THE SNAKE GAME

Computer Science 362
Fall 2017

Anusha Pai, Jeff Kaleshi, Ryan Chan and Margi Katwala

UNIVERSITY OF ILLINIOS AT CHICAGO

Table of Contents

Overview 3

Introduction 3

Materials Used 4

Implementation 8

Hardware Design 8

Software Design 12

Issues 14

Conclusion 16

Resources 17

Overview

The concept of the Snake game first originated in 1976 as an arcade game. As a variant, Nokia offered this game on their platform in 1998, which grabbed the attention of a larger audience at that time. Now over 300 snake games are available on any platform with its own variety. These days it's easy to download any type of game you want, whenever you want on your phone but it will still miss the presence of the original snake game. The goal for this project is to recreate the scene of the original game with additional functionality for the user to enjoy the experience.

Introduction

For this project, we have decided to create the "Snake Game". This is a classic game, found on many platforms ranging from Nokia phones to the MMOG version of the game, "Slither.io." Our intuition behind creating this game as our final project is to take our user to their childhood days and recreate their classic memorable days again. We are using physical components to render this digital game. This project contains 4 8x8 LED matrix boards (8x32 board), a 16x2 LCD, a potentiometer, a button, a joystick, a buzzer and a servo. The LCD matrix board is used to display the game. The user will be able to control the game using input devices placed to the right of the board. The goal of the game is to collect as many points/food as possible until the snake hits a wall. To collect points, the user needs to move the snake, via the joystick, in the direction of where the "food" is available. The movement of the snake is directed

through user input with the joystick. In addition to controlling the snake, the user is also given the option to restart the game, which is placed next to the joystick. While the game is playing, messages will be sent over to a webserver via an Ethernet shield. The webserver will either display “Playing” or “Game Over.” Once the game is over, a buzzer will go off and the servo will move 180 degrees with a little penguin on the top.

Materials Used

Materials	**Quantity**
Arduino Uno	4
Arduino Mega	1
8x8 LED Matrix board	4
Ethernet Shield	1
16x2 LCD display	1
Joystick	1
Potentiometer	1
Passive Buzzer	1
Servo	1
Button	1
Wires	~200

The materials we used for this project were more locally found within our Arduino kits. We first ordered 4 8x8 LED Matrix boards and servo. We were the most unfamiliar with the LED Matrix boards due to the confusing connection. In addition, we decided to implement the game using four Arduino Unos and one Arduino mega for the I2C connection. The four Arduino Uno given its name respectively as slaves and the Arduino mega was our master. As shown in

figure 1, our first slave contained a joystick, a button, and a potentiometer. The joystick was used to change the direction of the snake while the button was used to reset the game. The potentiometer was used to control the brightness of the LCD screen. Slave two, as shown in figure 2, contained our 16x 2 LCD display and the 8x32 LED matrix. Moving on, slave 3 was in charge of the buzzer, which takes in the game status, via I2C, and plays the sound if the status was *GAME OVER.* The last Arduino was in charge of the servo and Ethernet shield. Similar to Slave 3, this also received events via I2C and prompted the Ethernet shield to display “Playing” or “Game Over” and prompted the servo that the status is *GAME OVER* and it is time to move the penguin. This encourages our user to try again. Nowadays, every kind of accomplishment is posted on social media, so we decided this little happiness should be also displayed over the web server and this was done using the Ethernet shield. The master Arduino mega, the microcontroller board, in this game handled the event occurrence of the game. The Mega deals with the communication between the slaves. The slaves do not directly talk to one another, rather the slaves talk to the master and the master sends desired information to other slaves. The master will also request information from the slaves. This allows for an event-based implementation for our game.

Figure 1- Slave 1

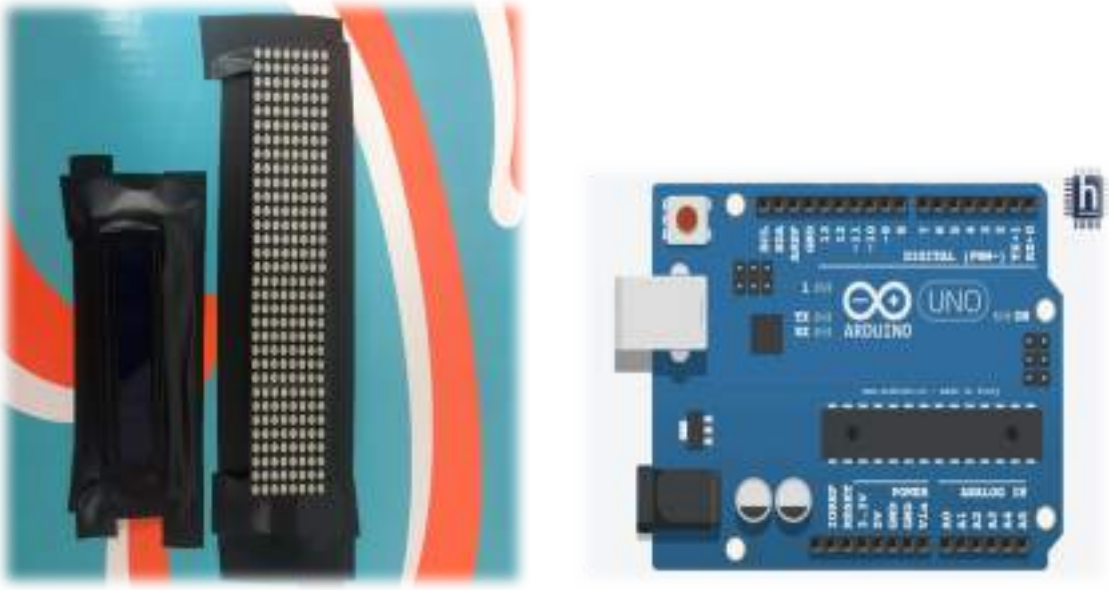

Figure 2 - Slave 2

Figure 3 - Slave 3

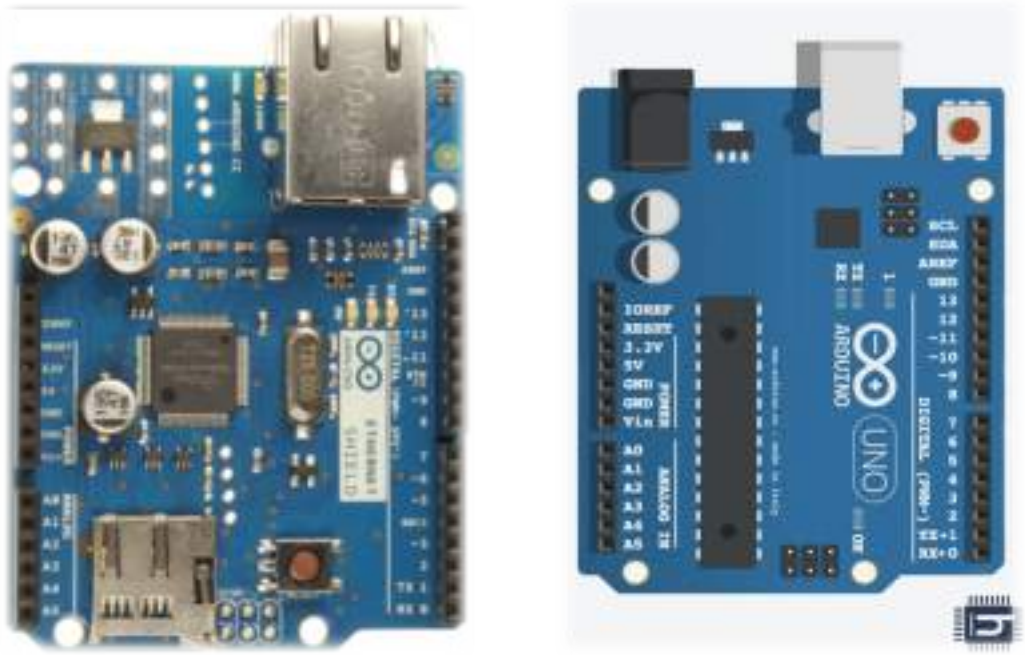

Figure 4 - Slave 4

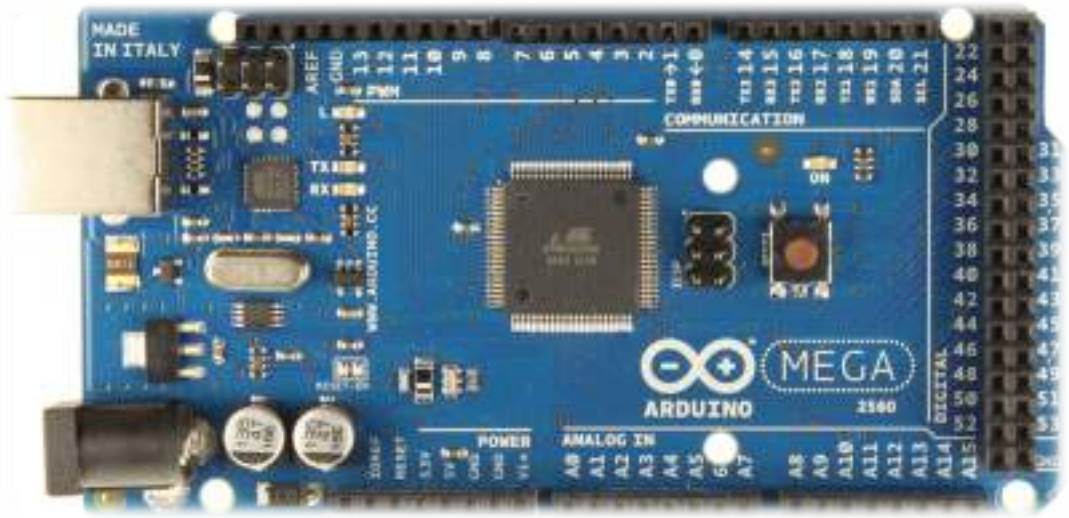

Figure 5 - Master

Implementation

Implementation of the snake game was done in the various steps, which was divided in a few months of work. Our first step was to test our devices with the code, given from the manufacturer when they were bought. The next step was our hardware design implementation, which was getting the screen for the user to play the game. Then second stop was to design the software design for the game. Our last step was to test both steps with different test cases in order to make sure a smoothly running game.

Hardware Design

To implement this game efficiently, we divide up the work depending on tasks for each device, as well as how many pins it takes on an Arduino Uno. First, we started off with a big empty shoe box, where our board will be placed inside the box and our screen for playing on top of the box. Then we punched five holes to make an outlet which connects the Arduino to the computer. Each of these Arduinos are connected to breadboards via wires, as shown in figure 6. With our mega Arduino, we place a small breadboard, connect three pins from each Arduino to two digital pins on mega and one to the *GND* to establish the I2C connections.

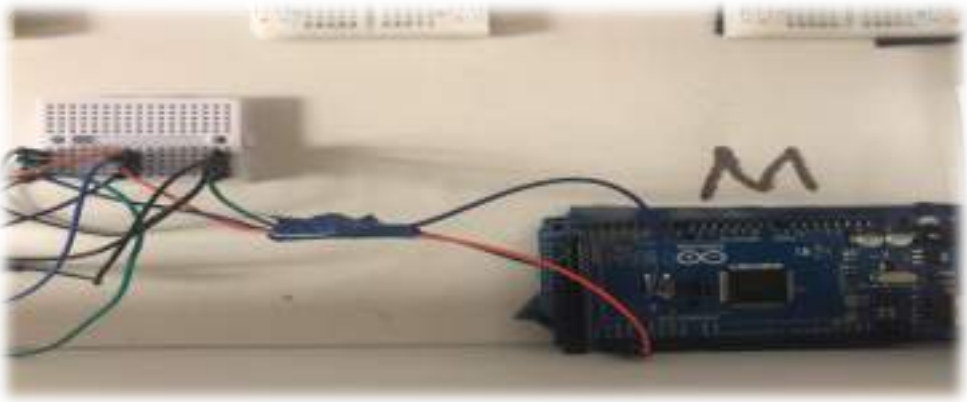

Figure 6 - Steps of the hardware design

To make sure each Arduino was overloaded with events and functionality, we made sure the work load of each Arduino is equal throughout our hardware design. Figure 7 shows our slave one. This slave was responsible to work with the joystick and the button. In this scenario, the Joystick was connected with two analog pins A1, A2 and the digital switch pin at 4. On the other side, button took only one digital pin which was pin 2. The complete connection is shown in figure 7, where we have implemented our slave one. Figure 8, shows our slave 2, which takes in the LED matrix and the LCD display on the Arduino. Our pins for this Arduino taken by the LCD display are in order. We included the Liquid Crystal libraries and it was initialized as *LiquidCrystal lcd (8, 9, 10, 11, 12, 13).* The LED matrix takes in three pin called Data in Clock pin and last Load. In our implementation, we connected Data in Into Pin 2, Clock pin to 4 and load to pin 3. Our slave three takes care of the buzzer which is activated at the end of the game to play a melody sound. The buzzer is connected to pin 2 in slave three which is shown in figure 9. Our last Arduino Uno connects with the servo and also has the implementation of the Ethernet shield, shown in figure 10. The servo is connected to the pin 3 and the Ethernet shield on the top of the Uno Arduino. There is one connection which is consistent overall through our hardware

designing is connected the two Analog pins, connected to A4, A5 and the ground pin with each Arduino connected to the master Arduino. The schematic to that implementation is shown in figure 11.

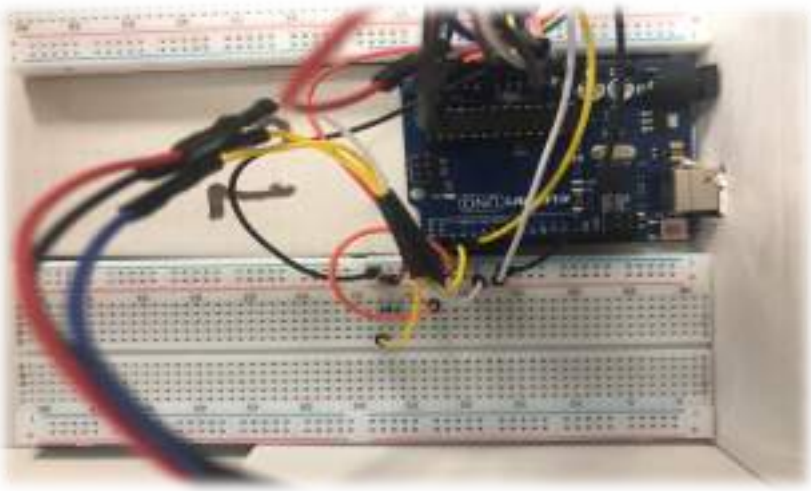

Figure 7 Slave 1 - Joystick and a button

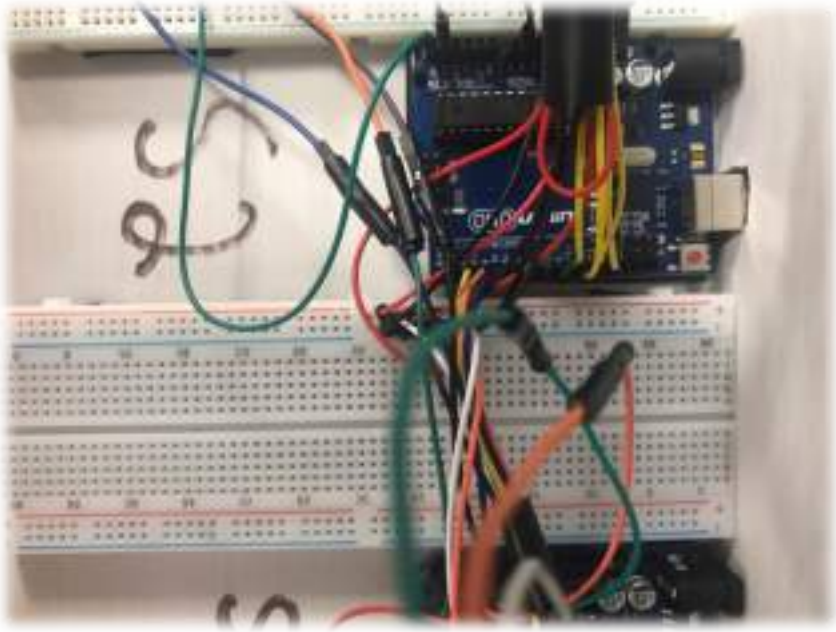

Figure 8 - Slave 2- LED matrix board and the LCD display and the potential meter

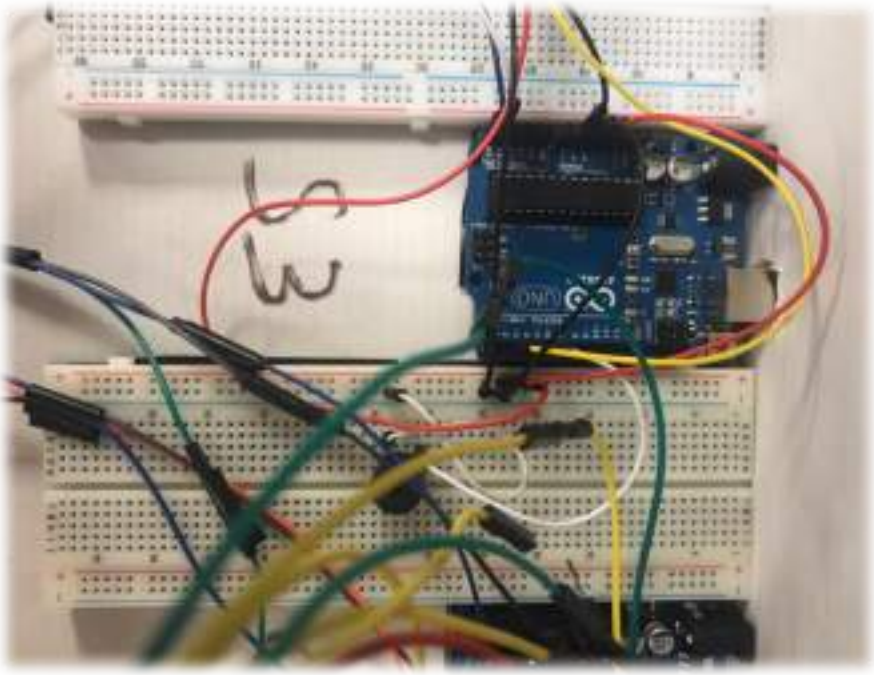

Figure 9 - Slave 3- Passive Buzzer and Servo

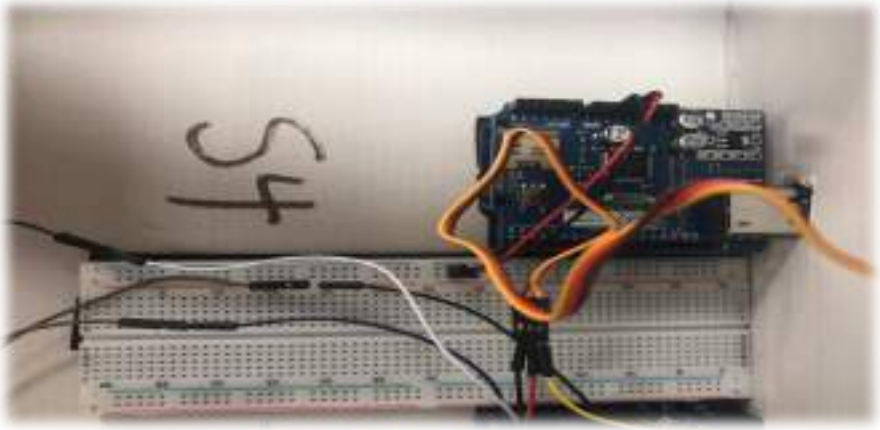

Figure 10- Slave 4 -the Ethernet Shield

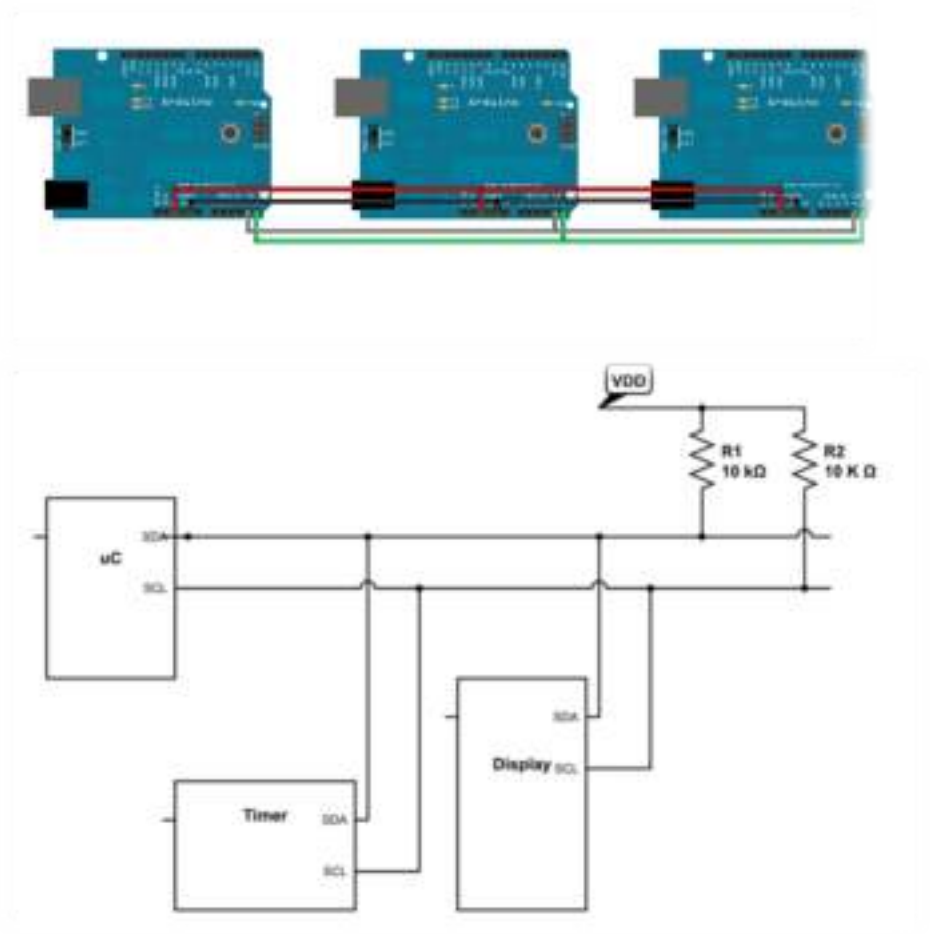

Figure 11- Master Arduino Mega - Schematic

Software Design

There are numerous ways to implement this game in Arduino IDE using programing language C. After our hardware design, we came up with several few ideas to program this game and have an efficient working game. To start off, we played with several inbuilt libraries such as *Link- List*, *FastLED*, *LedControl*, *MD_MAX72XX* and the *Servo*. These libraries were mainly used to work with the LCD matrix board, the servo and the snake implementation using the Link- List. The pixels from the LCD matrix board were stored into a 2-D array and the original snake was stored into a Link-list. The way we were moving snake was changing the remove the

pixel form the tail and head it to the head of the link-list and then change the direction by changing the rows and the col depending on the joystick receiving the event. While implementing this design we came across many issues such as by refreshing by itself, taking its time receiving the movement and also delaying the web server to send in the code.

After all of these issues we optimized our code by using the libraries same as earlier but also adding few more to the count. In order to make our code efficient we implemented the libraries in C++ language. The additional libraries that we added in our design two were *coordinate, Direction, Status and Snake.* To overcome the buffer overflow issue, we made a *Snake* library. This library does some basic function for the grid such as Initialize game settings, initialize grid array to all false, start snake at (0, 0), Add food to grid, get the snake coordinates, moves the snake, gets direction, updates the display and also snake stops when game over. In Status library we have implemented an enum of all the game status such as *GameOver , Game_Playing, Success, Spin_Motor, GameMode* and *Reset.* Now in our original five different files for each Arduino we only had to make function calls to from the libraries. So, the setup for each Arduino is similar to each other. Master Arduino has two loops one to check if the button was pressed and another one to play. In a loop master will call the play() which then using the I2C connection it begin transmission on slave 1. Which then gets the input from the joystick which is in slave one and then continuously requesting event using *Wire.requestFrom(2, 1) //2 is the slave name given* until the game is over. Once the Game_Over status is received then the master will send in request to pull the event from slave 3 and 4 which then perform their own events. The slave three was in charge of working with the buzzer and the servo. The slave four, worked with the Ethernet shield where the status of the game is displayed on the web server. The way we transferred our *Game_Mode* was using the Ethernet Shield Library which is mainly

useful to answer HTTP request. In order to open the browser, the user needs to input the Ethernet Shield's IP address which is the six analog pins. Our design in slave four was to listen for the game mode from the slave two *(client)* and if it's connected to the server which is then using print line we can perform our display for the web browser. Our second design was more consistent than any other design because it was more optimized and more logically easy to debug if any problems occurred in future. Using mixture of two languages and also using more strategic planned code helped us to successfully present our project.

Issues

There were several issues that we encountered during this project. Due to that we made quite few changes to our original design of the game. The first issue we came across was the 32x32 LCD matrix board turned out to be defective. When we first received it from *Adafruit* website it lit up properly but after the first run it started to display everything doubled on the screen. While talking to company representative, we found out that by mistake they sent out some manufactural defective items to their customers. In figure 12, it shows the difference of display changed after replacing the LCD matrix board. To resolve that issue we ordered those four of 8x8 LCD matrix and redesigned our display board for the user. Our initial plan with the snake game had ideas to use camera to take picture at the end of the game and make a tweet on the *Twitter*. The issue that we came while implementing the camera was that the camera we received from the online order turned out to be a 12 pins camera, shown in figure 13. There were no instructions neither in their manual or online that shows the connection of a 12 pins camera to the Arduino. To replace the camera, we installed the Servo which makes a spin with its little penguin on the top. We also weren't able to send the message on Twitter to replace that we used

The Ethernet Shield to send message on the server. As we talked about the software designing part for this project, we ran into several issues such as buffer overflow. While the game was running, it will randomly re-start the game. For days, that error was unknown to us as there is no error shown while compiling. After a while, we started to replace all the game handling code separated into several different files. The way we optimized the code reduced this error and made our game run more efficiently. Another change we made while implementing the game was the use of the Potentiometer. First, we were planning on implementing different levels of the game which the user can change it using the Potentiometer. The issue occurred here was the way we designed our software we were unable to change the speed of the snake because the way snake was implemented in our code. We implemented snake into a Link-List, the food into Link-List and the pixels of the board into 2-D array. Due to the implementation, we were unable to change the speed of the snake but instead we used the Potentiometer to change the brightness of the board. There were also small issues such as the user needed to hold the joystick in order to make the snake move on time or the buzzer not receiving the signal on time. These small issues were resolved by making the joystick more sensitive of the direction and replacing the buzzer with a new one passive buzzer. Overall, there were wide-ranging issues that we came across which varied from software to hardware issues. Although, few good adaptions helped us to give our unique implementation in the game.

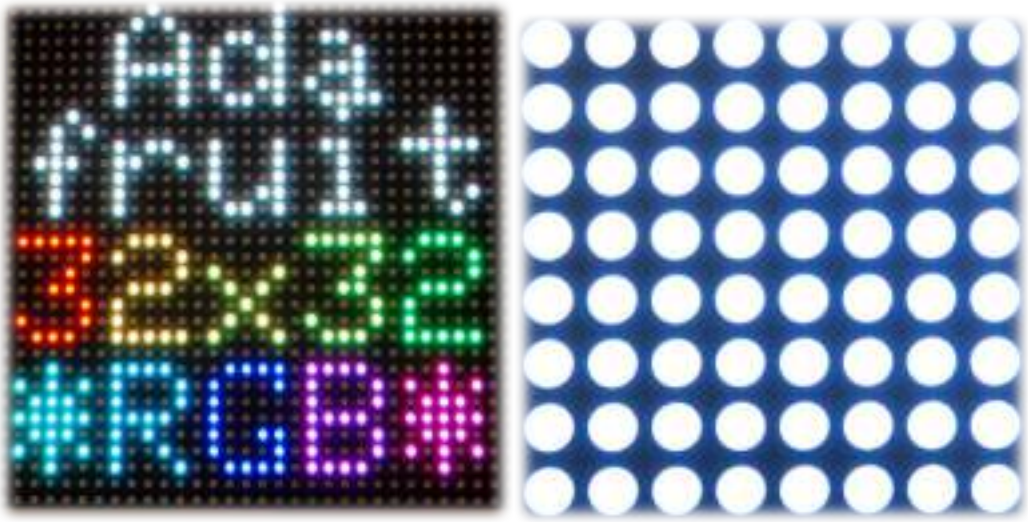

Figure 12 – Matrix board replacement #Issue 1

Figure 13– Camera replacement #Issue 2

Conclusion

In conclusion, we had an amazing time creating the Snake Game. Although we had issues with the connections and the ordered products, we successfully recreated the well-known Snake Game.

Resources

Implementation of 8x8 matrix
educ8s. “Arduino Tutorial: 8x8 RGB Led Matrix with WS2812 Driver with Arduino Uno from Banggood.com.” *YouTube*, YouTube, 27 Feb. 2016, www.youtube.com/watch?v=T-LYYBJsu4Y.

Arduino LCD Matrix library
majicDesigns. “MD_MAX72XX.” *Arduino Libraries*, 4 Dec. 2017, www.arduinolibraries.info/libraries/md_max72-xx.
Learning more about I2C connection
SFUPTOWNMAKER. “I2C.” *I2C*, learn.sparkfun.com/tutorials/i2c.

banksia. “How to Connect Multiple i2c.” *Electrical Engineering Stack Exchange*, 19 Jan. 2012, electronics.stackexchange.com/questions/25278/how-to-connect-multiple-i2c-interface-devices-into-a-single-pin-a4-sda-and-a5.