Johnathan Murad

EECS 120B

08/31/2017

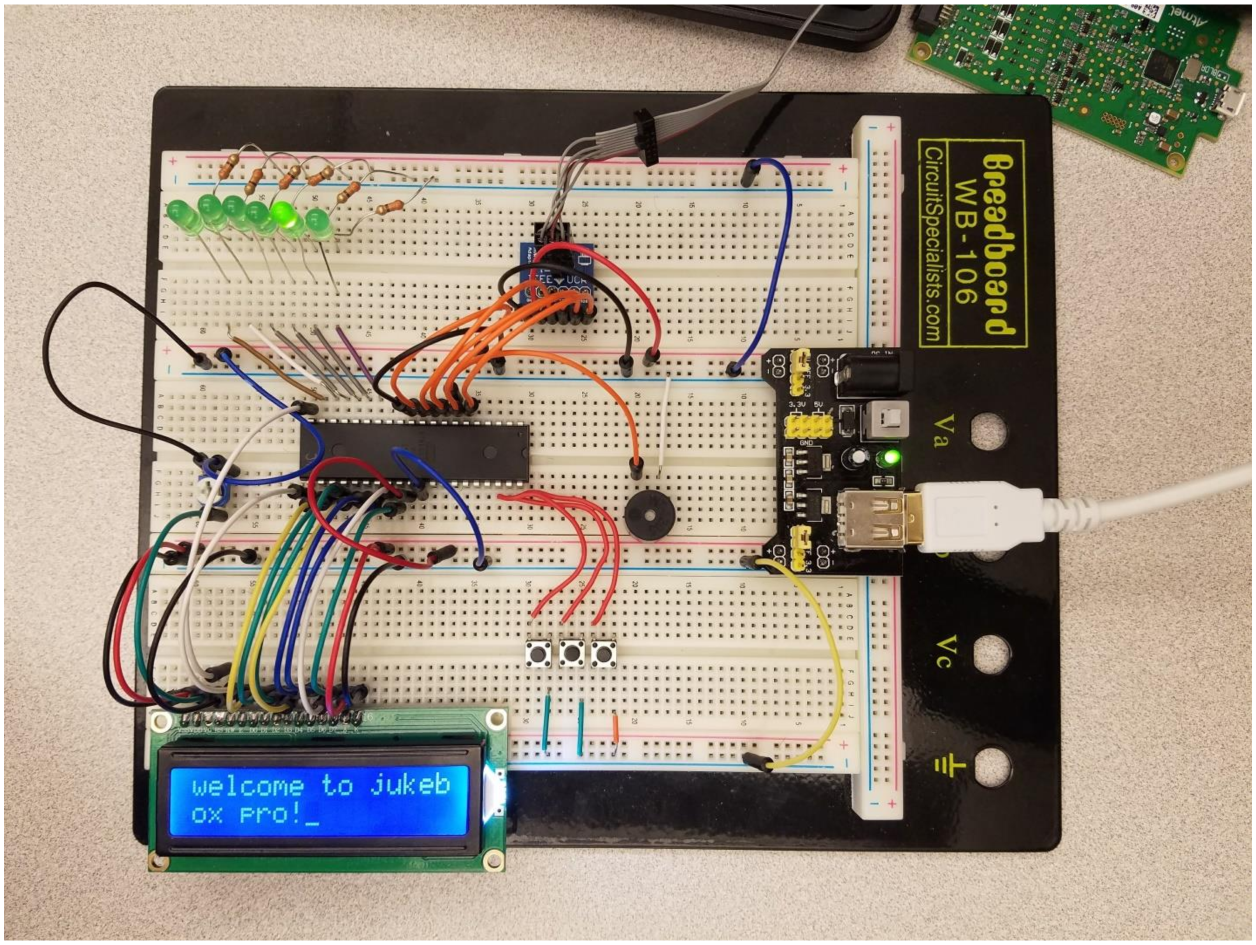
Custom Project

In my custom project for CS/EE 120b I created a juke box. My juke box has the ability to play three songs at a time and be stopped at any time.

In this project, the initial way I approached setting up everything was by first being able to make sure that I can play a couple of notes and make sure that they create the right sound. Immediately I noticed my first error which was that I used type char for frequencies instead of using short. With a char, the size can only reach up to 8 bits whereas short is up to 16 bits. By changing the type from char to short I was able to play all of the frequencies above 255. From there I was able to continue filling in an array with the frequencies I was planning on playing along with the length of each note and the delay in between notes. The song would play unless the user presses a button signaling the system to stop playing and go back to the beginning. I approached creating this project incrementally since it was quite larger and it did require me creating multiple songs. So I also made sure that each portion of music I was adding I was testing to make sure that I was getting the proper sounds and rhythms. One of the more interesting things I did with this project was converting the sheet music to frequencies, and converting the rhythm of the sheet music to the proper lengths and delays. This required some guess and checking but was mostly and accurate way of converting the sheet music to be able to play on the PWM. I wrote each song individually, meaning in other projects, just to make sure that it would play properly then I added them back to the original project to make sure that they would work together without any errors. Of course there were some minor errors with making sure other songs wouldn't play at the same time but eventually I was able to fix these issues and make everything work mostly to my expectations. In this process of fixing errors, I was able to learn how to utilize the handshake method of making requests and acknowledging requests.

For Hardware I used the breadboard given along with an LCD screen, three buttons, a buzzer, and six LED's, and the ATmega1284 microcontroller. The LCD uses up ports D and C, and the LED's are also connected to PORT D. The Buzzer is utilizing the ATmega1284's internal pulse width modulator to convert frequencies to sound. For this to work, the buzzer had to be connected to PB6 on the microcontroller or else the PWM would not work properly. The buttons are connect to PORTA and the three buttons each can play and stop a song. The LED's display a light show in a pattern correlating to the sound being played. I also put the light show within an array and I made it display at the same time that the note was being played so it can match up with the rhythm of the song. The only weird occurrence was the buzzer making a weird screeching sound after programming but that occurrence is normal. The buzzer did work normally however it did get annoying that the buzzer would continuously lose connection or get loose from the board. Overall there were not nearly as many hardware problems as there were software bugs.

Overall, the project was quite fun to build. I enjoyed debugging and putting song's together. I also really enjoyed being able to implement my own songs. In my project I did utilize an Arduino forum for getting all the right frequencies for the right notes. This made reading music and putting it to the code very easy since I had the names of the notes defined.



This is a picture of my working breadboard and the main menu being displayed on the LCD. The light being lit currently on the LED is signifying how the song is currently stopped and waiting for a new song to play.

https://youtu.be/OwE07SZdlR8