Cory Brown

EE 120B Summer 2017

cbrow025@ucr.edu

Jukebox Final Project Report

Physical Description:

* PWM controlled buzzer to create sound
* Array of 6 LEDs to create a reactive light display
* LCD display to show instructions and song names
* Four buttons to control the jukebox: play/pause button, scroll up button, scroll down button, stop song button

Software Description:

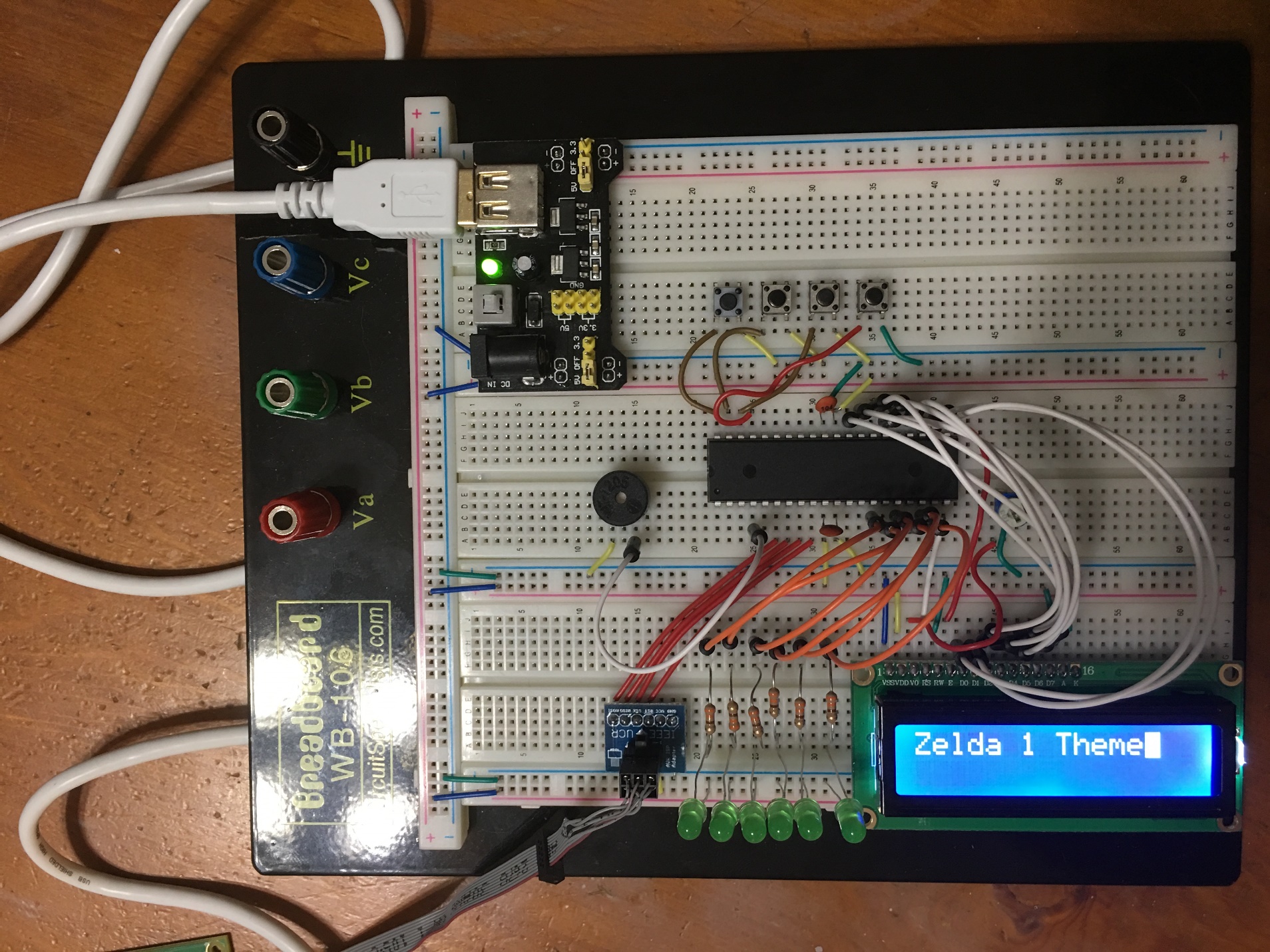
* Synch. SM to control all song playing based on button inputs
* Synch SM to control screen functionality
* Function to control light display

I used one state machine to control the song playing based on button input. This state machine includes the play/pause functionality, song select, and stop song functionality. This state machine is also what handles actually playing through the songs.

Songs are based on a three-array system. One array holds the notes that are going to be played in order, the second array contains how long each note she be held, and the last array contains the time in-between notes being played. This allows for a systematic method for adding songs making the system easily expandable.

The second state machine controls the screen. There is a welcome screen and an instruction screen. The state machine stays on the instruction screen until the first button press. The rest of the state machine doesn't actually need to pay attention to the button presses but simply the variable controlling which song is currently selected. This variable is controlled by the first state machine and shared with the second.

The light-display really didn't need a separate state machine simply a function that looks for which note is currently being played and based on that it decides which LED to light up. The display goes from low pitch to high pitch within a certain threshold. I could have represented more pitches individually if I had used more LEDs. The function is nested in the first state machine in each state that controls playing the current note.

Hardware Configuration:

Hardware Configuration Description:

* PB3 used for PWM output
* PA0-PA3 used for button input
* PD0-PD5 used for LED output
* LCD hooked up using PD6, PD7, PC0-PC7

Challenges:

The biggest challenge during the final project was reading through the Datasheet for the AtMega1284 to figure out exactly how PWM worked so that I could fix the code used in Lab 09 to use PB3 for PWM. The alternative was to use the previous work-around that involved using PB6 for PWM. I did not want to do this because then I would have to unplug the Buzzer every single time I programed the micro-controller.

I now have a much better understanding of how the AtMega1284 creates PWM so I think that it was worth investing time into making sure I had a complete understanding of the functions I was using.

The only other thing to mention is the songs themselves. I ended up using basic songs from old video games. Creating the songs turned out to take much longer than I thought because you have to know all of the notes, and then the timing for them as well as the timing for the rests. I would say this took up most of my time.

A problem I encountered when making songs was with the minimum frequency that the buzzer would output properly. I had to scrap one of the songs because it used frequencies that were lower than what the buzzer could output.

Here is a link to the Youtube video and the Github Repository containing my source code: