Name: Derrick Ho
Login: dho006
email: dho006@ucr.edu
lab section 021
assignment Game Project
Group: Mantej

Game report

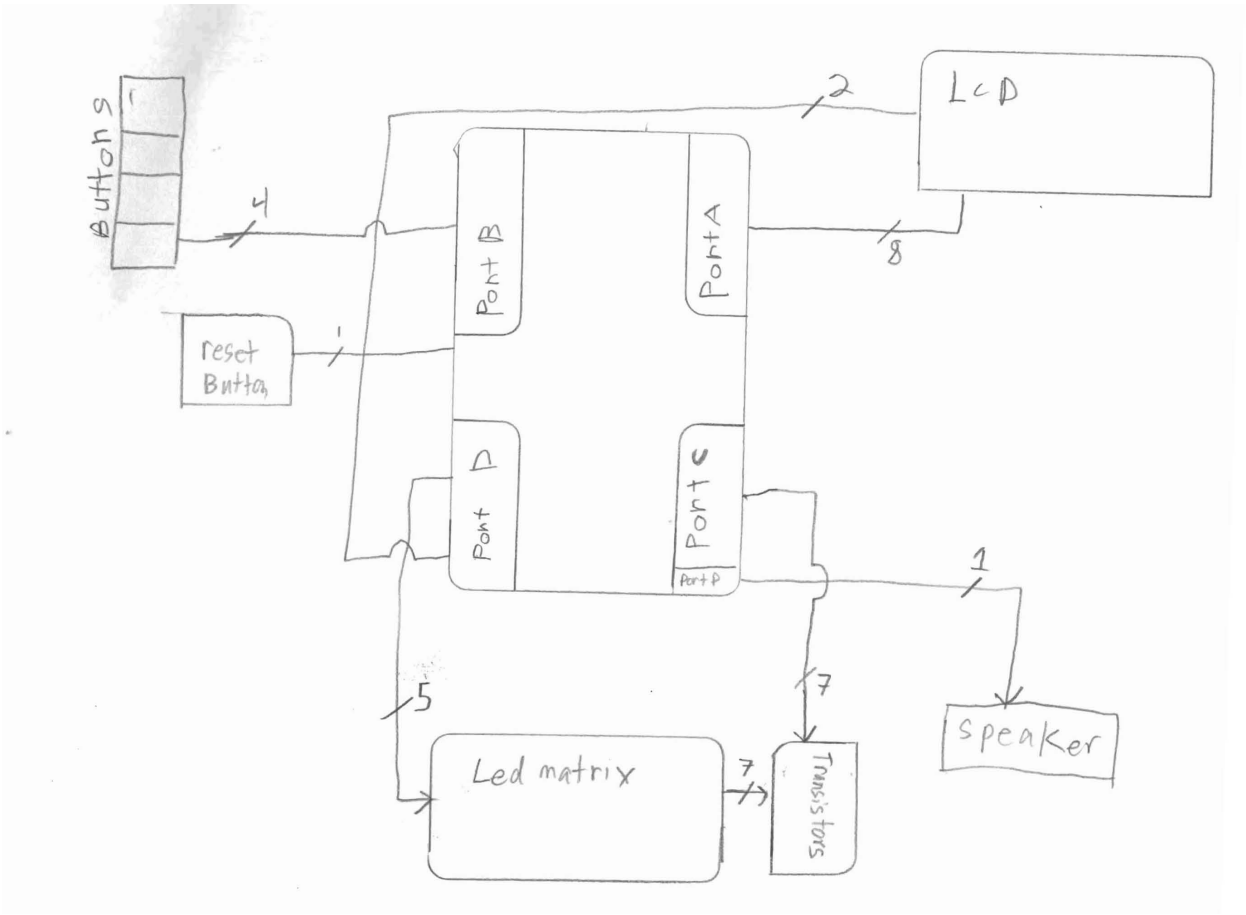
Game Description

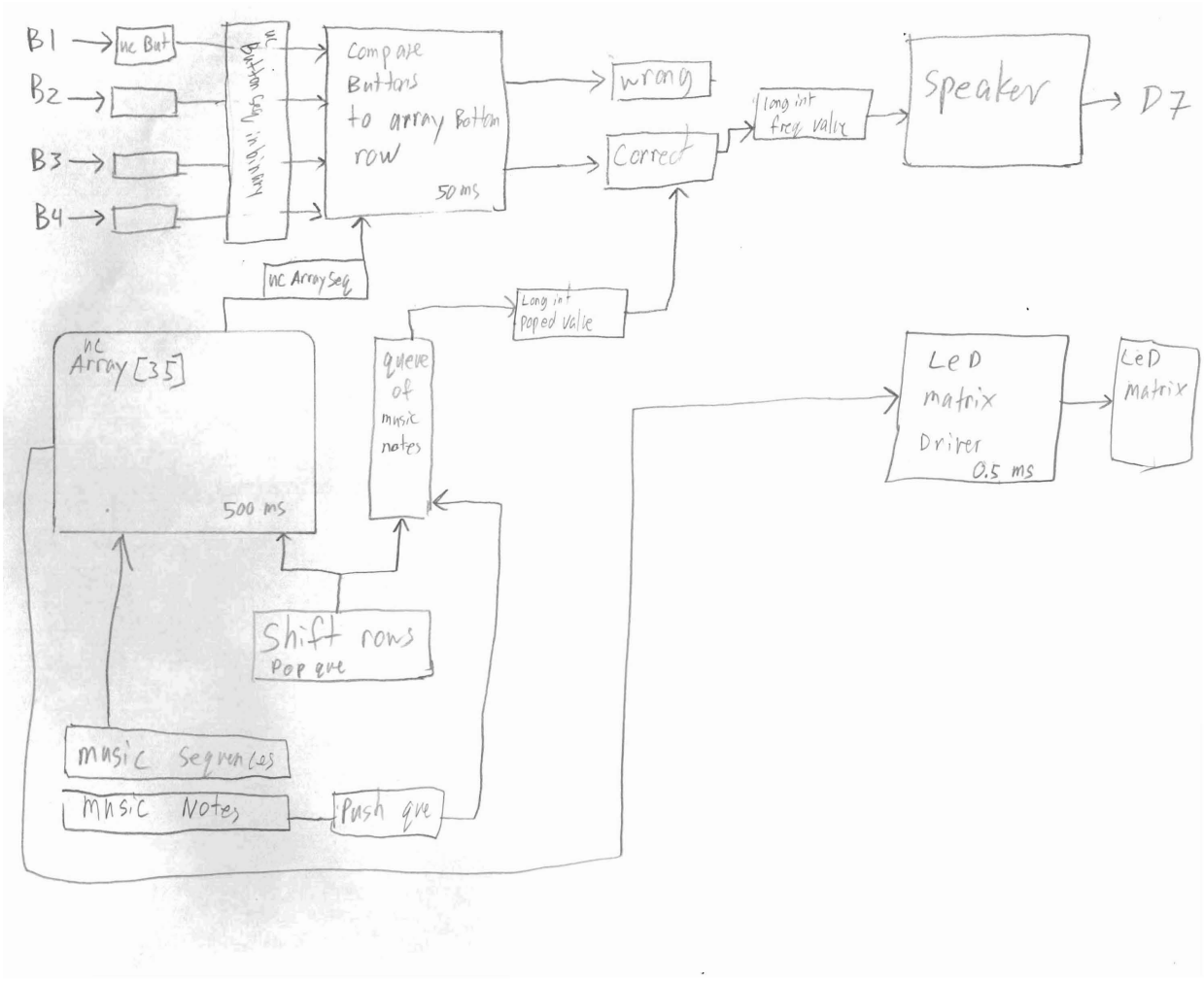
In the game of Guitar Hero, the concept is simple. Notes will start at the top, and shift downwards. This is a simple idea, and was very simple to implement. On every clock cycle, you take each row (each array), and you shift it downwards. When it gets to the bottom, you have to press the right button in order for it to play the sound, other wise you will be met with silence.

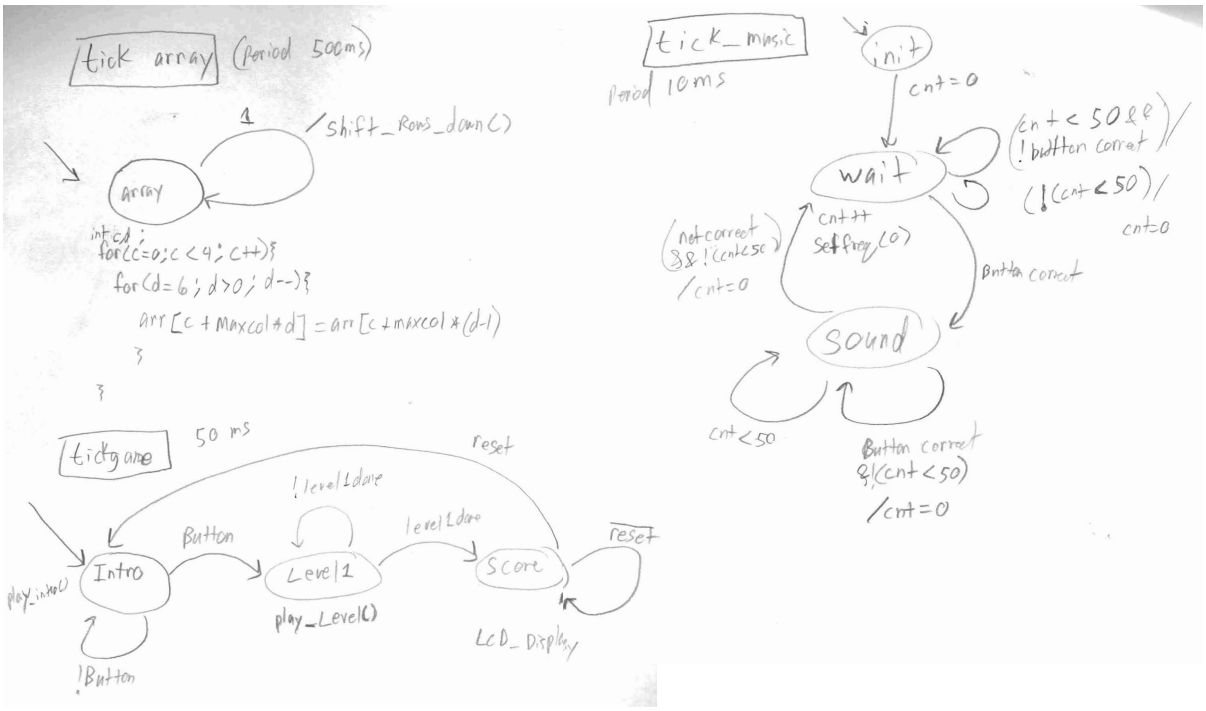
Game Instructions

1) Press the reset button (on the far left) to start a new game.
2) The LED screen will marque "Guitar Hero" in a loop until you press any of the first 4 buttons.
3) The marque title will shift down, and then game will start afterwards.
4) LED lights, or "notes", will start falling downwards from the top of the screen.
5) When the notes hit the bottom-most line, you will have to hit the button that corresponds to the column.
6) Button 1 corresponds to column 1, and so on, and so forth.
7) If you correctly press a button when the light is on, a sound will play, else, there will be no sound.
8) You will know when the game has come to an end when the bottom-right most LED starts to blink.
9) When the LCD light starts blinking, you must press reset to start a new game.

Block Diagrams and High Level State machines


Buttons
4
Port B
Port A
2
LCD
8
reset Butta
1
Port D
Port C
Port P
1
5
7
speaker
Led matrix
7
Transistors


B1
B2
B3
B4
inc But
Button Seq in binary
Compare Buttons to array Button row
50 ms
wrong
Correct
long int freq valie
Speaker
D7
inc Array Seq
long int poped valie
inc Array [35]
500 ms
queue of music notes
LeD matrix Driver
0.5 ms
LeD Matrix
Shift rows Pop que
music Segrences
music Notes
Push que



What was the desired result?

To make a fast paced game where you press button sequences that would correspond to a nice song. In this case, you are playing a song called “To Zanarkand”. When the button sequence is played correctly, you will be able to hear the song as it was meant to be heard.

The goal of the game was to press the right notes. Ever time you miss, you lose points. Ever time you get it right, you gain points. You want to keep your points high other wise, your points will go down and when it goes below a threshhold you loose the game. This would be indicated in terms of stars as indicated by the LCD Screen. Three stars would mean your points are high. One star would mean you are close to losing. Then when the level ends the LCD screen would tell you if you won or lost.

What was the actual results?

A slower paced game where you press button sequences that would correspond to notes being played. When the pace was faster, it would be difficult to play the game, so I made it slower. Because it was slow, the song is very difficult to hear and interpret as a song. Another issue was that the notes would tend to be inaccurate. It would not play the specified frequencies that would correspond to the notes. It would still play a sound, but not right ones. I believe that the problem was that the sampling rate of the sound was too small. If the sampling rate was higher, then the sound would be more right on spot. However, increasing the sampling rate GREATLY increases memory usage. The micro-controller doesn’t have enough space to fit the proper sounds. Plus, we also had the game’s code that took up space as well.

There is no concept of losing in this game. The only reward for hitting the notes is that the song gets played. Missing the notes means no sound. When the level ends,

the game ends. The LCD screen will tell you the game is over and that you need to press reset to play a new game.

Hardware components

Part	Role
one micro-controller	This runs the game and all its logic. It takes in as input, 5 buttons. It outputs a signal to the speaker to play a sound. It sends the commands to print things to the LED matrix. It sends signals to the LCD screen.
one 5x7 LED matrix	Visually shows the placement of the trickling notes.
one LCD screen	Shows a message
seven Transistors	Helps to channel the current. Originally I had PortD to power the columns and PortC to Power the rows. To turn on a column I would send a 1 down PortD and set the PortC to 0. To turn it off I would send a 1 out of PortC and then that would cause the potential difference on the diode to become zero. I felt that it was wrong to output something into another output. So I placed the transistors in there to act like a switch. When switched on, the current would flow and the Diodes would turn on. When switched off the current would be cut off and diodes wouldn't turn on.
five push buttons	serves as input devices to interact with the micro controller. One of them is used as a reset button.
one speaker	Used to play the song

Software components

Part	Role
Task manager	Organize the tasks of different periods. Has them run one after the other to give the illusion of concurrency
LED matrix driver	Looks at the Array Buffer to determine which lights are on. It parses the array and when it detects a 1 it will turn an LED on. If it detects a 0 it will turn the LED off. It works by calling a function which will determine the outputs of PortD and PortC and then outputs them through those ports.
Array Buffer	This is responsible for giving the LED matrix something to read. The Games logic modifies the array to produce different images.
Button unsigned char queue	An unsigned char is used as if it was a queue of 1's and 0's. Every Button tick left shifts the unsigned char. If a button is pressed, a 1 is added.
Game logic	Each row of the LED will shift down once ever 500ms. When the LED's get to the bottom row, the player must press a button and it will output a specific musical note. The sound will play until the rows shift again. If they press the wrong button or they miss, then no sound will happen.
Intro animation	The name "Guitar Hero" is marquee-ed from left to right of the LED matrix
Music	A file contains all the button sequences for the game and it also contains the specific musical notes to play.

Part	Role
Intro animation	<u>About the shirting leftwards, and the downwards shifting in the game</u> In the game of Guitar Hero, the basic concept is simple. Notes will start at the top, and shift downwards. This is a simple idea, and was very simple to implement. On every clock cycle, you take each row (each array), and you shift it downwards. Basically, all the elements of array 7 are equal to all of the elements of array 6, and so on, and so forth. The only exception is the array at the top, which is given new elements. There were no state machines involved in the shifting of bits. It was done basically as described, using for loops to shift the values in arrays. As far as the introduction, a similar design was used, except with bit shifting to the left. We hard-coded the letters "Guitar Hero" into an array. We used 10 arrays, each with 1 letter of the word. During the introduction, on each clock cycle, all columns were left-shifted by 1. Since we had 10 arrays, each with 5 columns, we had 50 columns that need to be transferred to the LED lights. As everything was left shifted, arr 1/50 was stored into the column on the far right. On the next left shift, arr 2/50 was stored into the column on the far right, while arr 1/50 was now in the column on the left of it. This process was continued for all 50 columns, so that the words "Guitar Hero" flashed across the screen.

Difficulties and Challenges

From the beginning the challenge had to do with making everything synchronous. The buttons , the row shifting and the music had to be played at the right time. Sometimes something would jitter causing any other the three to be un-synced. I had to

rearrange the task manager and cut down certain parts of the codes so it would jitter less.

The LCD would cause the led matrix to flicker. And it would also cause the syncing of the above mentioned three to be off. Why would it happen? I think it has to do with the LCD's NOP command which translates to No Operation and i think that would stop the microcontroller for a brief moment. Because of this problem I only used the LCD at the very end and never at the same time as the LED Matrix was running.

The speaker.c file would take up a lot of space because it would create a sine wave. The higher the sampling rate, the better the sound quality. High quality sound means more space to store the values for the sine wave. I wanted really good sound, so I would increase the sampling rate, but lowered it just enough to have space for it.

Labor Breakdown

Derrick Ho

- Hardware composition, Game logic, organizing tasks, state machines, block diagram, sound, LED matrix driver, and Level design.

Mantej

- Intro animation, array blocks shifting down