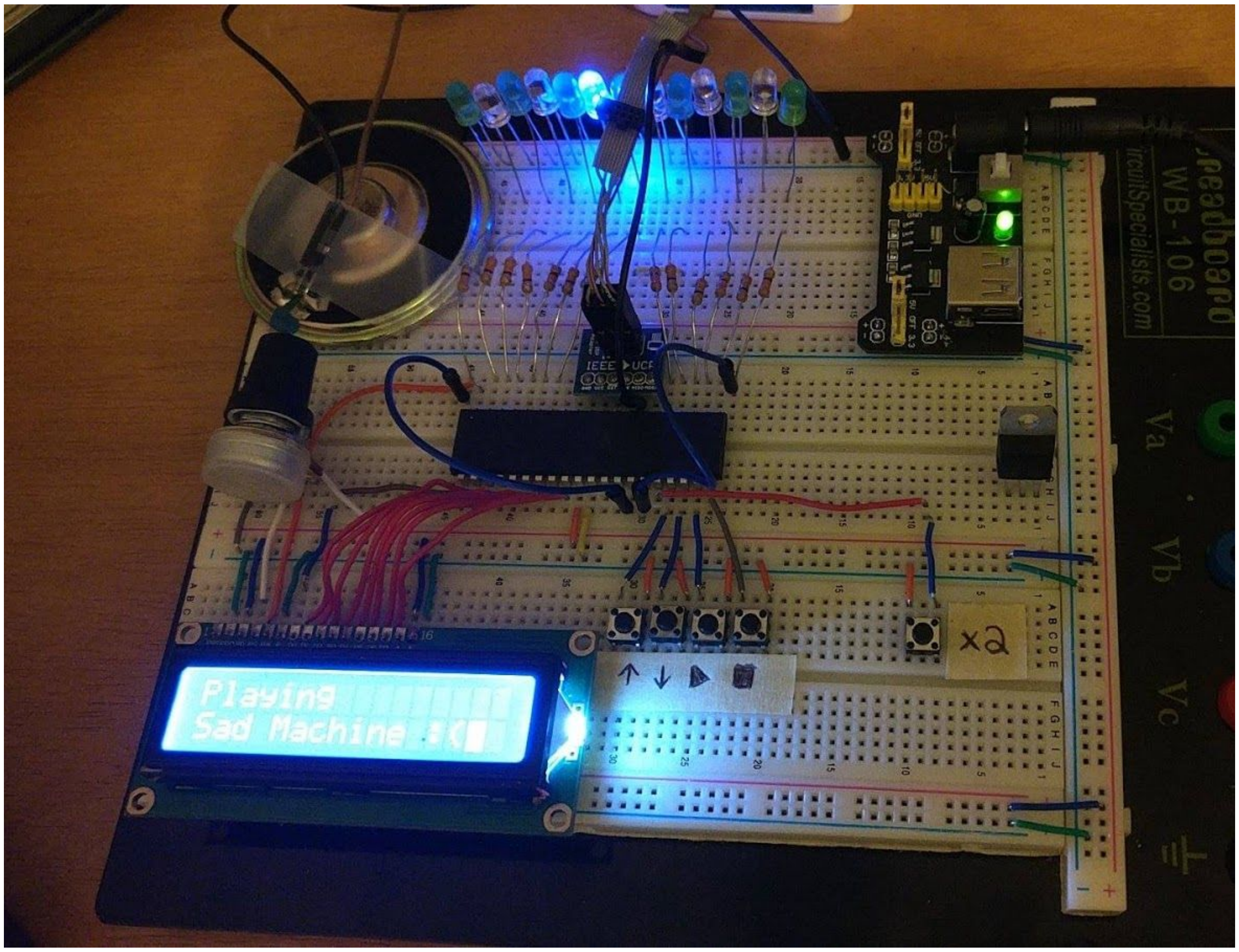
Kevin Nguyen
861069735

120B Custom Lab Project: Jukebox

Overview:
The project consisted of creating a simple jukebox where the user could select a song and play it. The jukebox has an LCD for song selections, a speaker for audio output, festive lights to go with the music, as well as buttons for user input.

Specs:
 Specs for Jukebox Project
LCD Screen in start mode
User presses button to select Song
LCD Screen must display song being played
User may press stop button to stop song
Bonus: Visual Feature that reacts to song being played



Components:

- Breadboard
- LCD
- Speaker
- ATMEGA 1284
- 13 LED’s (white, green and blue)
- 5 buttons
- Potentiometer
- Resistors
- Various wires

Logic and Implementation:

The project was broken up into 4 parts (not necessarily equal) which is reflected in the four state machines. These four parts were the LCD menu interface, the speaker for audio output, lights to react to the music, and the buttons for user input. Each of these were intially tested in the main function for simplicity and then converted to a state machine.

The LCD had 3 main stages, being the start screen with simple instructions to get the user started, the menu where the the user could scroll through the songs, and the now “now playing” display while a song plays. The song titles were placed into an array so that the the user could scroll through them. A same method was used in order to display the song title while it is currently playing. The LCD actually only displays the instruction message once when initially turned on. I went with the assumption that the user would understand after reading it once, so after the user presses an arrow button the LCD will only go between scrolling through the list or displaying the currently playing song. The LCD plays a big part in the system since I created it first, it also sets the flag whenever it enters its “play” state. This flag lets the speaker and LED SM’s to output to the selected song accordingly.

For the speaker and music output, it was a simple state between outputting audio and not outputting audio. A global flag was used communicate between the LCD SM and the speaker SM. Since I had already implemented the “stop” and “playing” states for the LCD to display information accordingly, it was a simple matter of adding the flag so the speaker would know when to output the song. I had first tested outputting songs via strings. Since it was a hassle to type in note frequencies every time, I created a function that took in a character and returned the corresponding frequency. Most of the letters represented their actual notes (‘D’ being a D note while ‘d’ was a D flat). Exceptions were made when I realized that I needed a higher range of notes so I used numbers or letters that didn’t quite match up. In any case it made creating songs efficient sings they were just strings, and timing was acquired through trial and error.

For the LEDs I mapped each of the 13 to two corresponding notes (since there was a range of 26). The LED output was fairly similar to the speaker output, since the SM was also toggling between outputting and not outputting. I used a modified version of the note to frequency function I created to output to corresponding ports instead of returning a frequency.

Buttons were a familiar part of the project. The main parts that I had to deal with were making sure that a held down button registered as one press. The up and down buttons were applied at first since you can't use the stop button if a song isn't playing. Once you press play, the arrow buttons are disabled since your only options are to finish the song, stop it, or toggle the speed and fast forward. The last button toggles the fast forward option previously mentioned; holding it down will double the speed and letting go returns it to normal.

Combining the individual parts required synchronizing "simpler" versions of a SM first. For example once the LCD was up and running, I first worked on just having the speaker output one continuous tone during "play" mode. Once that worked I made further adjustments. Since the LCD sm kept track of which song was currently showing/playing via array, I used the same index for an array of strings which were the notes of the song. Similarly with the light, I started by making them all night up whenever a song was playing, and then made specific adjustments so that only one would play during its note.

Challenges:

There were a few challenges along the way. Aside from the common problems that will forever plague programmers such as missing semicolons, the songs were slightly difficult to write due to the lack of exact note timing ability. For example, a song that may play "C, C, C, C" as four notes in rapid succession would just come off as a continuous tone. This could be solved by placing silences in between the notes but this would result in a choppier melody that would still not sound quite right. As a result the timing of the songs was slightly difficult to get without having control of the little nuances in music rhythm.

For the LEDS, the most difficult part was simply trying to keep track of of the outputs. Since there were 13 LEDs they were spread out through PORTA, PORTB, and PORTD. PORTA is shared with the buttons so I had to keep track of the input and outputs and change the LEDs without somehow affecting button input. The same concept applied to the other ports because some of the pins were also used for other things like the avr controller and LCD screen. It was a matter of using bit manipulation to make sure that I could light up the correct LED without affecting other things.