Keyboard Labels
October 10, 2024
Jim Hoagland, Myrrh Khan, Anthony Previte

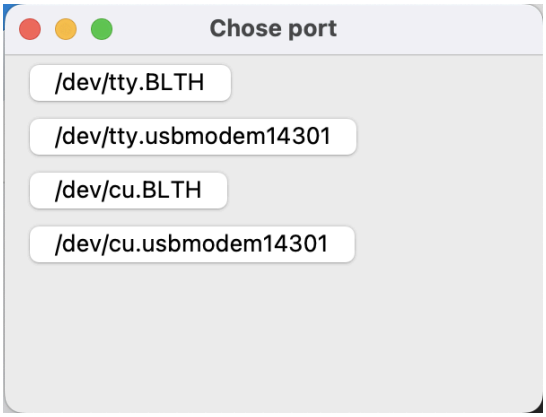
README

The goal of our project was to develop a proof-of-concept prototype for digitally controlled key labels on a MIDI keyboard. Our current prototype uses a graphical user interface (GUI) developed in Python with Tkinter, two LCDs mounted on the keyboard with a 3D printed frame, and an Arduino that connects the displays to the interface via pySerial. The two LCDs, each containing two lines of 16 characters, act as labels for four separate keys, with each line independently updateable. Users can configure labels through the GUI, which sends updates to the Arduino to be displayed on the LCDs. The codebase includes two main components: two Python files driving the GUI, and an Arduino file that interprets serial signals to control the LCDs. The interface allows for port selection, key selection, and text input.

Background Research and Ideas

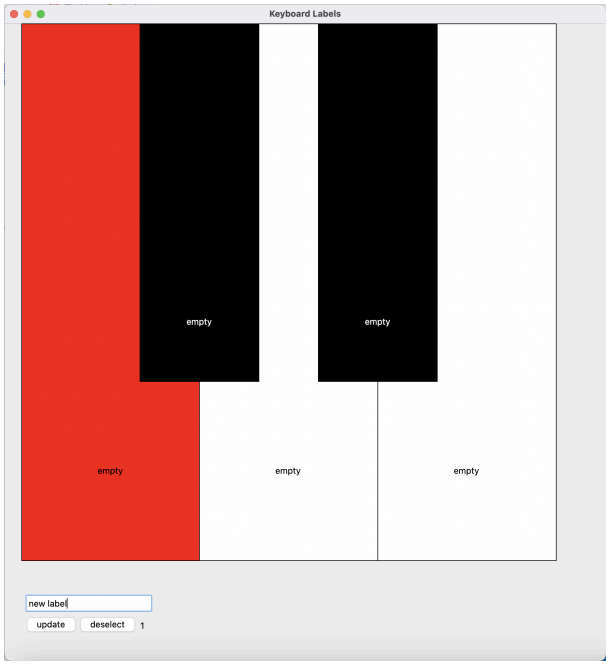
When deciding on a direction to take this project, we had to research where to put the displays, what type of display to use, and how to control the displays from a computer. We decided that mounting the displays on the bezel above the keys would be the best option because the keys are small and it is difficult to embed displays and electronics in them. It would also require disassembling a MIDI controller. A final design would have a display mounted into the MIDI controller itself, but for our prototype, we mounted external displays on top of the controller. We considered LCD and eInk displays and opted for LCD because they do not require an external light source, and because they have easy integration with microcontrollers. We chose displays that were narrow enough to fit within the width of two keys while still having enough space to display a reasonable amount of text. We used an Adafruit Feather Bluefruit M0 microcontroller to drive the LCDs, which we obtained from Professor Manzo's lab.

GUI

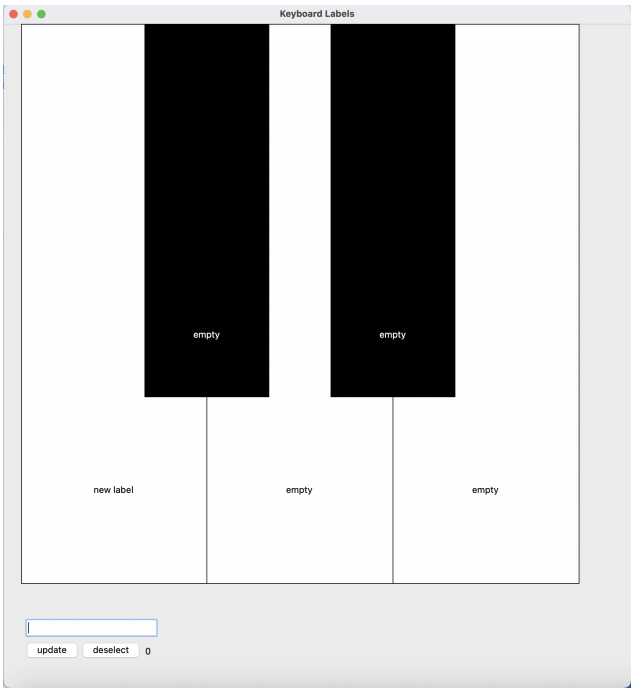


When you first run KeyboardLabelGUI.py, the startup screen pictured above will pop up that allows you to select the port that your Arduino is connected to. NOTE: it usually takes up to five seconds for the program to process which port you select and move to the next screen.

Once a port is selected, the screen pictured above will come up that has options to select a key to update, enter the new label text, and shows the current labels being displayed.



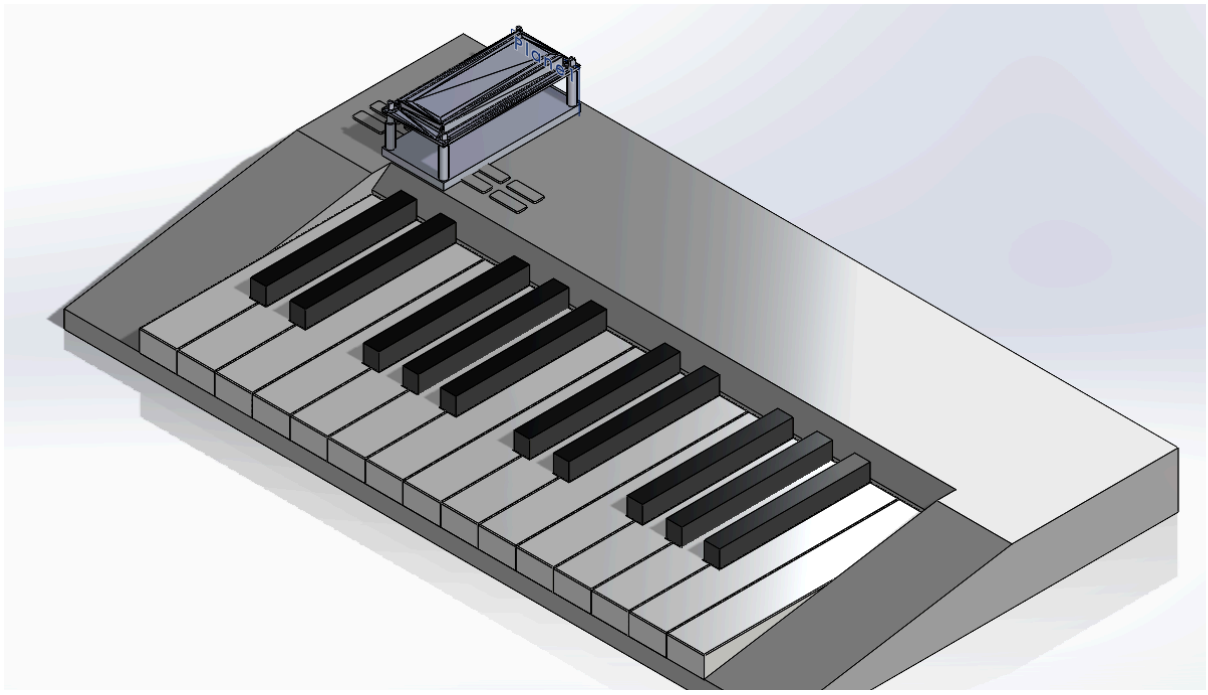
Clicking on a key will select it and turn it red. From there you can enter the label text in the entry box and press update to send the string to the LCD.



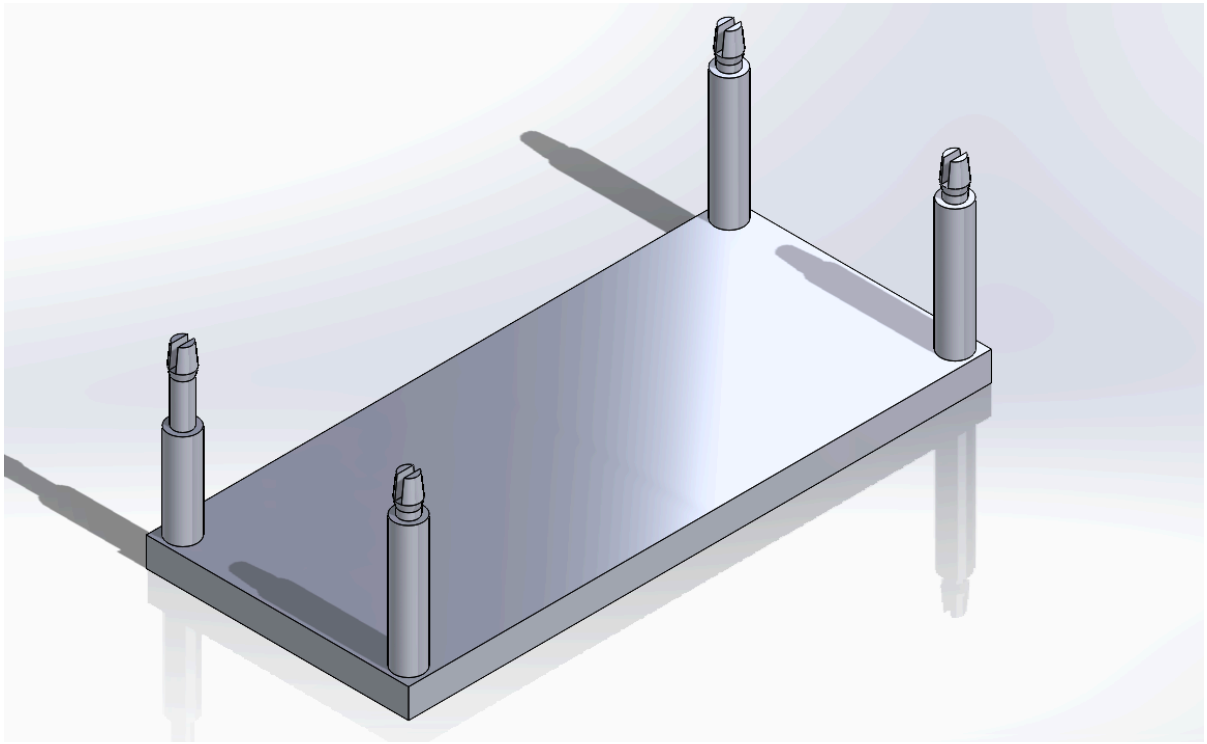
After pressing update, press deselect to make all the keys the normal colors again. The label on the keys in the GUI will update with the new text as well so you can see the existing labels and key availability.

3D Modeling

A CAD model of a MIDI controller and the display were created to create a mount for the display. The backpack creates an uneven surface on the back of the display that would be infeasible to mount something to. The 3D printed mount securely clips into the display/backpack assembly using the four mounting holes on the corners of the display. The mount provides a flat surface for the back of the display to allow easy attachment to the MIDI controller.

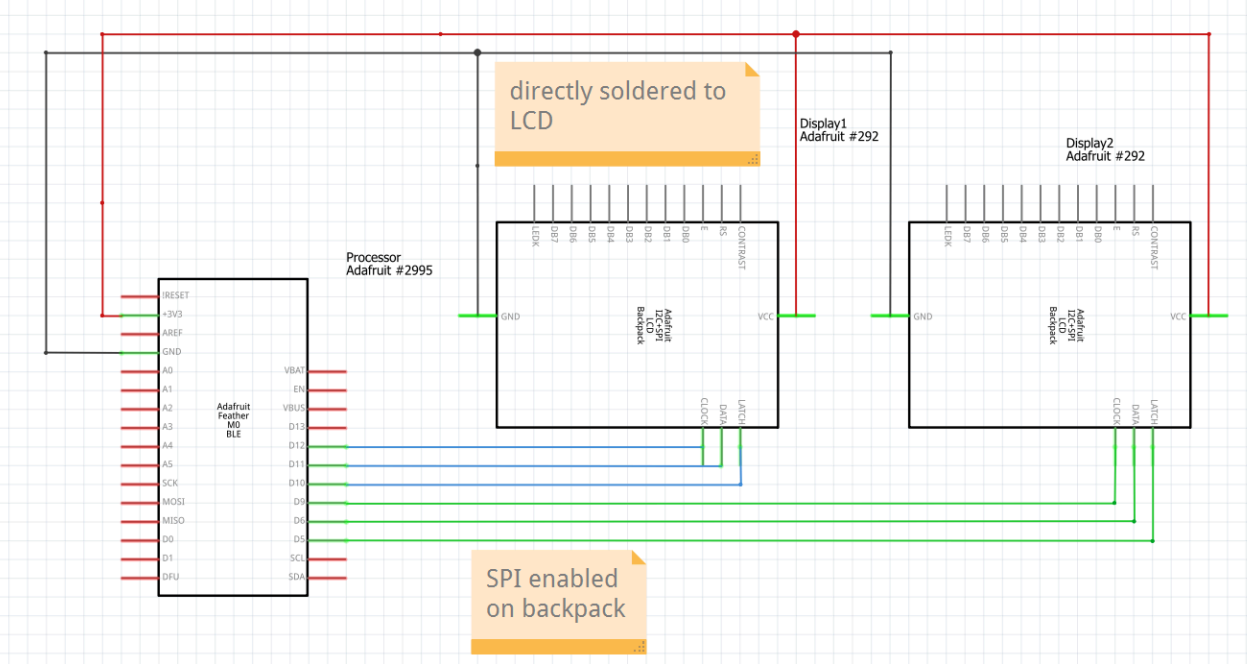


Display mounted to MIDI controller.



Mount for display

Circuit Diagram



Future Ideas

This was just the beginning of this project and there are many directions we can see future work taking. One idea is embedding the labels on the keyboard or the keys themselves rather than mounting just above. Another idea would be touching up the GUI visually, and in terms of accessibility. This means improving from a python script desktop app to potentially a downloadable app, plug in for a DAW, or website. Additionally, the code can be refactored to allow support for more than two displays and automatically adjust the number of keys accordingly.

Components

- *Adafruit Feather M0 Bluefruit LE*
- Adafruit 16x2 LCD display (*Standard LCD 16x2 + extras - white on blue*)
- LCD Backpack (*i2c / SPI character LCD backpack - STEMMA QT / Qwiic*) – reduces number of pins needed for LCD from 16 to 3
- Breadboard
- Wires
- Mount for displays (see CAD models)