Report

Demo Folder:
https://drive.google.com/drive/folders/173Dt0iPtX2V9g_A5oSe08jxe19fHSaDX?usp=sharing

Project Description:

For this project, I have implemented the piano tiles game. The piano tile is a musical game that challenges the player to press the correct button/tile at the correct timing. This requires the system to parse a song into its respective tones and outputs the note to the player. For the player inputs, I have used buttons as the piano keys. Each button corresponds to each key/note. The notes represented in the game are A, B, C, D, E, F. Therefore there are 6 buttons for the player's input during the game and 1 soft reset button.

In order to fit the tiles into the display, I have chosen to represent the stream of music as a string on the bottom row of the LED display. The string shifts leftward as the music goes on. For example, if the music contains the note E -> E -> E -> C. The string will be E_E_E_C and on the next update it will be _E_E_C.(The underscore indicates a pause). The player must press the button associated with the note E before the update in order to gain a point. The player's current point is displayed on the top right of the led display. During a pause, the player does not have to press any button. When a note reaches the left of the display, the speaker will play the sound associated with that note.

In order to load the tiles into the atmega, I wrote a python program using the aubio module to first parse the audio file into a c file with a double array variable named gameSong. This double array contains the note that is played in the song at each time. The array is then compiled with the program and loaded into the atmega flash. For this project, the song that I have chosen to load onto the atmega is jingle bells.

User Guide:

Rules:

The player gains a point if he presses the button associated with the note displayed at the bottom left corner of the LED display at the correct time.

The player loses a point if he does not press the button associated with the note displayed at the bottom left corner of the LED display at the correct time.

The player loses if his point reaches 0.

The player wins if he manages to finish the song.

Controls:

The player has 6 button 1 for each note
The player has 1 button as the soft reset button

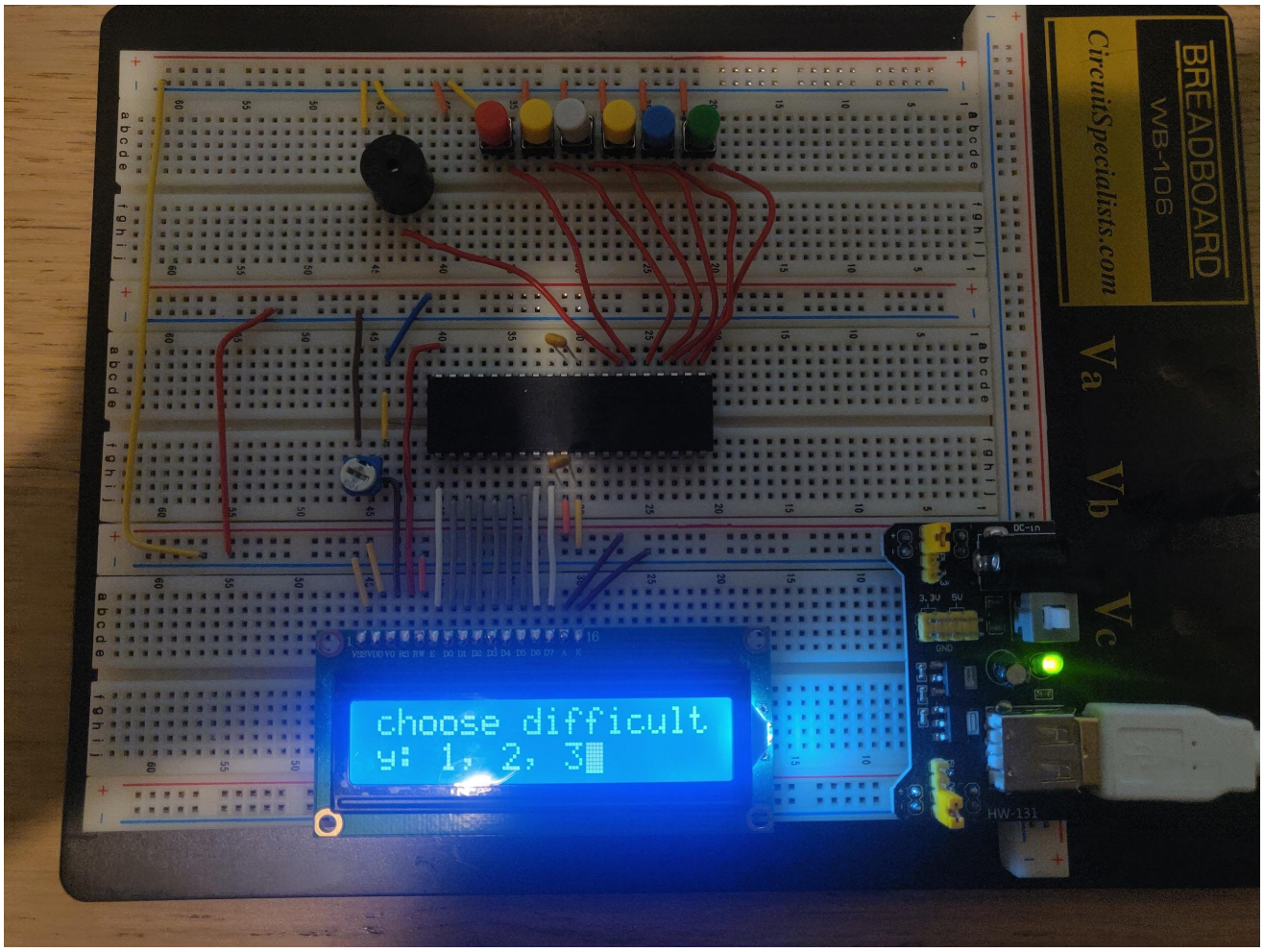
Special Considerations:

The player can choose the speed(1-3) at which the song is played at (difficulty level)
1 being the easiest, 3 being the fastest

Source File Links:

https://github.com/giathuan123/cs120b_embedded_system/tree/game

Component Visualization



Technologies Learned

- Wav file format. During parsing the audio file for the notes in the game, I learned more about the wav format and how audio is formatted.
- Using display buffers. Since the string for the song is very large, I had to use a smaller buffer for the display and update it every tick.
- Aubio module for parsing wav files. I used the aubio module and python to parse the wav file into my desired representation.