CryptoColor

Team: Sunfire

Name 1: Mike Apreza	**NetID:** maprez3	**Email:** maprez3@uic.edu
Name 2: Hsien-Hao Chang	**NetID:** hchan43	**Email:** hchan43@uic.edu
Name 3: Xiudeng Zhang	**NetID:** xzhan82	**Email:** xzhan82@uic.edu
Name 4: Xiao Chen	**NetID:** xchen247	**Email:** xchen247@uic.edu

Abstract: CryptoColor consists of a main game controller that runs the game with three players. First, the controller displays a welcome message and wirelessly connects to the players' controllers. Then, the controller randomly selects a word from its library before lighting up the appropriate number and colored LEDs according to the code where consecutive letters of the alphabet are assigned a color. The players have limited time to guess the word according to the code. If a player guesses correctly in time, they earn a point. Else, they earn nothing. The game can be restarted at any time.

Project Description:

Overall Description: The project is based on the board game called "Code Stack!". Just as in the original game, certain colors represent certain consecutive letters of the alphabet. Unlike the original game, there is a master Arduino that automatically controls the game (i.e selects the next word, assigning players their points, keeping track of time, deciding if the word guessed by the player is correct, etc.) without depending on a player to control it beyond resetting the game. The master Arduino will select a 3- to 8-lettered word and display the appropriate colors according to the code. Then, it displays the time the players have to guess the word.

The players input their guess using the LCD display (to see the guess), joystick (to change and select characters), and buttons (to delete or submit their guess). Those who guess correctly in time will receive a point and see it reflected on their screen while their green LED lights up. Those who do not, receive nothing, which is shown on the display as their red LED lights up. The first who reaches 10 points wins the game before the whole system restarts.

Communication Mechanism: The master Arduino and players' Arduinos will communicate with each other wirelessly via NRF24L01 modules. The master and players' Arduinos send a data package that is made up of a struct containing a character array, three integers, and a boolean. Using this form of communication does not require the use of more pins that are in one arduino. Also, it allows for multiple receivers and transmitters, which is required in the project; The master Arduino needs to send to and receive data from the three player Arduinos. Some other forms of communication would not work as the project needs, and may require more pins than available. The rest of the communication required between an Arduino's I/O devices will be done within itself.

Design with Inputs/Outputs:

1. Master Arduino: This subsystem waits for a few seconds to let the players have their controllers on and display a message stating that the game is going to start soon. The subsystem will randomly select a word from its dictionary. According to the letter, the neopixels will light up in that order where the bottom lit LED corresponds to the first

letter of the word, the one above it being the second letter, and so on. After, the timer will start at 80 seconds and send data to the players to activate the guessing system for player input. It receives submitted guesses from players and checks them for correctness. If correct, the controller returns a point for the player, which lights up their green LED. Otherwise, 0 is returned, and their red LED lights up, signifying that the player loses the round. All of this will be done within the time limit. When the time reaches 0, it will select another word for the next match. If the button on the master Arduino is ever pressed, the game will restart. The player who reaches 10 points wins, making the controller send everyone data back with the gameOver variable equal to true. Mike is responsible for this.

a. Input Devices:
    i. Button: When pressed, the controller will send a value to the players via the data package to reset the players' Arduinos before resetting the master Arduino itself.
    ii. NRF24L01 Module: Receives data (guess, ID, score, reset value, gameOver boolean) from a player.
b. Output Devices:
    i. LCD Display: Communicates with the players welcoming them, letting them know that the game is going to start soon, letting them know that a word is being selected, and displays the countdown, which is updated every second.
    ii. NeoPixel Stick - 8 x 5050 RGB LED Strip: When the word is selected, the program will light up the appropriate neopixel in the color corresponding to each letter as stated by the code.
    iii. NRF24L01 Module: Sends player data (guess, ID, score, reset value, gameOver boolean).

2. Player 1: This subsystem will connect to the master Arduino. When all players have connected, the game will start, and the controller will receive a data package that contains an integer to let the player start guessing. Moving the joystick up or down changes the current letter while pressing it moves the letter down the display, signifying that it is the next letter the player chose for their guess. If button one is pressed, the second line of the LCD display, where the guess is, is cleared. If button two is pressed, it will send the player's guess along with other information to the controller via the wireless module. The score from the package will be received and modified by the master Arduino according to whether or not the player's guess is correct. The player receives this modification. If the score is one point higher, the display tells the player that a point is earned as their green LED is lit. Otherwise, the red LED is lit while the message on the LCD display communicates that the player's guess was wrong. If they reach 10 points, they will be let know that they won the game. Hsien-Hao is responsible for it.

    a. Input Devices:
        i. Button 1: When pressed, the player's guess will be deleted on the LCD display.

ii. Button 2: When pressed, the player's updated guess and other data will be sent to the master Arduino.
iii. Joystick Module: When moved up, the current letter will be increased. When moved down, the current letter will be decreased. The letter will be changed in the LCD display. When pressed, it moves the letter to the second row of the LCD display letting the player know that the letter has been selected.
iv. NRF24L01 Module: Receives data from the master Arduino, letting the player guess, restart the controller, and see the new score.

b. Output Devices:
i. Red LED Light: Lights up when the player guessed incorrectly.
ii. Green LED Light: Lights up when the player guessed correctly.
iii. LCD Display: Displays the player's score and guess.
iv. NRF24L01 Module: Sends data (player's guess and ID) to the master Arduino.

3. Player 2: This subsystem will connect to the master Arduino. When all players have connected, the game will start, and the controller will receive a data package that contains an integer to let the player start guessing. Moving the joystick up or down changes the current letter while pressing it moves the letter down the display, signifying that it is the next letter the player chose for their guess. If button one is pressed, the second line of the LCD display, where the guess is, is cleared. If button two is pressed, it will send the player's guess along with other information to the controller via the wireless module. The score from the package will be received and modified by the master Arduino according to whether or not the player's guess is correct. The player receives this modification. If the score is one point higher, the display tells the player that a point is earned as their green LED is lit. Otherwise, the red LED is lit while the message on the LCD display communicates that the player's guess was wrong. If they reach 10 points, they will be let know that they won the game. Xiudeng is responsible for it.

a. Input Devices:
i. Button 1: When pressed, the player's guess will be deleted on the LCD display.
ii. Button 2: When pressed, the player's updated guess and other data will be sent to the master Arduino.
iii. Joystick Module: When moved up, the current letter will be increased. When moved down, the current letter will be decreased. The letter will be changed in the LCD display. When pressed, it moves the letter to the second row of the LCD display letting the player know that the letter has been selected.
iv. NRF24L01 Module: Receives data from the master Arduino, letting the player guess, restart the controller, and see the new score.

b. Output Devices:
i. Red LED Light: Lights up when the player guessed incorrectly.
ii. Green LED Light: Lights up when the player guessed correctly.

iii. LCD Display: Displays the player's score and guess.
iv. NRF24L01 Module: Sends data (player's guess and ID) to the master Arduino.

4. Player 3: This subsystem will connect to the master Arduino. When all players have connected, the game will start, and the controller will receive a data package that contains an integer to let the player start guessing. Moving the joystick up or down changes the current letter while pressing it moves the letter down the display, signifying that it is the next letter the player chose for their guess. If button one is pressed, the second line of the LCD display, where the guess is, is cleared. If button two is pressed, it will send the player's guess along with other information to the controller via the wireless module. The score from the package will be received and modified by the master Arduino according to whether or not the player's guess is correct. The player receives this modification. If the score is one point higher, the display tells the player that a point is earned as their green LED is lit. Otherwise, the red LED is lit while the message on the LCD display communicates that the player's guess was wrong. If they reach 10 points, they will be let know that they won the game. Xiao is responsible for it.

    a. Input Devices:
        i. Button 1: When pressed, the player's guess will be deleted on the LCD display.
        ii. Button 2: When pressed, the player's updated guess and other data will be sent to the master Arduino.
        iii. Joystick Module: When moved up, the current letter will be increased. When moved down, the current letter will be decreased. The letter will be changed in the LCD display. When pressed, it moves the letter to the second row of the LCD display letting the player know that the letter has been selected.
        iv. NRF24L01 Module: Receives data from the master Arduino, letting the player guess, restart the controller, and see the new score.
    b. Output Devices:
        i. Red LED Light: Lights up when the player guessed incorrectly.
        ii. Green LED Light: Lights up when the player guessed correctly.
        iii. LCD Display: Displays the player's score and guess.
        iv. NRF24L01 Module: Sends data (player's guess and ID) to the master Arduino.

Original Work:

Similar Projects: As of the date this report is due, there have been no similar projects to ours in any way. Obviously, there are tutorials on how to use certain parts on Arduinos (i.e. connect a LCD display, read a button's value, etc.), but those do not count. The following projects have been found, which implement a somewhat similar game:

"Arduino LED Guess Word Game" (viiicky0309).

For the LED Guess Word Game from viiicky0309, the project does not contain any display of colors and display of messages. Also the project doesn't contain the use of a wireless transceiver module. Besides that, the project doesn't have the controller designs as we do, meaning the playing experience will be massively different compared to ours. In terms of similarity, color stack game and guess word game are both looking to puzzle out the word, the first one to figure out the secret word will be the winner.

"A Simple Guessing Game" (Thesharanmohan).
For the other guessing game arduino project, the original only uses one board, meaning this one does not have any subsystem. On the other hand, our project uses NRF24L01 Module to connect other 3 players' subsystems to play the game, and we limit the time, players only have 80 seconds to play the game, so they need to hurry. We use a joystick to make our choice, and then the main system will check the correctness, then feedback the result to the player. Our project can display different colors to indicate the character. If one of the players reaches 10 points, they win the game.

Description of the Original Work: The creation of the original board game into an electronic version has not been done as far as we can tell, which is pretty common for a lot of games. Unlike the original game, we are setting a time limit to let players guess, the chosen words will be from 3 to 8 characters in length, and the word chosen will be done by an electronic entity. The latter allows all the players to play without anyone having to sit out to be a word-chooser. Not all the original rules are included. This project has made the game less physically-interactive.

List of Materials:

- For Master Arduino:
  - 1 Arduino UNO Board,
  - 1 USB Cable,
  - 1 Breadboard,
  - 1 Push Button Switch,
  - 1 LCD Display,
  - 1 10K Ohm Potentiometer,
  - 1 220 Ohm Resistor,
  - 1 10K Ohm Resistor,
  - Wires,
  - 7 Female-to-Male Dupont Wires,
  - 1 NeoPixel Stick 8 x 5050 RGB LED,
  - 1 NRF24L01 module
- For EACH Player (total of 3):
  - 1 Arduino UNO Board,
  - 1 USB Cable,
  - 1 Breadboard,
  - 2 Push Button Switch,
  - 1 LCD Display,
  - 1 Joystick Module,

- 1 10K Ohm Potentiometer,
- 1 Red LED,
- 1 Green LED,
- 2 220 Ohm Resistor,
- 1 330 Ohm Resistor,
- 2 10K Ohm Resistor,
- Wires,
- 7 Female-to-Male Dupont Wires,
- 1 NRF24L01 module

How to Build:

Main Arduino:

1. Gather the materials.
2. Insert the button, LCD display, potentiometer, and NeoPixel Stick on the breadboard such that each pin, or column of pins, has its own row.
3. Connect a 10K Ohm resistor on one side of the button to ground (negative side).
4. Connect a 220 Ohm resistor on the same row as the LCD display's A pin.
5. Connect a wire from the 5V pin in the Arduino board to power (positive side).
6. Connect a wire from the GND pin in the Arduino board to ground.
7. Connect a wire one end of the potentiometer to ground. Connect another wire from the other end to power.
8. Connect a wire from the middle pin of the potentiometer to the LCD's V0.
9. Connect a wire from the LCD's VSS to ground.
10. Connect a wire from the LCD's VDD to power.
11. Connect a wire from the LCD's RS to digital pin 7.
12. Connect a wire from the LCD's RW to ground.
13. Connect a wire from the LCD's E to digital pin 6.
14. Connect a wire from the LCD's D4 to digital pin 2.
15. Connect a wire from the LCD's D5 to digital pin 3.
16. Connect a wire from the LCD's D6 to digital pin 4.
17. Connect a wire from the LCD's D7 to digital pin 5.
18. Connect a wire from the furthest end of the 220 Ohm resistor to power.
19. Connect a wire from the LCD's K to ground.
20. Connect a wire from a GND of the NeoPixel stick to ground on the breadboard.
21. Connect a wire from the DIN of the NeoPixel stick to digital pin 8.
22. Connect a wire from the 5VDC of the NeoPixel stick to power on the breadboard.
23. Connect a female-to-male dupont wire to connect the NRF24L01's GND to ground on the breadboard.
24. Connect a female-to-male dupont wire to connect the NRF24L01's CE to digital pin 9.
25. Connect a female-to-male dupont wire to connect the NRF24L01's SCK to digital pin 13.
26. Connect a female-to-male dupont wire to connect the NRF24L01's MISO to digital pin 12.
27. Connect a female-to-male dupont wire to connect the NRF24L01's VCC to the 3.3V pin

28. Connect a female-to-male dupont wire to connect the NRF24L01's CS to digital pin 20.
29. Connect a female-to-male dupont wire to connect the NRF24L01's MOSI to digital pin 11.

Player Arduino:

1. Gather the materials.
2. Insert the LCD display, potentiometer, joystick module, green LED, red LED, button 1, and button 2 on the breadboard such that each pin has its own row (shown as numbers).
3. Connect a wire from the 5V pin in the Arduino board to the positive side of the breadboard.
4. Connect a wire from the GND pin in the Arduino board to the negative side of the breadboard.
5. Connect a wire from LCD K to the negative side, or ground, of the breadboard.
6. Connect a wire from LCD A with to the positive side of the breadboard via 220 Ohm resistor.
7. Connect a wire from LCD D7 to digital pin 8 in the Arduino board.
8. Connect a wire from LCD D6 to digital pin 7 in the Arduino board.
9. Connect a wire front LCD D5 to digital pin 4 in the Arduino board.
10. Connect a wire front LCD D4 to digital pin 3 in the Arduino board.
11. Connect a wire from LCD E to digital pin 9 in the Arduino board.
12. Connect a wire from LCD RW to the negative side, or ground, of the breadboard.
13. Connect a wire from LCD RS to digital pin 10 in the Arduino board.
14. Connect a wire from LCD V0 with the middle leg (sig) of potentiometer.
15. Connect a wire from LCD VSS to the negative side, or ground, of the breadboard.
16. Connect a wire from LCD VDD to the positive side, or power, of the breadboard.
17. Connect a wire from an outer leg of the potentiometer with the positive side, or power, of the breadboard.
18. Connect a wire from the other outer leg of the potentiometer with the negative side, or ground, of the breadboard.
19. Connect a wire from a leg of the red LED to digital pin 6.
20. Connect a wire from the other leg of the red LED to the negative side, or ground, of the breadboard via a 330 Ohm resistor.
21. Connect a wire from a leg of the green LED to digital pin 5.
22. Connect a wire from the other leg of the green LED to the negative side, or ground, of the breadboard via a 330 Ohm resistor.
23. Connect a wire from one side of button 1 to analog pin A0.
24. Connect a wire from the other side of button 1 to the negative side, or ground, of the breadboard via a 10K Ohm resistor.
25. Connect a wire from one side of button 2 to analog pin A2.
26. Connect a wire from the other side of button 2 to the negative side, or ground, of the breadboard via a 10K Ohm resistor.
27. Connect a female-to-male dupont wire from the NRF24L01 Module GND to the negative side, or ground, of the breadboard.
28. Connect a female-to-male dupont wire from the NRF24L01 Module VCC to the 3.3V pin

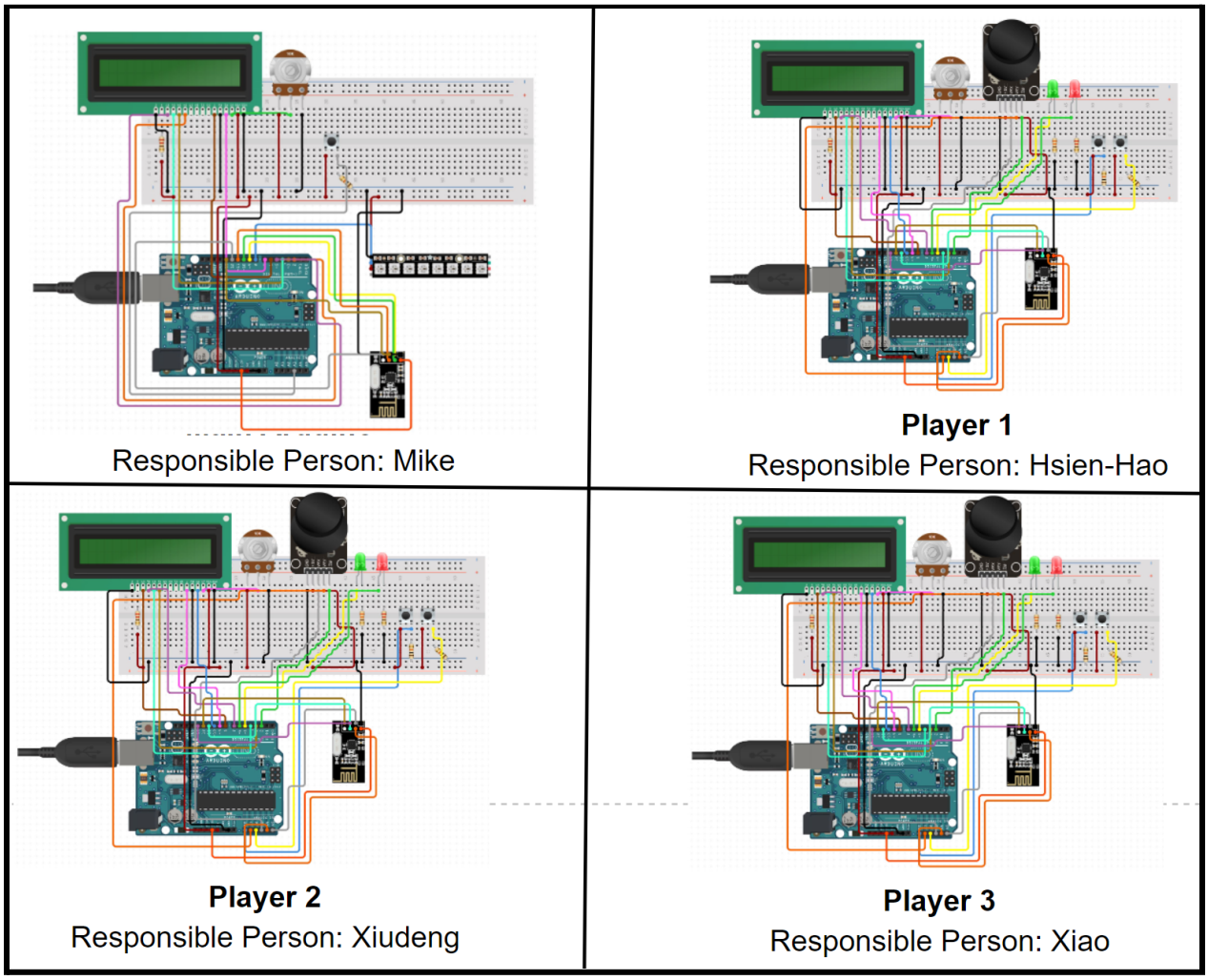
of the Arduino board.

29. Connect a female-to-male dupont wire from the NRF24L01 Module CE to the analog pin A5 of the Arduino board.
30. Connect a female-to-male dupont wire from the NRF24L01 Module CS to the analog pin A4 of the Arduino board.
31. Connect a female-to-male dupont wire from the NRF24L01 Module MOSI to the digital pin 11 of the Arduino board.
32. Connect a female-to-male dupont wire from the NRF24L01 Module MIOS to the digital pin 12 of the Arduino board.

How it is to be Used:

1. Turn on the master Arduino board and other players' Arduino.
2. The master Arduino will display a few messages.
3. Once the word is selected, the pixels in the NeoPixel stick will light up according to the code.
4. The players' Arduino will let you guess now. The '#' character is an empty character. Move the joystick up to go back a character (a goes to z) or down to go forward a character (a to b). Press the joystick to select the character. You will see it "move" down and append it to the string in the LCD's second row. You have 60 seconds to guess.
5. If the player wants to delete the guess, you can press button 1 (closest to the LCD).
6. If the player wants to submit their guess, you can press button 2 (furthest from the LCD).
7. The master Arduino receives the guess and checks it for correctness. If you are right, you will see a message displayed on the player's LCD display saying that it was correct and the score being one more point. Otherwise, the message will show you that it was wrong and received no point.
8. The green LED is lit if the guess was correct. Otherwise, the red LED is lit.
9. The master Arduino stops receiving guesses and selects a new word for the next round.
10. If a player reaches 10 points, the game stops to show if the player won or lost the game.
11. After a few seconds, the whole system resets for a new game.
12. Anyone can press the button on the master controller to reset the game.

Big Picture Diagram:



Responsible Person: Mike

Player 1

Responsible Person: Hsien-Hao

Player 2

Responsible Person: Xiudeng

Player 3

Responsible Person: Xiao

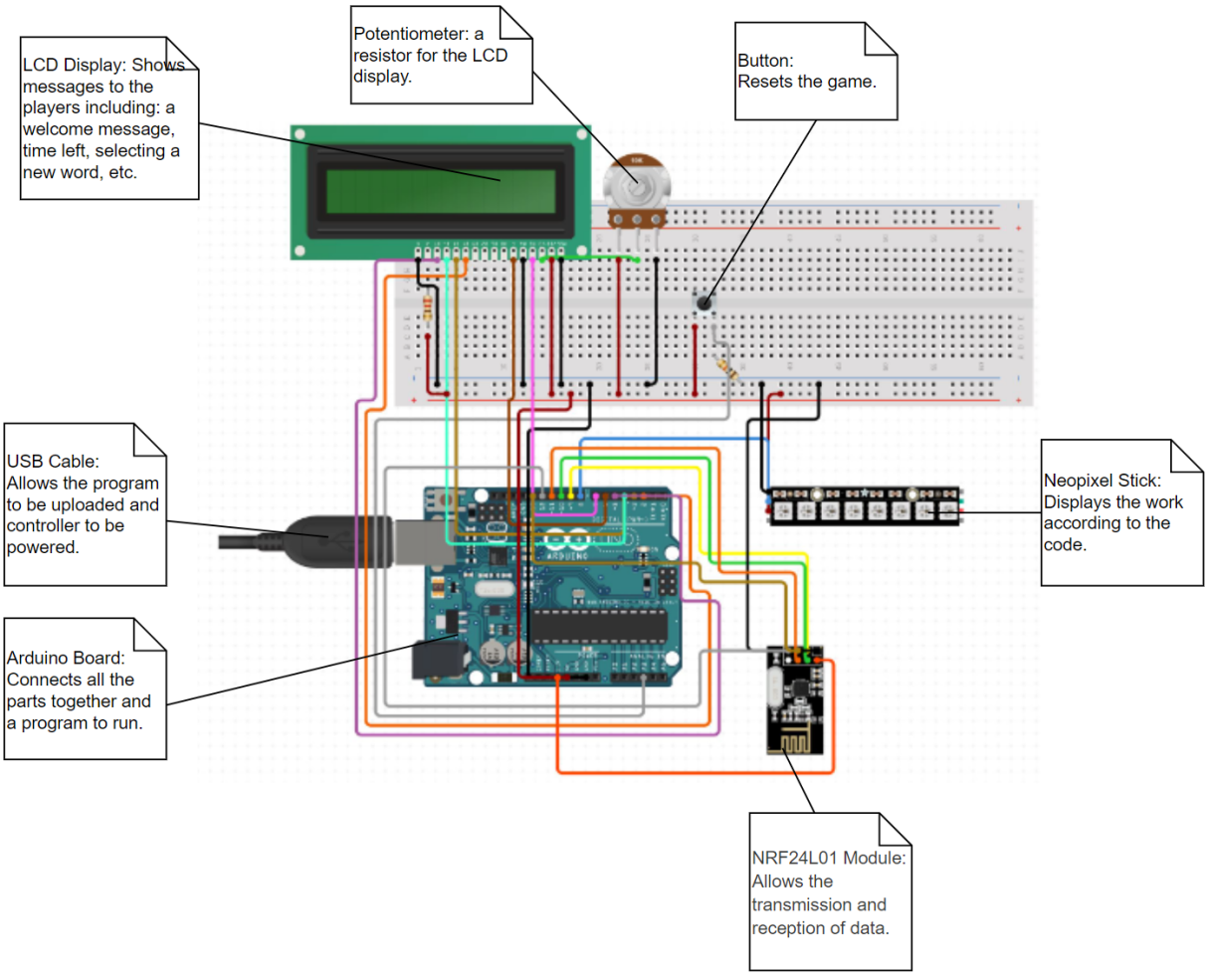
Hardware Diagrams:

Applies to all:

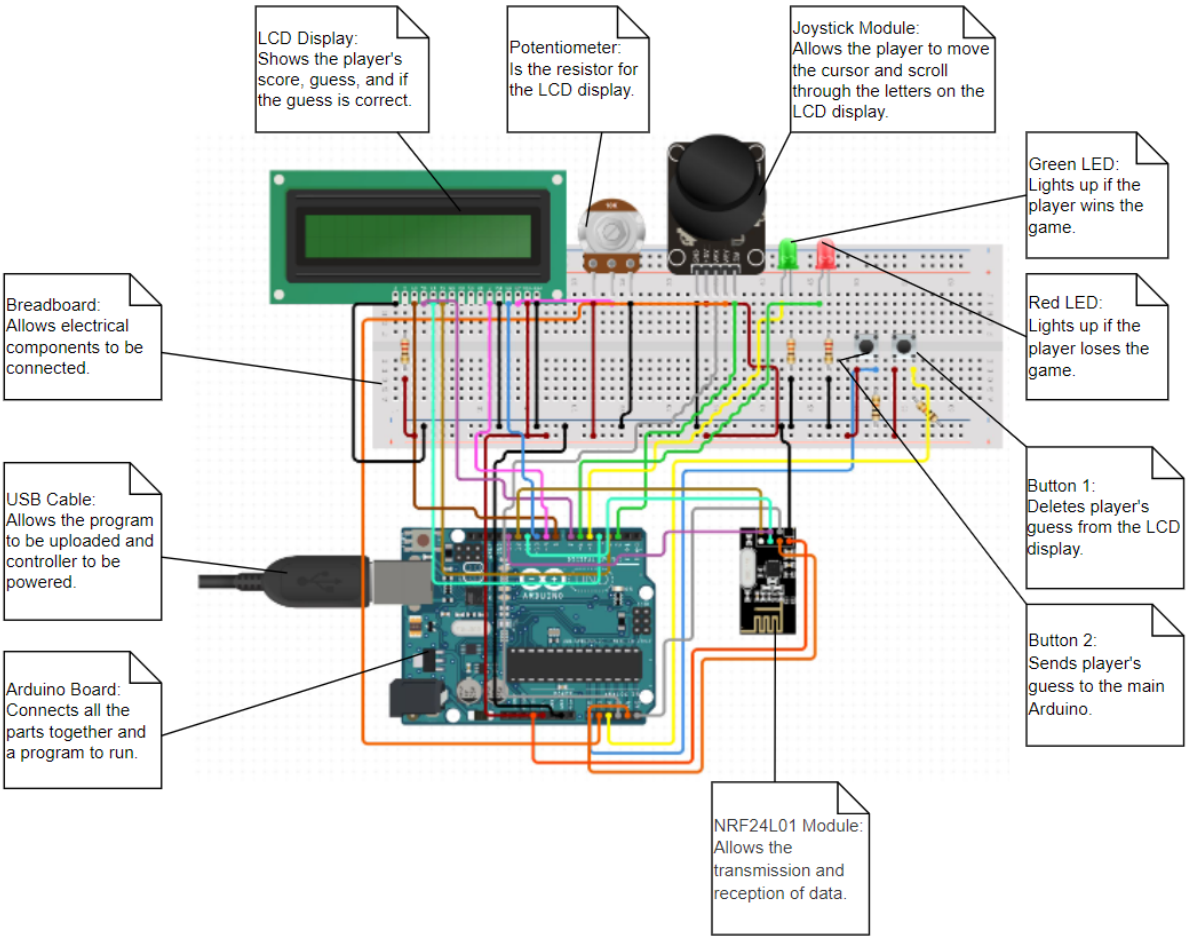
Wire (as the shape seen to the left):
Connects electrical components in series or parallel.

Resistor (as the shape seen to the right):
Limits the flow of electrical current.

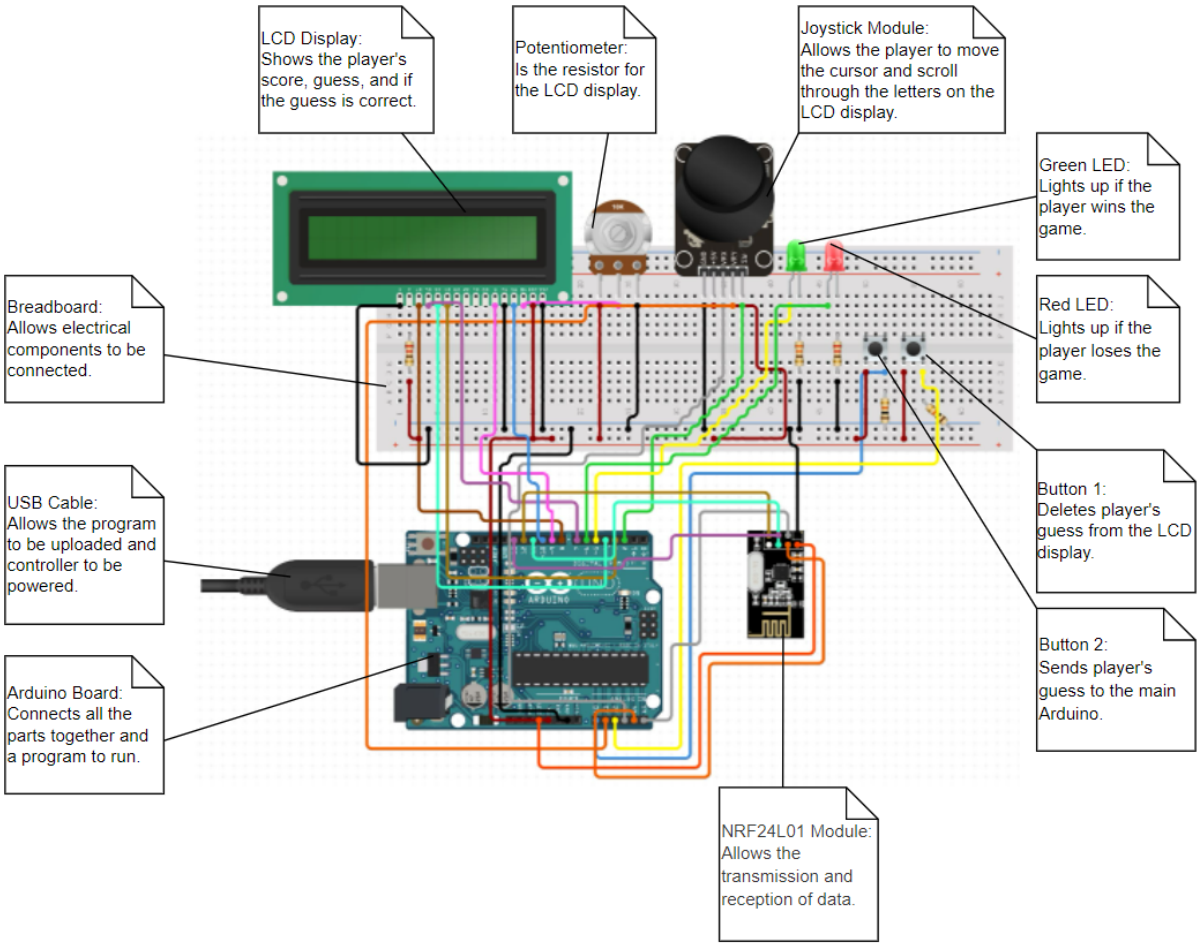
Main Arduino



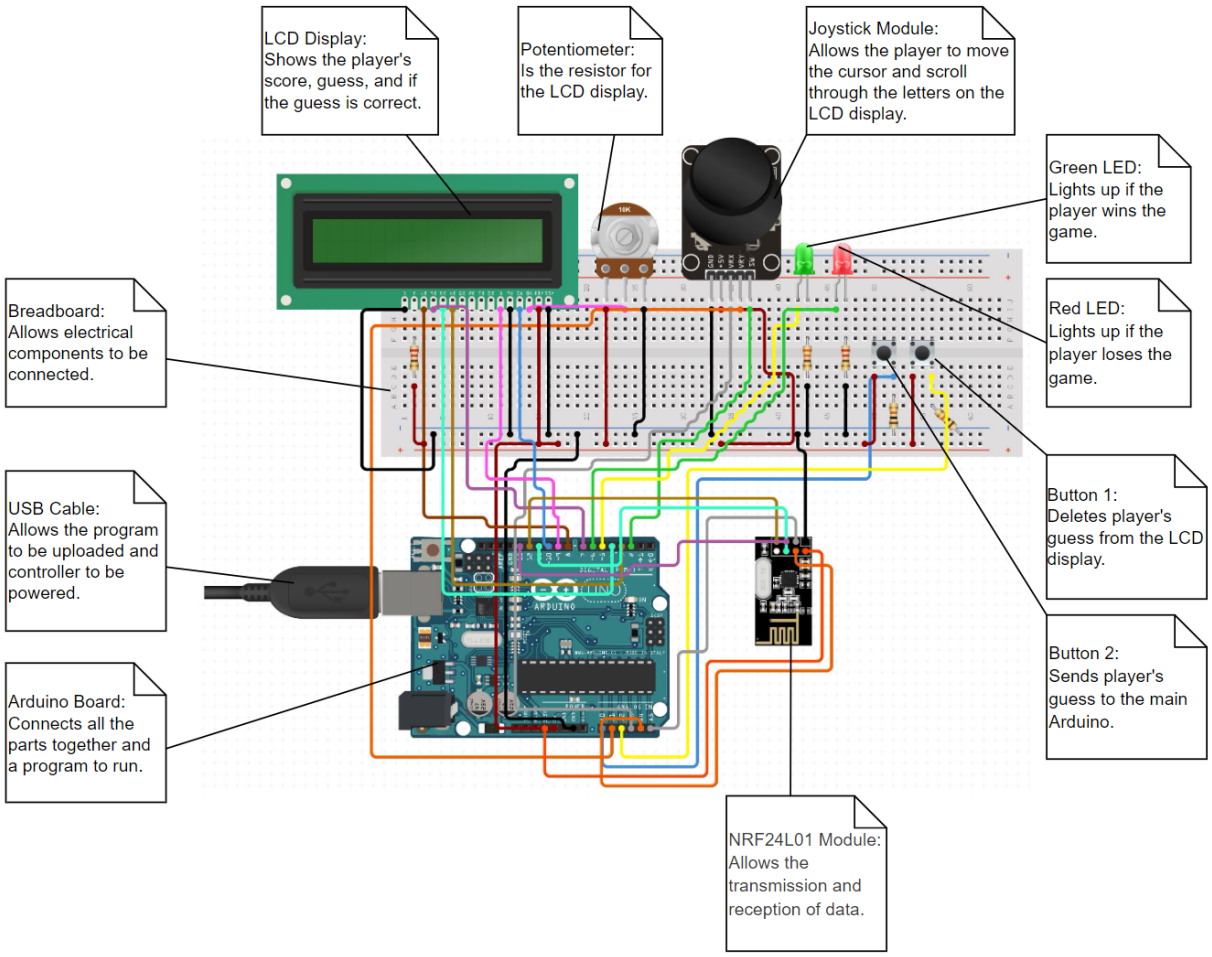
Player 1


LCD Display: Shows the player's score, guess, and if the guess is correct.
Potentiometer: Is the resistor for the LCD display.
Joystick Module: Allows the player to move the cursor and scroll through the letters on the LCD display.
Green LED: Lights up if the player wins the game.
Red LED: Lights up if the player loses the game.
Breadboard: Allows electrical components to be connected.
Button 1: Deletes player's guess from the LCD display.
USB Cable: Allows the program to be uploaded and controller to be powered.
Button 2: Sends player's guess to the main Arduino.
Arduino Board: Connects all the parts together and a program to run.
NRF24L01 Module: Allows the transmission and reception of data.

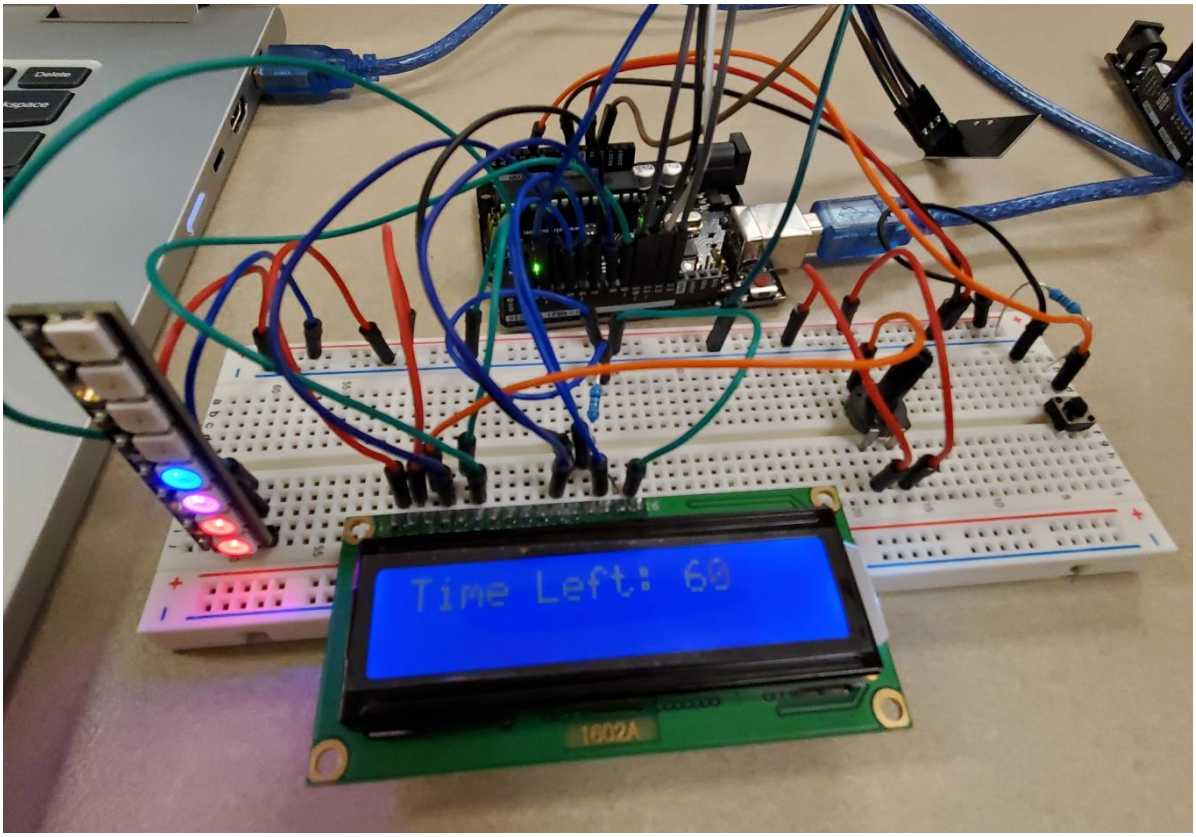
Player 2


LCD Display:
Shows the player's score, guess, and if the guess is correct.
Potentiometer:
Is the resistor for the LCD display.
Joystick Module:
Allows the player to move the cursor and scroll through the letters on the LCD display.
Green LED:
Lights up if the player wins the game.
Red LED:
Lights up if the player loses the game.
Breadboard:
Allows electrical components to be connected.
Button 1:
Deletes player's guess from the LCD display.
USB Cable:
Allows the program to be uploaded and controller to be powered.
Button 2:
Sends player's guess to the main Arduino.
Arduino Board:
Connects all the parts together and a program to run.
NRF24L01 Module:
Allows the transmission and reception of data.

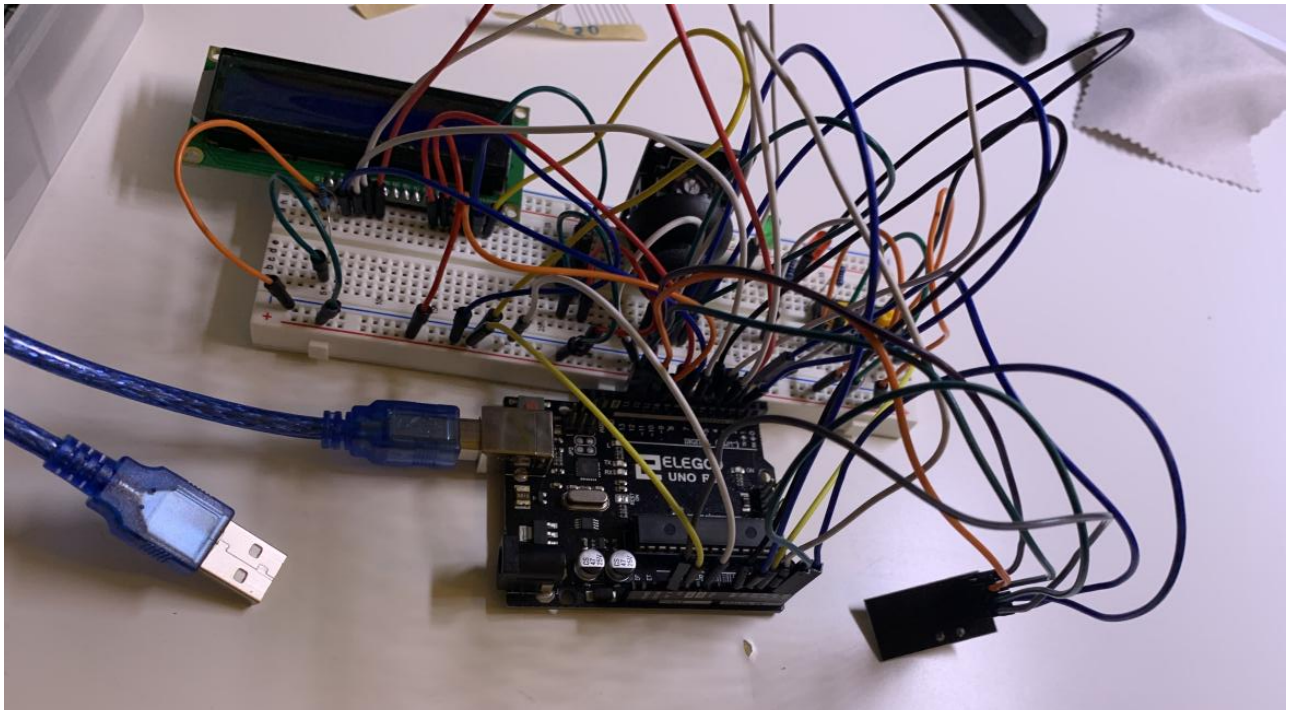
Player 3


LCD Display:
Shows the player's score, guess, and if the guess is correct.
Potentiometer:
Is the resistor for the LCD display.
Joystick Module:
Allows the player to move the cursor and scroll through the letters on the LCD display.
Green LED:
Lights up if the player wins the game.
Red LED:
Lights up if the player loses the game.
Breadboard:
Allows electrical components to be connected.
USB Cable:
Allows the program to be uploaded and controller to be powered.
Button 1:
Deletes player's guess from the LCD display.
Button 2:
Sends player's guess to the main Arduino.
Arduino Board:
Connects all the parts together and a program to run.
NRF24L01 Module:
Allows the transmission and reception of data.

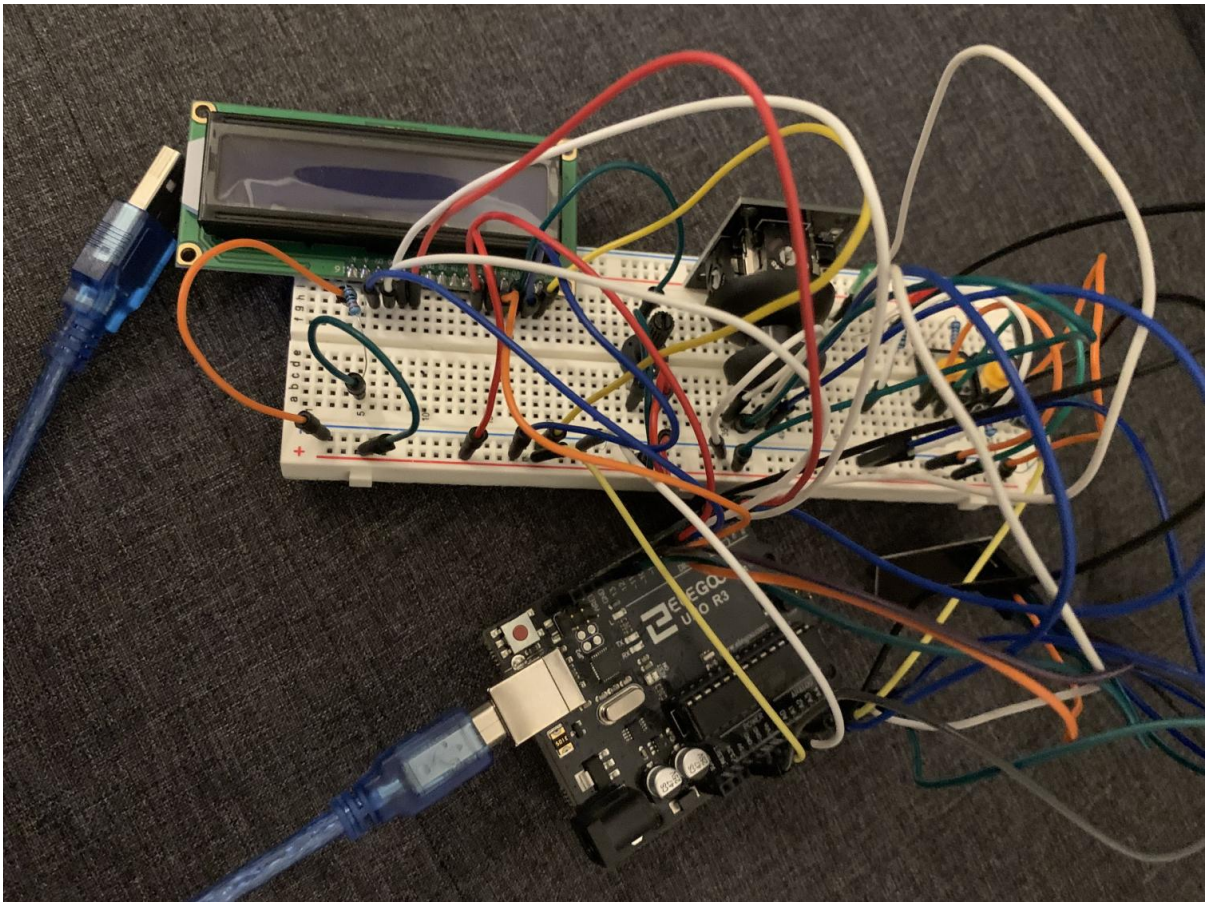
Project Pictures:



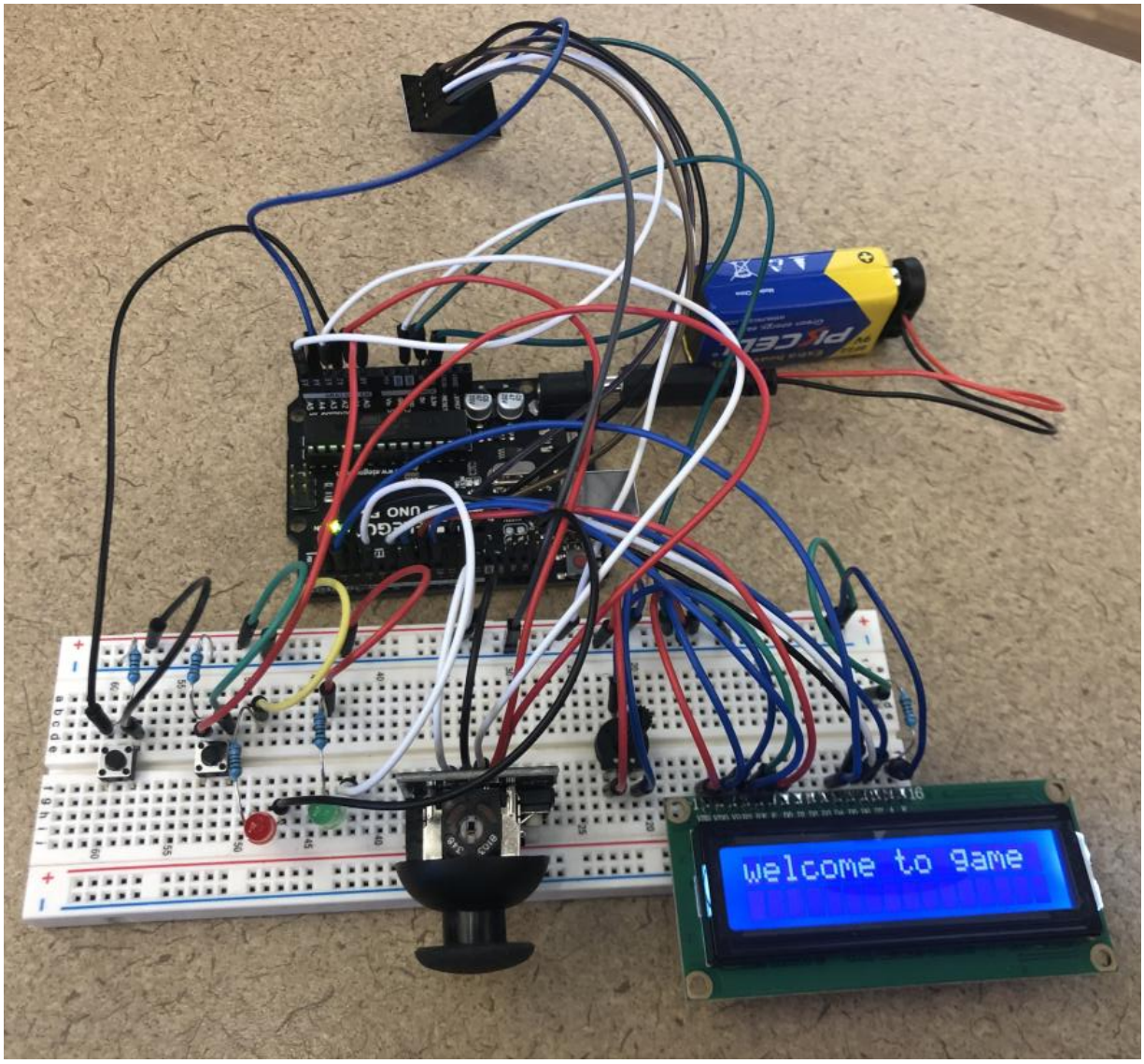
Main Arduino: Mike Apreza



Player 1: Hsien-Hao Chang



Player 2: Xiudeng Zhang



Player 3: Xiao Chen

Link to Project Functionality: https://www.youtube.com/watch?v=SVKoLArgF4Y

Source Code:

GameController.ino (for Master Arduino)

```
// Game Controller Arduino for CryptoColor Game
//
//
// References:
//    NeoPixel: https://elearn.ellak.gr/mod/book/view.php?id=4561&chapterid=2406 ,
//            https://github.com/adafruit/Adafruit_NeoPixel
//    NRF24L01:
https://create.arduino.cc/projecthub/muhammad-aqib/nrf24l01-interfacing-with-arduino-wireless-co
mmunication-0c13d4
//
https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/
//    Random: https://www.arduino.cc/reference/en/language/functions/random-numbers/random/
//    Reset: https://www.theengineeringprojects.com/2015/11/reset-arduino-programmatically.html
//
//
//
------------------------------------------------------------------------------------------------
-----------------------------

// Add the libraries required
#include <LiquidCrystal.h>
#include <Adafruit_NeoPixel.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

// Player will be sending data like this:
struct Data_Package {
  char guess[8] = {'H'};
  int ID = 0;
  int score = 0;
  int reset = 0;
  bool gameOver = false;
};

// Data variables for players
Data_Package data1;
Data_Package data2;
Data_Package data3;


// Button pin to restart a game
```

```
const int buttonPin = A1;
int buttonVal = 0;


// LED Strip pin
const int ledStick = 2;
const int numLED = 8;
// Start the LED strip
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(numLED, ledStick, NEO_GRB + NEO_KHZ800);


// Set up LED Display
const int rs = 4, en = 9, d4 = 8, d5 = 7, d6 = 6, d7 = 5;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


// NRF24L01 module pins
const int NRF24L01_CE = 3;
const int NRF24L01_CS = 10;
const int NRF24L01_MOSI = 11;
const int NRF24L01_MISO = 12;
const int NRF24L01_SCK = 13;

// Start the RF24 radio
RF24 radio(NRF24L01_CE, NRF24L01_CS);

// Address of this node in Octal format (04,031,etc.)
const byte thisAddr[6] = "00001"; // Controller
const byte player1[6] = "00001"; // Player 1
const byte player2[6] = "00003"; // Player 2
const byte player3[6] = "00004"; // Player 3

// This determines if the controller should listen for guesses
bool br = false;


// Used for reading value of button
unsigned long r;


// Time variables
const int MAXTIME = 80;
int currTime = 0;


// The array of words available in the game
const int sizeArrayWords = 16;
String currentWord = "";
```

```
const String words[] = {"dog", "cat", "grow", "bottle", "water", "gum", "glasses", "shirt", "photo", "coin",
"phone", "key", "pencil", "case", "notebook", "laptop"};
//const String words[] = {"gum", "cat"};
//const int sizeArrayWords = 2;


// Boolean variable to see if someone has won the game
bool isWin = false;

// This function is used to reset the arduino
void(* resetFunc) (void) = 0;


void setup() {

  // Being serial
  Serial.begin(9600);

  // LCD Display
  lcd.begin(16,2);

  // LED Strip
  pixels.begin(); // Initializes the library
  pixels.show(); // Sets the LEDs off

  // NRF24L01 module
  radio.begin();
  radio.openReadingPipe(0, thisAddr);
  radio.openWritingPipe(thisAddr);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();

  // For random number
  randomSeed(analogRead(0));

  // Print a welcoming message
  lcd.print("Welcome to");
  lcd.setCursor(0, 1);
  lcd.print("CryptoColor!");
  delay(2000);
  // Print that the game is going to being soon
  lcd.setCursor(0, 0);
  lcd.print("The game is");
  lcd.setCursor(0, 1);
  lcd.print("starting soon!");
  delay(4000);
  currTime = MAXTIME;
}
```

```
// This starts when all the players are ready.
void loop() {

  // Check to see if button is pressed
  isPressed();

  // 1. Select the word
  selectWord();

  // Check to see if button is pressed
  isPressed();

  // Start listening
  radio.startListening();

  // 4. Listen for guesses
  while(!br && currTime > 0) {

    // 3. Start the timer (80 seconds)
    if (currTime == 1) {
      lcd.clear();
      lcd.print("Time Left: " + String(currTime));
      currTime--;
      br = true;
    } else {
      lcd.clear();
      lcd.print("Time Left: " + String(currTime));
      currTime--;
    }

    // Read the incoming data from all the Players
    radio.read(&data1, sizeof(data1));
    if (data1.ID == 1) {
      Serial.println("Data recieved: ");
      Serial.println(data1.guess);
      Serial.println(data1.ID);
      Serial.println(data1.score);
    }
//    radio.read(&data2, sizeof(data2));
//    radio.read(&data3, sizeof(data3));
//
    // 5. If guess is recieved, check to see if it's correct
    if(isCorrect(data1.guess)) {
      // 6. If it is, add points to the player's score
      sendPoint(data1.ID, 1);
    } else {
```

```
      // 7. If not, send a value to display a message
      sendPoint(data1.ID, 0);
    }
//
//    if(isCorrect(data2.guess)) {
//      sendPoint(data2.ID, 1);
//    } else {
//      sendPoint(data2.ID, 0);
//    }
//
//    if(isCorrect(data3.guess)) {
//      sendPoint(data3.ID, 1);
//    } else {
//      sendPoint(data3.ID, 0);
//    }

    // "Make a sec pass"
    delay(1000);
    if(br == true) {
      lcd.clear();
      lcd.print("Time Left: 0");
      lcd.setCursor(0, 1);
      lcd.print(currentWord);
    }

  } // End of while


  radio.stopListening();
  // Check to see if button is pressed
  isPressed();

  // 8. Do not recieve guesses and see if someone won
  if(isWin) {
    isWin = false;
    // reset the game
    delay(5000);
    resetFunc();
  } else {
    // 9. If time is up, display the answer to the players and turn off the LED lights
//    radio.write(&currentWord, sizeof(currentWord));
//    radio.write(&currentWord, sizeof(currentWord));
//    radio.write(&currentWord, sizeof(currentWord));
    pixels.clear();
    pixels.show();
  }

  // Check to see if button is pressed
```

```
  isPressed();

} // END



// This function is used to select a word and display it on the LED strip
void selectWord() {

  pixels.clear();

  lcd.clear();
  lcd.print("Selecting a");
  lcd.setCursor(0, 1);
  lcd.print("random word...");

  delay(1500);

  // 1. Select a random word
  long randomNum = random(0, sizeArrayWords); // From 0 to sizeArrayWords-1
  currentWord = words[randomNum]; // Save it to the currentWord variable

  // 2. Parse through the word and output the correct colors in the LED lights
  char W[8];
  currentWord.toCharArray(W, 8);
  // pixels.Color takes RGB values and pixels go from 0-7

  Serial.println("Word: " + currentWord);

  for(int i = 0; i < sizeof(currentWord); ++i) {

    if(W[i] > 96 && W[i] < 100) {
      pixels.setPixelColor(i, pixels.Color(255, 0, 0)); // RED - ABC
    } else if (W[i] > 99 && W[i] < 103) {
      pixels.setPixelColor(i, pixels.Color(0, 0, 255)); // BLUE - DEF
    } else if (W[i] > 102 && W[i] < 106) {
      pixels.setPixelColor(i, pixels.Color(0, 255, 0)); // GREEN - GHI
    } else if (W[i] > 105 && W[i] < 110) {
      pixels.setPixelColor(i, pixels.Color(255, 255, 255)); // WHITE - JKLM
    } else if (W[i] > 109 && W[i] < 113) {
      pixels.setPixelColor(i, pixels.Color(255, 255, 0)); // YELLOW - NOP
    } else if (W[i] > 112 && W[i] < 117) {
      pixels.setPixelColor(i, pixels.Color(128, 0, 128)); // PURPLE - QRST
    } else if (W[i] > 116 && W[i] < 123) {
      pixels.setPixelColor(i, pixels.Color(255, 165, 0)); // ORANGE - UVWXYZ
    } else {
      // Then it is not valid. Do nothing.
```

```
    }
  }
  pixels.setBrightness(5);
  pixels.show(); // Send the updated pixel color to hardware

  currentWord.toCharArray(data1.guess, sizeof(data1.guess));
  data1.ID = 1;
  data1.score = 0;
  data1.reset = -1;
  data1.gameOver = false;

  radio.write(&data1, sizeof(data1));
  radio.write(&data1, sizeof(data1));
  radio.write(&data1, sizeof(data1));

}


// This function is used to see if the player's guess is correct
bool isCorrect(String guess) {
  if (guess == currentWord) {
    Serial.println("CORRECT");
    return true;
  } else {
    return false;
  }
}


// This function is used to send the point of 1 or 0
void sendPoint(int ID, int point) {

  if (ID == 1) {
    String r = "STRING";
    r.toCharArray(data1.guess, sizeof(data1.guess));
    data1.ID = 1;
    int oldScore = data1.score;
    data1.score = oldScore + point;
    Serial.println("New Score: ");
    Serial.println(data1.score);
    data1.reset = 0;
    data1.gameOver = false;
    if(isThereWin(ID, data1.score)) {
      // Send data to everyone
      data1.gameOver = true;
      radio.stopListening();
      radio.write(&data1, sizeof(data1));
      //radio.write(&data2, sizeof(data2));
```

```
      //radio.write(&data3, sizeof(data3));
    } else {
      radio.stopListening();
      radio.write(&data1, sizeof(data1));
      radio.write(&data1, sizeof(data1));
      radio.write(&data1, sizeof(data1));
    }
//  } else if (ID == 2) {
//    data2.score += point;
//    if(isThereWin(ID, data2.score)) {
//      // Send data to everyone
//      radio.write(&data1, sizeof(data1));
//      radio.write(&data2, sizeof(data2));
//      radio.write(&data3, sizeof(data3));
//    } else {
//      radio.write(&data2, sizeof(data2));
//    }
//  } else if (ID == 3) {
//    data3.score += point;
//    if(isThereWin(ID, data3.score)) {
//      // Send data to everyone
//      radio.write(&data1, sizeof(data1));
//      radio.write(&data2, sizeof(data2));
//      radio.write(&data3, sizeof(data3));
//    } else {
//      radio.write(&data3, sizeof(data3));
//    }
  } else {
    // It is not a valid ID
  }

  radio.startListening();

}


// This function checks if the winning condition is met
bool isThereWin(int ID, int points) {
  if (points == 10) {
    // Update the bool variable in Data_Package
    data1.gameOver = true;
    data2.gameOver = true;
    data3.gameOver = true;
    // Update variable for controller
    isWin = true;
    return true;
  }
  return false;
```

```
}


// This function is used to see if the button is pressed
void isPressed() {

  int isPress = analogRead(buttonPin);
  if(isPress > 1000) { // It's pressed
    // Send a number to the players to reset their arduino
    String r = "press";
    r.toCharArray(data1.guess,sizeof(data1.guess));
    data1.ID = 1;
    data1.score = 0;
    data1.reset = -1;
    data1.gameOver = false;

//    r.toCharArray(data2.guess,sizeof(data2.guess));
//    data2.ID = 2;
//    data2.score = 0;
//    data2.reset = -1;
//    data2.gameOver = false;
//
//    r.toCharArray(data3.guess,sizeof(data3.guess));
//    data3.ID = 3;
//    data3.score = 0;
//    data3.reset = -1;
//    data3.gameOver = false;

    radio.write(&data1, sizeof(data1));
    radio.write(&data1, sizeof(data1));
    radio.write(&data1, sizeof(data1));
    //radio.write(&data2, sizeof(data2));
    //radio.write(&data3, sizeof(data3));
    // Reset this Arduino
    lcd.clear();
    lcd.print("Restarting...");
    delay(1000);
    resetFunc();
  }

}
```

Player.ino (for Player Arduinos)

```
// Player Arduino for CryptoColor Game
#include <LiquidCrystal.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24Network.h>
// Button pin to submit player's guess
const int guessButtonPin = A2;
// Button to delete player's guess
const int deleteButtonPin = A0;
int stateOpen = 0;

// Red and Green LED lights
const int redPin = 6;
const int greenPin = 5;
int letterCounter=0;
int cursorCounter=0;
int oldScore = 0;
int newScore = 0;
bool gameOver =true;

// Address of this node in Octal format ( 04,031, etc)
const uint16_t this_node = 02;
const uint16_t controller = 00;

// The LCD's layout is seen as the following in terms of (column, row):
//       (0, 0) . . . (15, 0)
//       (0, 1) . . . (15, 1)
const int rs = 10, en = 9, d4 = 3, d5 = 4, d6 = 7, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); // Set up the LCD display
char chararr[]={'#','a','b','c','d','e','f','g','h','i','j','k','l', 'm','n','o','p','q','r','s','t','u','v','w','x','y','z' };
String answer ="";
char realAnswer[8] ={};
// This is the struct that will be sent to the main arduino

struct Data_Package {
  char guess[8] ={};
  int ID = 3;
  int score = 0;
  int reset = 0;
  bool gameOver =false;

 };

 //NRF24L01 module pins
const int NRF24L01_CE = A5;
```

```
const int NRF24L01_CS = A4;
const int NRF24L01_MOSI = 11;
const int NRF24L01_MISO = 12;
const int NRF24L01_SCK = 13;

const byte address[6] = "00001";

 // Start the RF24
RF24 radio(NRF24L01_CE, NRF24L01_CS);
// Include radio in the network
RF24Network network(radio);
RF24NetworkHeader header0(controller);


 Data_Package dataPackage;

// Joystick to let the player select their letters for their guess

// this one may look wrong in device, but it works
int joyX = A1;
int joyY = A3;
int joySwitch = 2;


int xPosition = 0;
int yPosition = 0;
int switchState = 0;
int mapX = 0;
int mapY = 0;
//String realAnswer="";
// The NRF24L01 - 2.4G Wireless Transceiver Module



void setup() {
  // Buttons are input
  pinMode(guessButtonPin, INPUT);
  pinMode(deleteButtonPin, INPUT);

  // LED lights are output
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);


  // Set up the LCD's number of columns and rows
  lcd.begin(16, 2);
  lcd.print("welcome to game");
  delay(2000);
```

```
  // Set up the joystick
  Serial.begin(9600);

//
  pinMode(joyX, INPUT);
  pinMode(joyY, INPUT);
  pinMode(joySwitch, INPUT_PULLUP);

  // Connect to the Master arduino
  radio.begin();
  radio.openWritingPipe(address);
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();


}
void resultGame(){
  if(newScore == 1 ){
    digitalWrite(greenPin,HIGH);
    lcd.clear();
    lcd.print("You Won !");
    delay(5000);
    }
  else{

     digitalWrite(redPin,HIGH);
     lcd.clear();
     lcd.print("You lost !");
     delay(5000);


    }
  }
void getScore(){
   if (radio.available()) {
  radio.startListening();
  radio.read(&dataPackage,sizeof(dataPackage));

   }
//  newScore = dataPackage.score;
  delay(500);
  Serial.println("string: ");
  Serial.println(dataPackage.guess);

  Serial.println(dataPackage.score);
  Serial.println("testing string: ");
```

```
  Serial.println(dataPackage.guess);
  char temp[8]={};
  answer.toCharArray(temp,sizeof(temp));
  Serial.println("temp");
  Serial.println(temp);
//  Serial.println(String(dataPackage.guess) == String(temp));



  if(String(realAnswer) == String(temp) ){
    lcd.print("here");
    newScore= newScore+1;
   Serial.println("newScore");
    Serial.println(newScore);
    }

    resultGame();



//  if(newScore - oldScore == 1){
//    lcd.clear();
//    lcd.print("Good guess");
//    // This one contains the score
//    lcd.setCursor(5,1);
//    lcd.print("score: ");
//    lcd.print(newScore);
// // result =true;
//  } else if (newScore == oldScore) {
//    lcd.clear();
//    lcd.print("Wrong guess");
//    // This one contains the score
//    lcd.setCursor(5,1);
//    lcd.print("score: ");
//    lcd.print(newScore);


//  }
// oldScore = newScore;
}



void sendData(){
   Serial.println("this is a testing");
   radio.stopListening();
   answer.toCharArray(dataPackage.guess,sizeof(dataPackage.guess));
   radio.write(&dataPackage,sizeof(dataPackage));
```

```
   getScore();
}


void(* resetFunc) (void) = 0;

void loop() {

 // receive
  //  restart the game
   if (radio.available()) {
   radio.read(&dataPackage, sizeof(dataPackage));

      if(dataPackage.reset == 1){
        resetFunc();
        lcd.clear();
        lcd.print("Retset...");
        delay(1000);

   } //reset

    Serial.println("data: ");
    Serial.println(dataPackage.guess);
    Serial.println(dataPackage.ID);
    Serial.println(dataPackage.score);
    Serial.println(dataPackage.gameOver);
    Serial.println(dataPackage.reset);

    radio.stopListening();
    memcpy(realAnswer, dataPackage.guess, sizeof(dataPackage.guess));
    newScore=dataPackage.score;
    Serial.println("new");
    Serial.println(newScore);



//    if(newScore - oldScore == 1){
//      lcd.clear();
//      lcd.print("Good guess");
//      // This one contains the score
//      lcd.setCursor(5,1);
//      lcd.print("score: ");
//      lcd.print(newScore);
//      delay(1000);
//      lcd.clear();
//      lcd.print("Next round..");
//      oldScore = newScore;
//      //resultGame();
```

```
//     }


 while(dataPackage.reset==-1 && dataPackage.gameOver == 0){

  if(analogRead(deleteButtonPin)>900){
    lcd.clear();
    answer="";
  }


  delay(120);

  xPosition = analogRead(joyX);
  yPosition = analogRead(joyY);
  switchState = digitalRead(joySwitch);
  mapX = map(xPosition, 0, 1023, -512, 512);
  mapY = map(yPosition, 0, 1023, -512, 512);

  lcd.clear();

  if(mapY > 400){
    //lcd.print("show");
    letterCounter++;

    if(letterCounter>26)
      letterCounter = 0;
  }
  if(mapY < -400){
    //lcd.print("show");
    letterCounter--;
    if(letterCounter<0)
      letterCounter = 26;
  }
//  if(mapX > 400){
//    //lcd.print("show");
//    cursorCounter++;
//    if(cursorCounter >16)
//      cursorCounter=0;
//    //Serial.println(cursorCounter);
//
//  }
//   if(mapX < -400){
//    //lcd.print("show");
//    cursorCounter--;
//    if(cursorCounter <0)
//      cursorCounter=16;
```

```
//    //Serial.println(cursorCounter);
//    }

  lcd.setCursor(cursorCounter,0);
  lcd.print(chararr[letterCounter]);
  if(switchState == 0){
    answer = answer+chararr[letterCounter];
  }
  lcd.setCursor(0,1);
  lcd.print(answer);

  // Serial.println(analogRead(guessButtonPin));
  if(analogRead(guessButtonPin)>900){
      sendData();
  }

}

  } // radio

}
```

Timeline of Development:

Week of	Tasks	Completed	Milestone Deadlines
2/21	Work on the initial design document.	Initial design document.	M3 - 2/26
2/28	Buy materials. Register for the EXPO.	Material at hand. EXPO registration.	
3/7	Figure out the game logic and start to code.	Basic game logic in comment form. Basic I/O pin and type connections in code.	
3/14	Figure out the communication technicalities.	Communication technicalities in code.	
3/21	Start milestone 4 and work on the updated document. Set up the I/O devices for use in source code.	Half-way done with M4. Completed setup() in source code. Identified functions needed for code.	
3/28	Finish the M4 document. Code the functions.	Finished the updated design document Finished the functions (not the loop function) in the source code.	M4 - 3/29 (Tue)
4/4	Create the presentation slides and record the video. Do the loop() for source code.	Presentation slides. Video presentation for EXPO. Finished the loop() for both files.	M5 Part 1 (slides) - 4/8 M5 Part 2 (Video) - 4/8
4/11	Debug code.		
4/18	Debug. Go to in-person demo to do M6 and M7.	Debugged code for all subsystems. Peer reviews. In-person demo (M6).	M6 - 4/21 M7 - 4/21
4/25	Do the final design document and team assessment.	Final design document and team assessment.	M8 - 4/27 M9 - 4/28

References

Banovic, Sasa, et al. “Arduino Wireless Network with Multiple NRF24L01 Modules.” How To Mechatronics, 31 July 2018, https://howtomechatronics.com/tutorials/arduino/how-to-build-an-arduino-wireless-network-with-multiple-nrf24l01-modules/.

Circuito.io, Roboplan.io, https://www.circuito.io/app?components=512%2C11021.

Circuito.io, Roboplan.io, https://www.circuito.io/app?components=97,512,11021,216577,341099,748665

Circuito.io, Roboplan.io, https://www.circuito.io/app?components=97,97,512,9590,9591,11021,341099,611984,748665

Code Stack!: Rules Summary. Amigo.games, 27 Mar. 2019, https://www.amigo.games/content/ap/rule/19008-tack%20Rules%20Summary%20030419.pdf.

“Game Profile: Code Stack!” Amigo.games, 27 Mar. 2019, https://www.amigo.games/game/codestack#1527495873835-fdb97cc7-b7b8.

“Arduino LED Guess Word Game” Viiicky0309,

https://www.instructables.com/Arduino-LED-Guess-Word-Game/

“A Simple Guessing Game” Thesharanmohan, 24 Nov. 2019,

https://thesharanmohanblog.wordpress.com/2019/11/24/a-simple-guessing-game/