Report (Daniel Ko, 862153939)

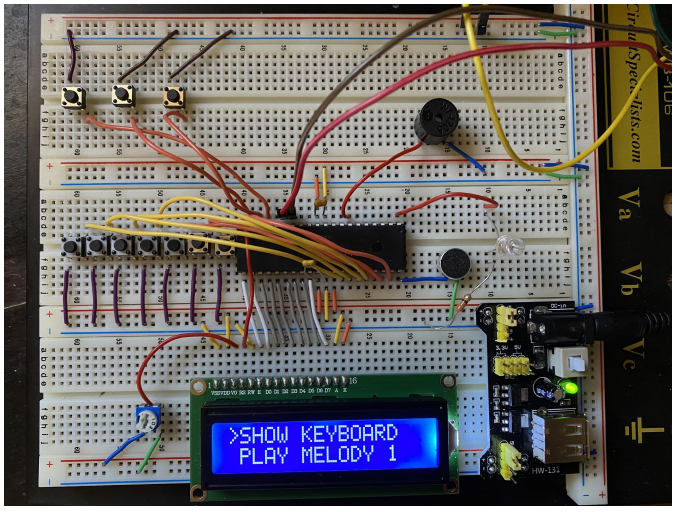
Full Demo link: https://youtube.com/playlist?list=PLbQc7FSzU3zvgQQ1oXXl9lvFC6RS1id9I

- Complexity 1: https://youtu.be/QwFSXZ6Jds0
- Complexity 2: https://youtu.be/jh2mRpE3iTU
- Complexity 3: https://youtu.be/xWpH7MeLUec

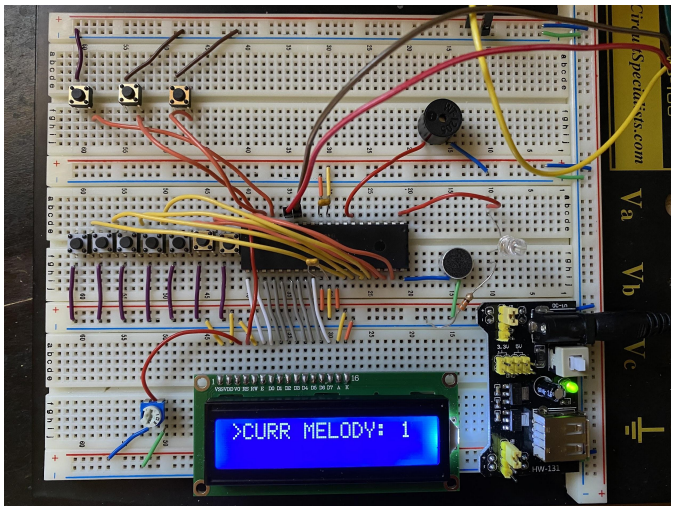
Project Description:

This custom lab project involves a playable miniature piano that has a real-time LCD model of the piano keys being played. The LCD screen includes the note names on the bottom row, and the keys being played on the top row, which are differentiated between whole (full-bar) and half (half-bar) keys with custom LCD characters. Pressing the piano keys will emit the corresponding piano note from a speaker. The miniature piano system also includes several melodies that are stored into the EEPROM of the microcontroller, which can be played using the speaker. The LCD screen can also display a menu with the options of showing a real-time keyboard, playing the current EEPROM melody, and switching between EEPROM melodies. To switch between EEPROM melodies, the user can tap a small microphone to cycle through the available melodies when that option is selected in the menu screen. A joystick is used to navigate through the menu options, and buttons are used for opening and closing the menu.

User Guide:



(first menu screen)



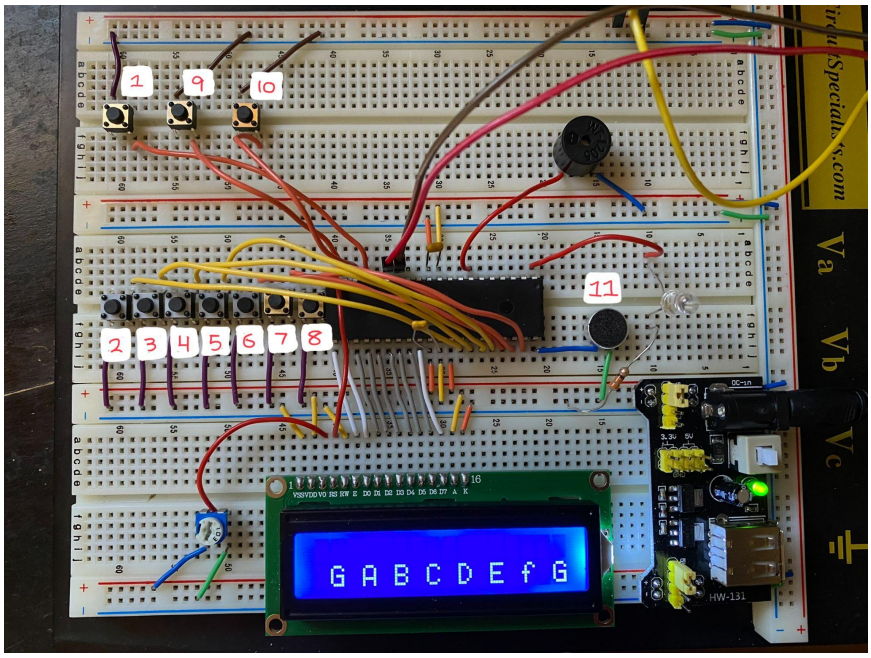
(second menu screen)

Rules:

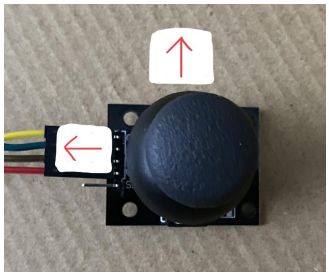
By default, the LCD screen will display the miniature piano. When the piano is displayed, buttons 9 and 11 (labeled in diagram below) will not be registered. If the user presses button 10, the menu will be displayed, and buttons 1 - 8 will not work, since the piano is no longer being displayed.

When inside the menu, the user can press button 9 to select a menu option, and button 10 closes the menu without any selection. By default, pressing button 10 in the menu will display the miniature piano to the user again. In order to cycle through the EEPROM melodies, the user has to first navigate to the second screen; pressing the microphone will not change the EEPROM melody in any other screens.

Controls:



- 1: Piano key 1 (G on the left)
- 2: Piano key 2 (A)
- 3: Piano key 3 (B)
- 4: Piano key 4 (C)
- 5: Piano key 5 (D)
- 6: Piano key 6 (E)
- 7: Piano key 7 (f)
- 8: Piano key 8 (G on the right)
- 9: Select menu option
- 10: Open menu
- 11: Select next EEPROM melody



- Push left: Move menu cursor down
- Push up: Move menu cursor up

Special Considerations:

The microphone I received from Sparkfun turned out to be nearly broken with a very weak signal. It's weak enough to detect a finger tap, but not strong enough to detect specific frequencies that my original project required (I originally wanted to use the microphone to record melodies into EEPROM).

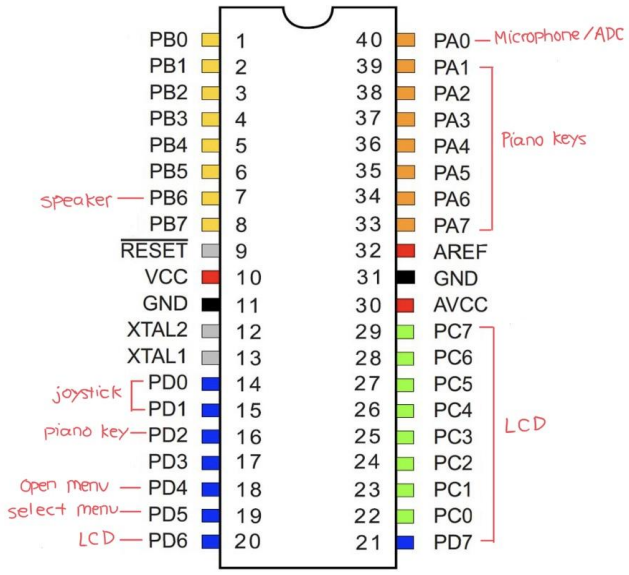
In order to get the microphone to work in such a short amount of time, I had to work around it by repurposing it as a button to cycle through my EEPROM melodies. Although the microphone now works by tapping your finger on it, it still doesn't work 100% of the time, so the user will sometimes have to tap the microphone a few times to get it to register.

For reference, here is my updated project proposal: Copy of CS120B Project Proposal .

Source File Links:

- main.c - main file to run the program. This file contains all the state transitions and main functions required to run the program. This file also contains the EEPROM readings for melodies, and all the logic required to navigate through the menu system and miniature piano. **Code obtained from** https://www.nongnu.org/avr-libc/user-manual/group__avr__eeprom.html
- io.c - source file for all lcd functions. This file also contains the function for creating custom characters to display on the lcd screen. **Code obtained from** https://openlabpro.com/guide/custom-character-lcd-pic/
- io.h - header file for io.c.
- simAVRHeader.h - responsible for all AVR functions.
- timer.h - includes all timer functions required for synchSM to run correctly.

Component Visualization



Technologies Learned

- Working with EEPROM that's built into the Atmega1284 microcontroller. I learned how to write floats into the EEPROM and read them based on their indexes.
- Using a joystick to navigate through menu systems. I learned how to connect a joystick to my breadboard and how to make the joystick's X and Y axis work independently and read into PINs.
- Using a microphone as a button. I learned how to read a microphone's inputs using ADC and checking if a finger tap is detected.
- Printing custom characters on the LCD screen. I learned how to create my own custom characters in the io.c file and print them correctly on the LCD screen.