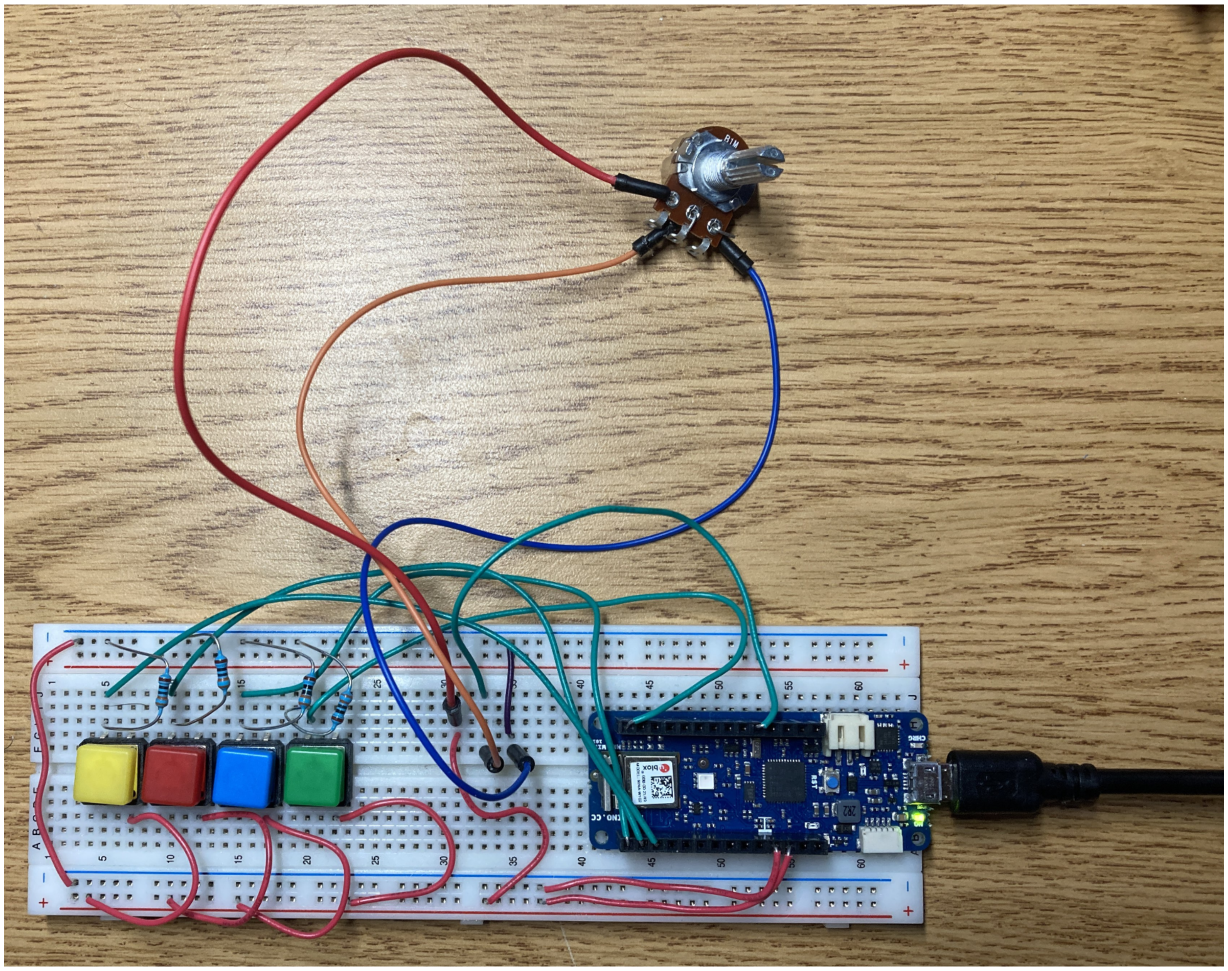
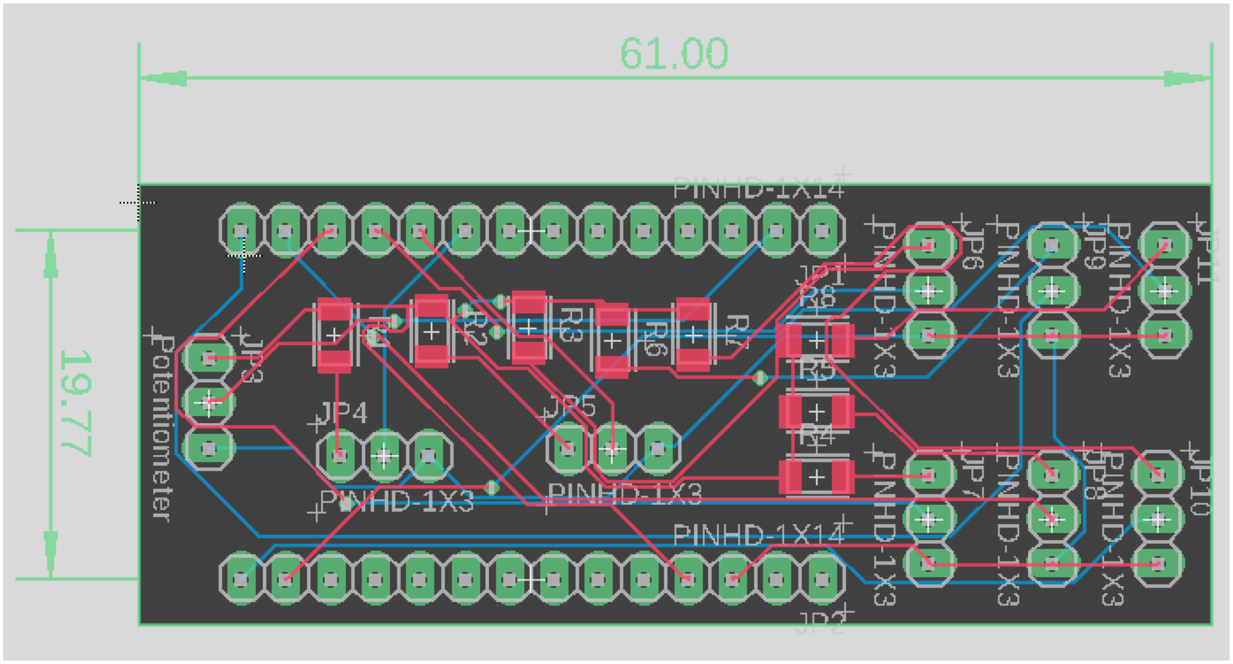
Bluetooth Potentiometer

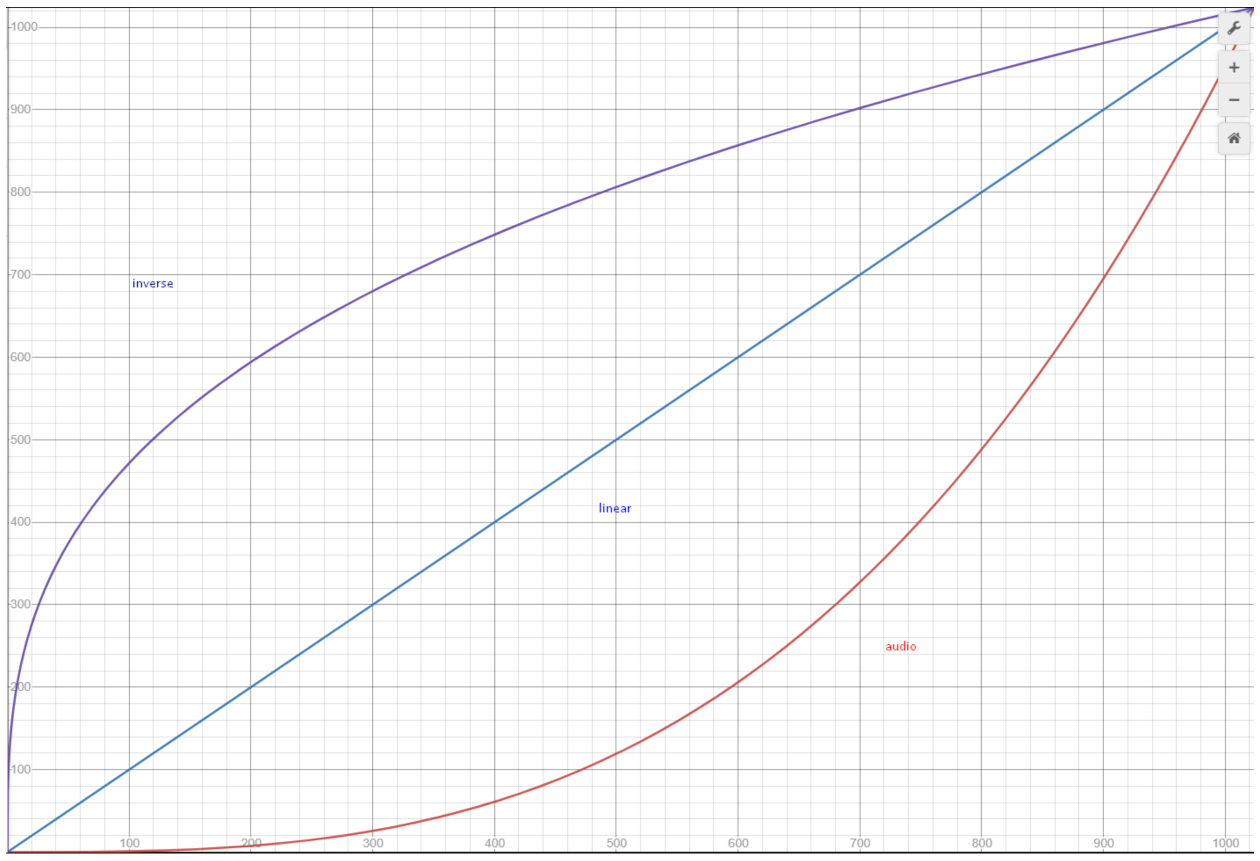
In this project we continued off the design of the Les Paulverizer Project of transmitting a button push over the BluetoothBLE of an Arduino MKR WIFI 1010 to be received by a MIDI device. We took the original code from the prior group and added in the ability to be able to transmit a variable value using a potentiometer by converting its analog output to a digital value throughout the Arduino's ADC converters. Some other changes should allow the user to easily be able to attach 21 buttons or 14 buttons and 7 potentiometers to Arduino, though no testing has been done to see if the Arduino can handle that many inputs all at once.

The user needs to download the libraries MIDIUSB and ArduinoBLE into an IDE capable of uploading code onto the board, such as the Arduino software or an alternative like VSCode with the PlatformIO extension. At the top of the PERIPHERAL.ino file there is an abundance of defines, from which the user can comment/uncomment the different button/potentiometers based on their placement on the Arduino. For buttons there are two things they need to do, first uncomment the buttons they plan to use and comment the ones they don't. It is not necessary to comment ports they do not plan to use except for the ones that share pins with the potentiometers which are listed separate from the main 14 buttons. The second thing the user needs to do is select the note value that will be transmitted to MIDI through channel 1. After the configured code is uploaded, the arduino is setup. The board can be connected to its peripherals with a breadboard or the Printed Circuit Board that is pictured below.

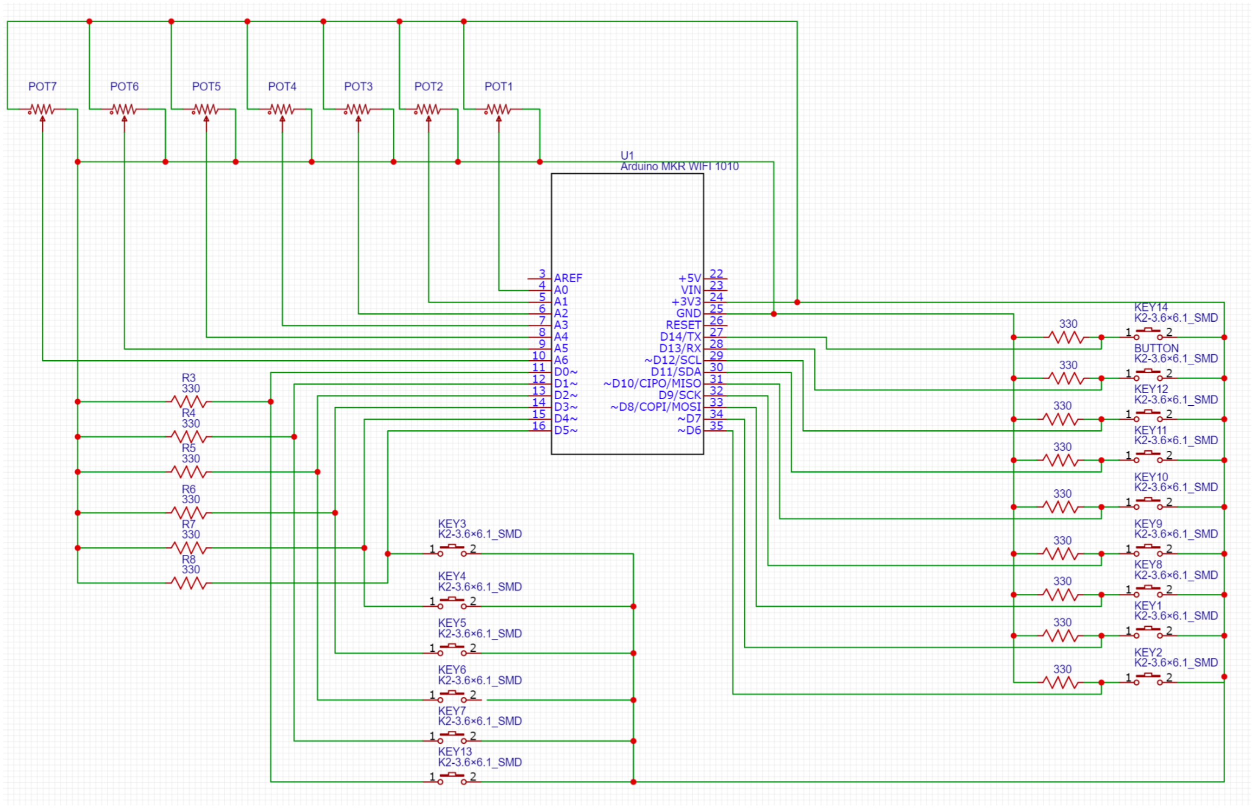



61.00
19.77
PINHD-1X14
JP1
PINHD-1X14
JP2
Potentiometer
JP3
JP4
PINHD-1X3
JP5
PINHD-1X3
R2
R3
R6
R7
R8
R5
R4
PINHD-1X3
JP6
PINHD-1X3
JP9
PINHD-1X3
JP11
PINHD-1X3
JP7
PINHD-1X3
JP8
PINHD-1X3
JP10

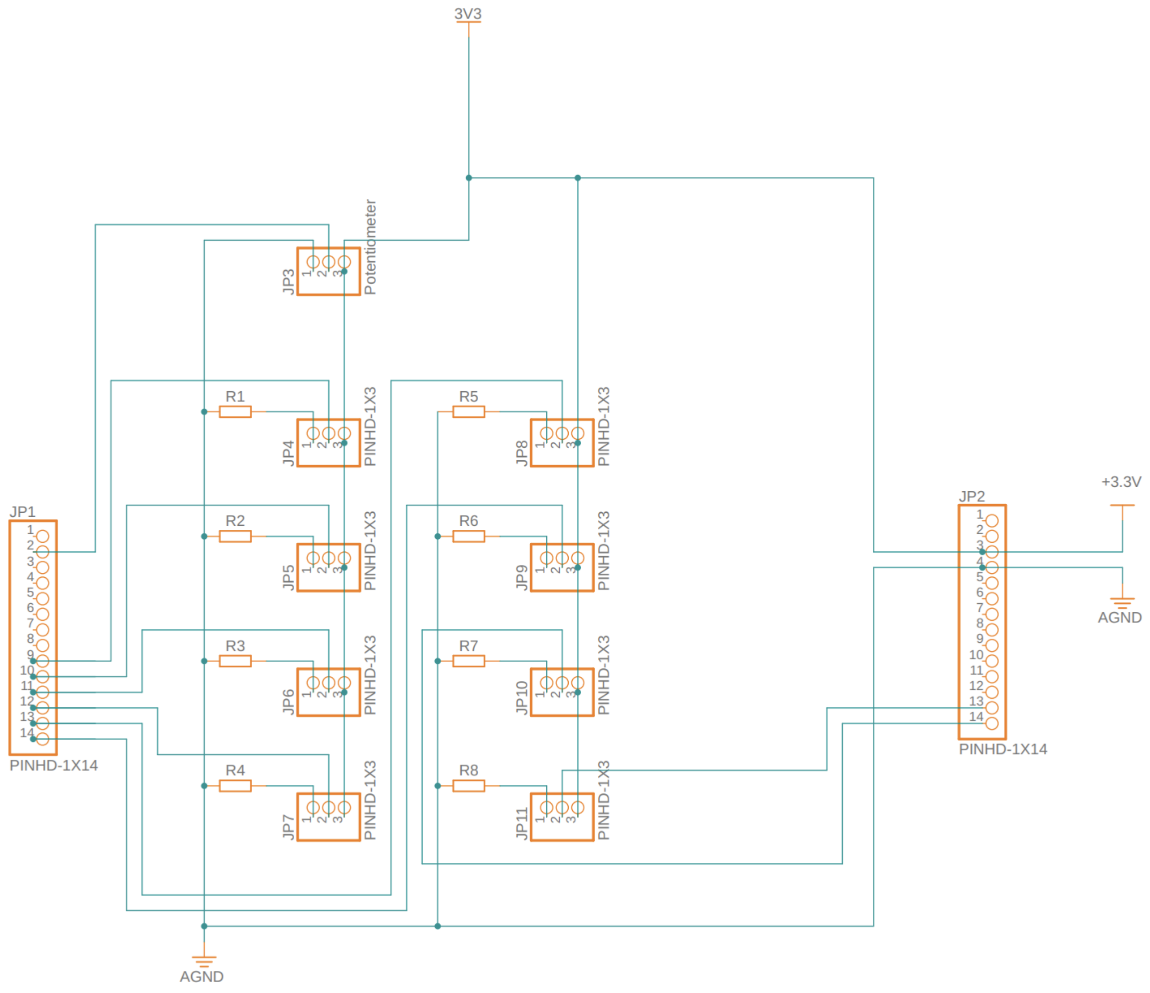
Below where the buttons are defined on the same file is where the user can set up potentiometers. There are 7 potentiometers that the user can select in a similar fashion to the buttons but must make sure they leave the buttons above commented out or it will try to double the pinout for certain pins. The note that the MIDI value can be changed to a desired MIDI value on channel 7 below that. Lastly for potentiometers, the user can define what type of taper they want to use for the potentiometer with three options: LINEAR, AUDIO, and INVERSE. They can use those words when defining the curve rather than a value since these variables have a defined value to them. The curve of these can be seen on the graph below.



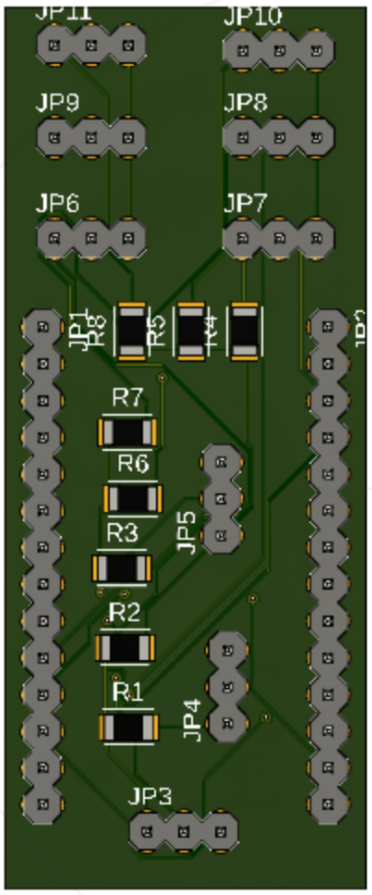
To wire the device, the Arduino can be wired following the schematic listed below on a breadboard using buttons and potentiometers. This schematic shows an instance where all the ports of the Arduino are populated.

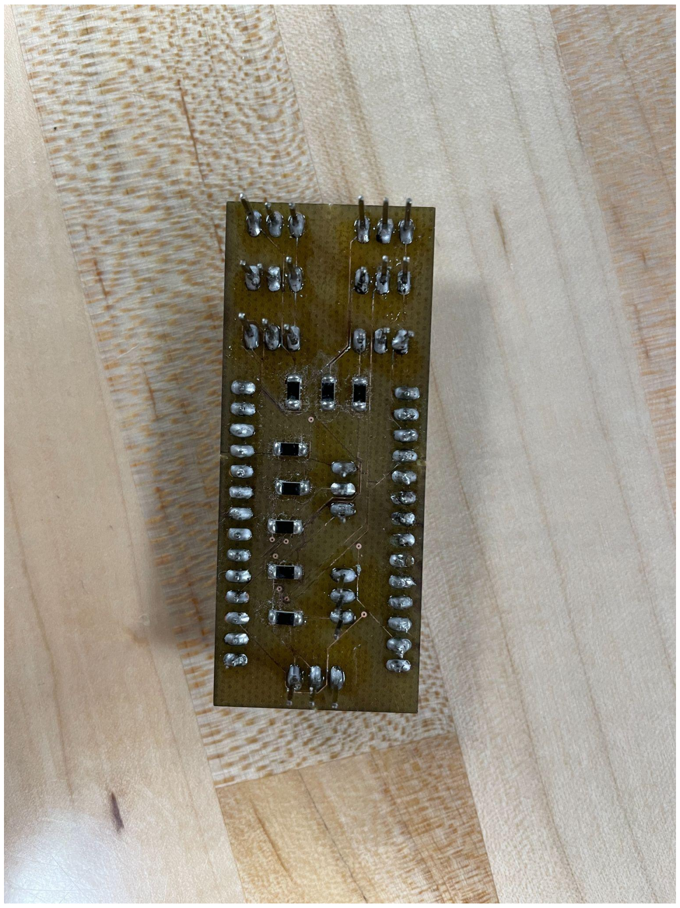


The alternative and more space efficient way would be to use the PCB that was developed to simplify the resistive network. The schematic of the PCB is listed below. The GERBER files for the PCB are also located in the GitLab.



The resistors that were soldered onto the PCB are 330k SMD 1206 package resistors. The ports used for the potentiometer do no not have any silkscreening due to the limitations of the makerspace. However in the diagram below, JP3 should be used for the potentiometer and the other 3 pin headers can be used for buttons. The pin descriptions (GND, VCC, or DATA) can be referenced in the schematic. A LiPo 3.6 volt battery can be used to power the device so that it can be used wirelessly in a guitar and also be recharged through the USB port.





Possible areas of improvement include the approximation of the audio taper is very accurate but the peak of the inverse taper is in the wrong place. The inverse taper should be the audio taper mirrored across the y=512 axis, while our current inverse taper is mirrored across the y=x axis. There was research done into different resistance ranges of potentiometers for a noiseless result, but there is still a significant amount of noise. The noise is slightly reduced with the 10-bit ADC reduction to the 8 bit 127 max of MIDI. This can be entirely negated with a capacitor in series with the potentiometer but there was not enough testing done on this. There was a code solution but it would decrease the polling rate of the potentiometer too much. Future teams should find a permanent and more elegant solution. In addition, these tapers should be approximated as the actual precise math uses too many cycles to keep the polling rate fast when using a large amount of potentiometers. A full pickguard integration is definitely desirable in the future.