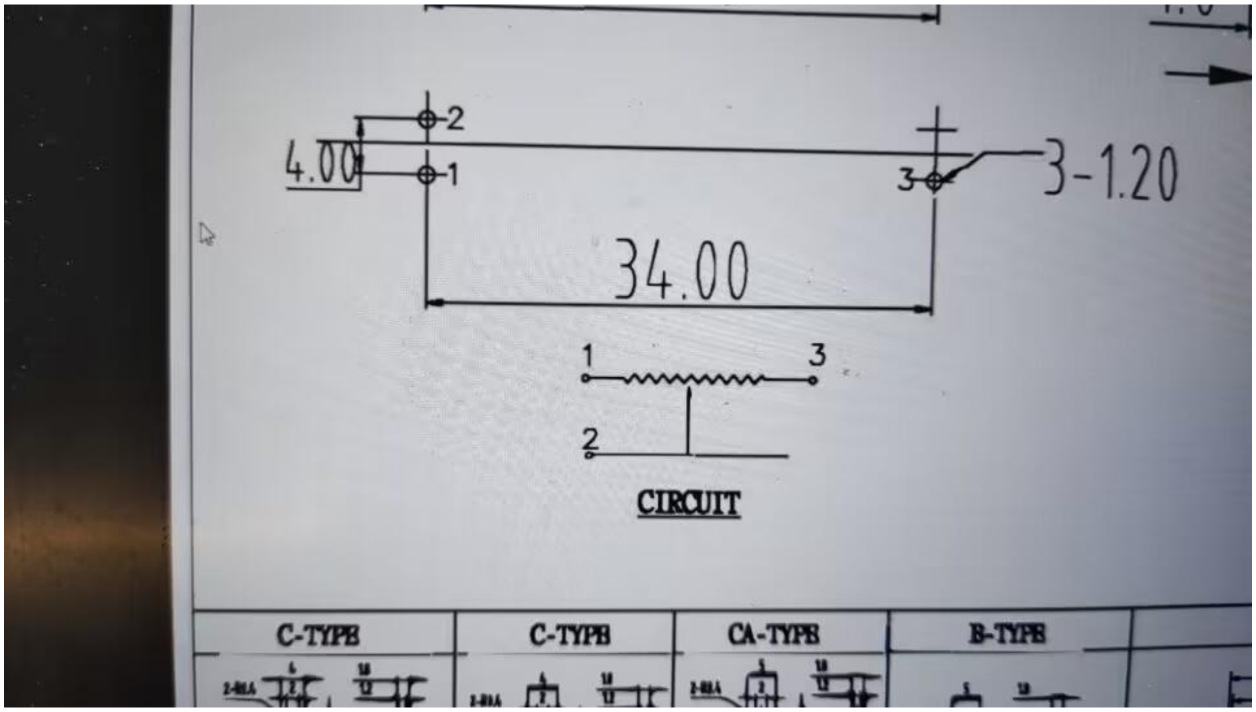


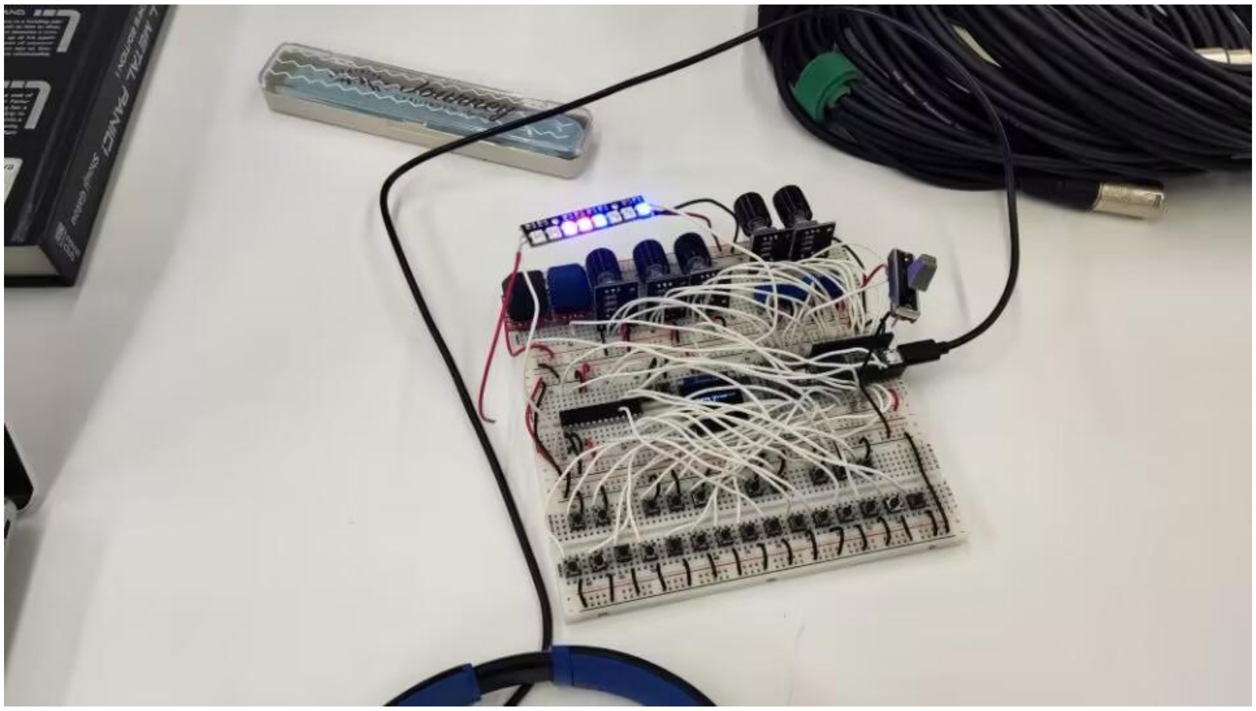


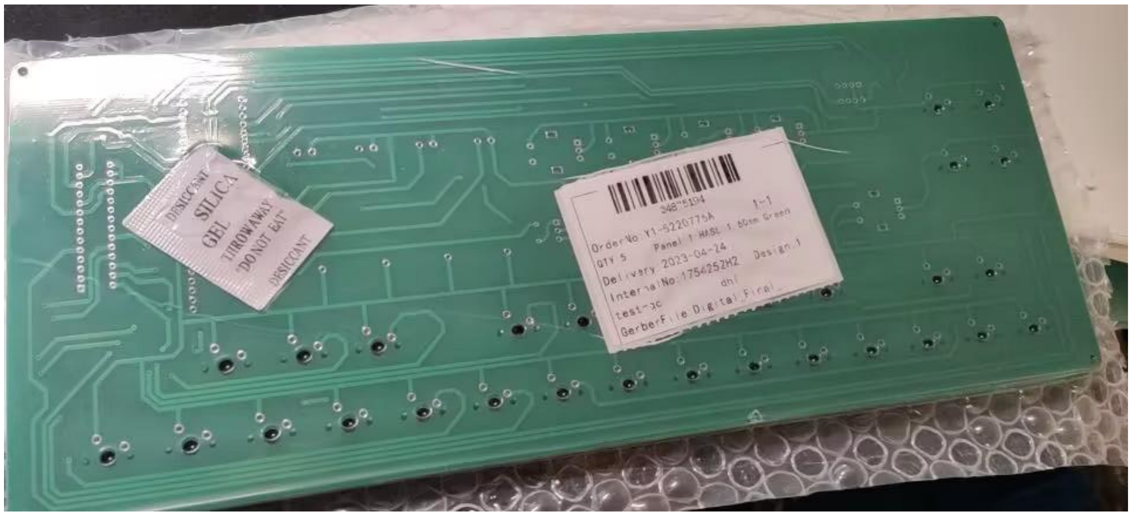

Welcome to Teensy® 4.1
www.pjrc.com/teensy

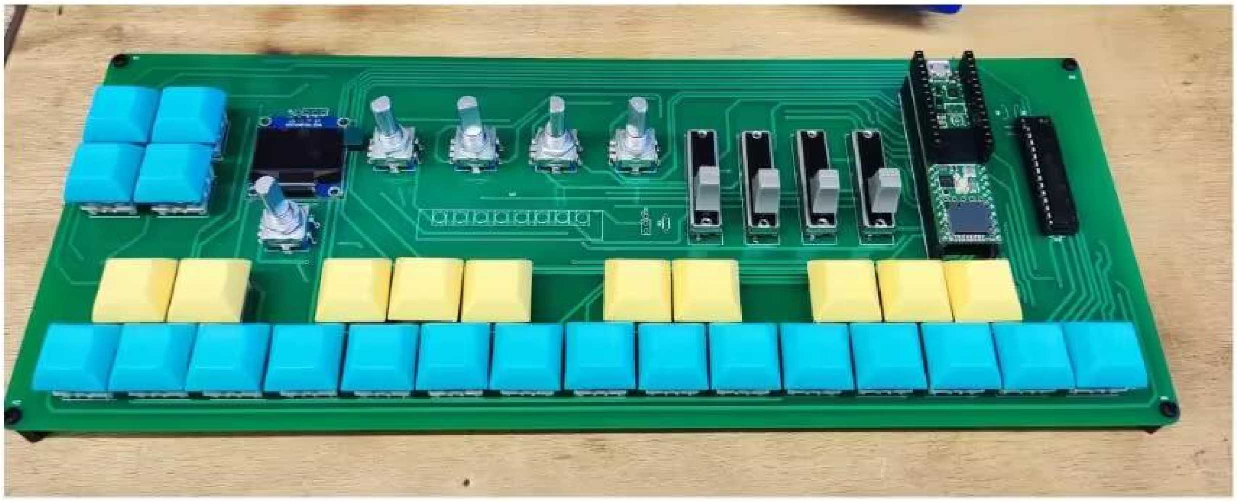

Digital Final Project — PCB Editor
File Edit View Place Route Inspect Tools Preferences Help
Appearance
Layers Objects Nets
Selection Filter



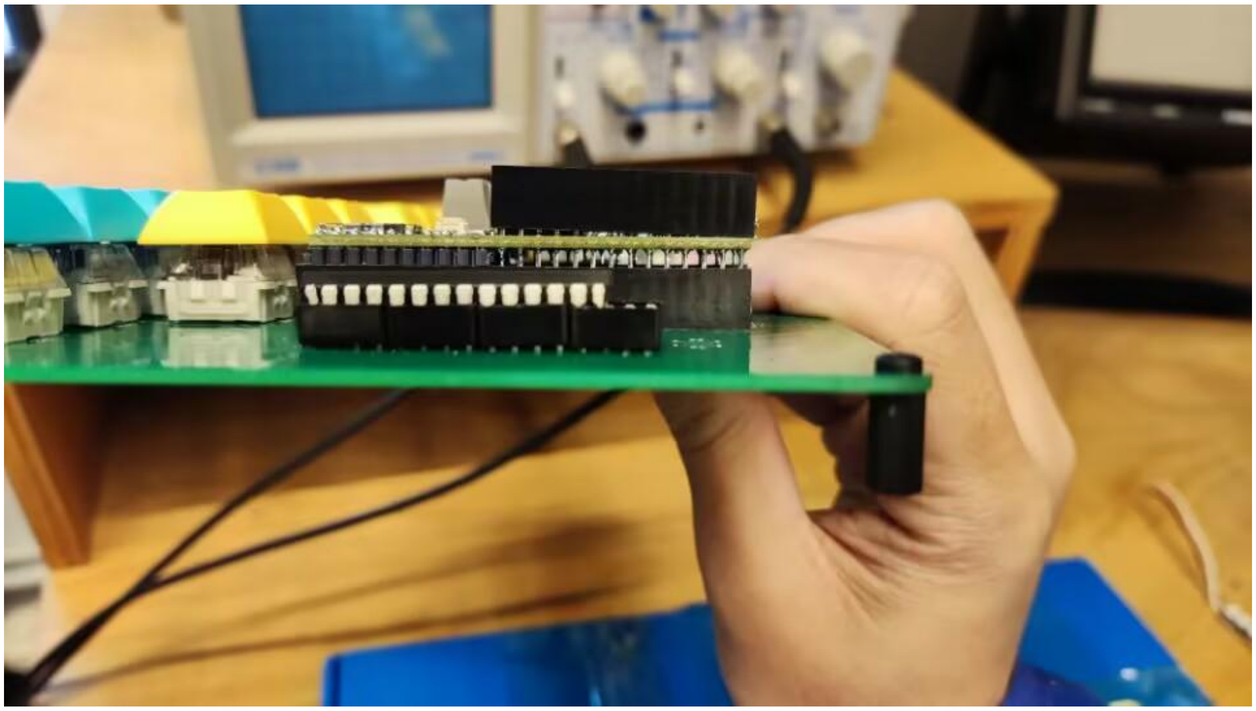

2
4.00
1
3
3-1.20
34.00
1
3
2
CIRCUIT
C-TYPE
C-TYPE
CA-TYPE
B-TYPE




SILICA GEL
THROWAWAY
"DO NOT EAT"
DESICCANT
348"5194 1-1
OrderNo:Y1-5220775A
QTY:5 Panel:1 HASL 1.60mm Green
Delivery:2023-04-24
InternalNo:1754252H2 Design:1
test-xc dhf
GerberFile:Digital_Final_



MINILAB3
ARTURIA
SUB
M-S



For Faster PCB Proto
For Faster PCB Prototype

Faster PCB Prototype
For Faster PCB Prototype