University of Washington
EE 299 Lab 4
Third Steps to Design

Mitchell Szeto
Bert Zhao
Feifan Qiao

TABLE OF CONTENTS

ABSTRACT

This is the lab report for the final project of EE 299. This report provides an overview of the project, along with a design specification and specific breakdown of the hardware and software implementation of the system. An analysis of the project is provided at the end of the report along with program schematics. Key pieces of code are included in the appendix.

INTRODUCTION

In this final lab, we will practice the formal design process that we have learned throughout this quarter and design a complex system. We are tasked to create our own project and work through the entire design process with this idea. The only requirements for this final lab are that there are two arduinos in the architecture that communicate through XBee communication, serial monitor and the LCD screen are utilized, and we need to use at least three features of Arduinos that we haven't used before (one of them has to be the tilt or temperature sensor).

DISCUSSION OF LAB

Design Specification

The project that we're worked on is an interactive Wheel of Fortune game that is supported by two Arduinos. In this specific implementation of Wheel of Fortune, two players are able to play against each other. The game has two main aspects: a hangman like minigame and the spinning of the wheel. At the beginning of each mini game, a third person inputs the hangman phrase through a button input on a LCD screen. The two players then take turns playing the hangman game. And at the beginning of each player turn, the player spins the wheel to determine a dollar value and guess a letter in the hangman. The player guesses the letter in on the input LCD and the result is then displayed on the output LCD. Calling a correct letter earns the value on the wheel, multiplied by the number of times that the letter appears in the hangman. At the end of the game, the player with the most money wins.

On a more specific level, this game will be controlled through two Arduinos. One Arduino will handle all of the peripheral control and serial monitor. The peripherals that are use are the tilt, button, and rotational sensor. These three sensors will be used to spin the wheel. The tilt sensor is shaked by the player to emulate spinning the wheel. The rotational sensor is used to gauge the spin power. And the button sensor is used to signal that the user is done spinning the wheel.
This Arduino will also handle user input through serial monitor and run the hangman game.

The other Arduino will be connected to the LCD screen and handle the user display that is not part of serial monitor. This will show the wheel spinning, the hangman game, and player information that includes who the player is and how much money they currently have won.

Hardware Implementation

The hardware implementation can be splitted into two distinct parts: the peripherals on the master Arduino and the peripherals on the slave Arduino.

The master device is connected to a wireless sd shield, a lcd display, a rotary potentiometer, and a pushbutton. Onboard the wireless shield there is the XBee wireless communication chip. This chip enables wireless communication between multiple arduinos. Inputting information to the master device is achieved through the potentiometer. When the user rotates the potentiometer, the cursor scrolls through the alphabet. The pushbutton is used to confirm character choices. A single click of the pushbutton selects the character while double click of the pushbutton finishes the input.

Similar to the master devices, the slave device also has a wireless sd shield and a lcd display. Other than those peripheral devices, the slave arduino is connected to a buzzer. Wireless sd shield is equipped with a Xbee chip and is used to communicate with the master device. Lcd display on the slave device is used for showing the spinning of wheel to the wheel as well as showing the score earned by each player. Buzzer on the slave device will sound when the wheel finishes spinning.

Software Implementation

The software implementation can be splitted into two main parts: the software design on the master and slave Arduino. We designed our software so most of the game is ran on the master Arduino. This includes all the user input and game functionality. On the other hand, the slave handles the user display so the user knows the game statistics and the game state.

The master arduino handles many of the key game functions. The most notable functions are the game state system and player state system. The game state is split into 5 key states. State 0 is used when setting up the game, where the host can enter in the phrase for the player to guess. Once the phrase has been entered, the game transitions to state 1. Game state 1 means that it is currently player 1's turn. The player will have the option to spin the wheel and make a guess at a letter that might be in the word. Once player 1's turn is over, the game transitions to state 2, which is player 2's turn. The game cycles between state 1 and 2 until all letters have been guess, at which point it moves to game state 4. State 4 is the game over state. It informs the players that the game is over and that displays the final scores of each player. The fifth and final state is a hold state and it becomes the active state once the program is finished displaying the game over information.

For the players, each turn is also split into multiple states. There are 6 states within each player's turn. Player state 0 is the beginning of the player's turn where they determine their spin strength by shaking the tilt sensor. Once the spin strength has been determined, the player moves on to state 1, where they wait for the wheel to spin and land on a value. Once the value has been determined, the player continues to state 2. State 2 is where the player inputs a letter to guess. Once they submit their guess, the program progresses to player state 5, which clears the serial buffer. Then the game automatically progresses to state 3, which checks the guess against the phrase, and updates the score. Then finally in state 4, the game

determines whether the game is over, or if its the next players turn. When starting the next player's turn, the player state is reset to 0.

The slave device is used to implement the wheel spin. When designing the software for the wheel, we wanted it to mimic the real wheel as close as possible, so we introduced several variables that inputted through sensors. Strength of the spin is inputted through the tilt sensor; friction of the wheel can also be modified by values from the potentiometer. The main loop itself is structured into different game states. Game state 1 reads the input transmitted from the master; game state 2 processes the inputs and spins the wheel. Spinning the wheel is achieved by repeatedly scrolling through a string that stores all the wedges of the wheel. Depending on the strength and friction of the wheel, the string variable that stores the wedges is displayed multiple times. Using substring and compare functions in string class, the loop compares the part of the string variable that matches the values transmitted from the master and puts that part of the string variable on the very front of the lcd display.

ANALYSIS OF ANY ERRORS

One of the main problems was that the XBee modules worked through serial communication. So the XBee chips used the Arduino Serial commands to communicate with each other. This was a problem because our original design consisted of user input through the serial monitor. This clearly did not work when we needed to use serial for the XBee communication. So, our solution was to handle user input through a LCD screen and have button toggles so the user can select letters and view them through a LCD screen.

When XBee is switched to "mirco" mode, it sends and receives information through the hardware serial. Initially, we thought only Serial.write() would send the information over the air while Serial.println() only prints to the serial monitor without sending the data over the air. However, when we started testing sending and receiving information with XBee, the slave XBee would often receive random information between intended information. Having garbage information in between wanted information poses a serious challenge for us. Eventually, we found out that Serial.println() not only prints to the serial monitor but also send that information over to the slave device, which means that if we wanted to receive proper information, we would have to use only Serial.write() and not us Serial.println() to avoid interference between useful and garbage data.

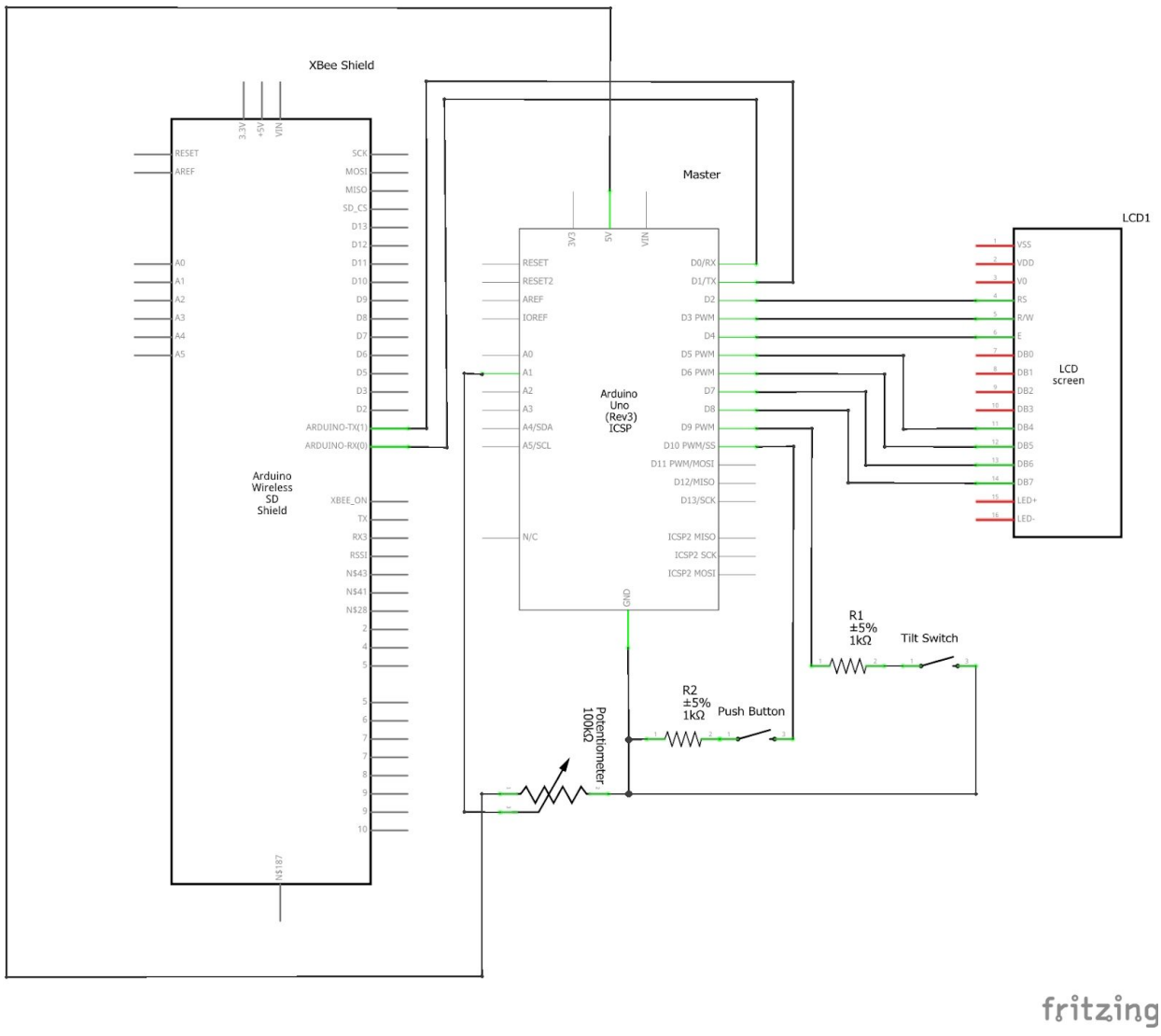
Another source of error that arose during our testing the reading the state of the button. On paper, reading the state of the button might sounds like a trivial task. All that is required for read the state of the button is using digitalRead(). However, when reading consecutive button presses, button states received is often random and is seemed that the function cannot distinguish the difference between one and two button presses. The explanation for this is that when the motion of pressing and releasing the button is a relatively long process when compared to the speed the microcontroller processes the code. Oftentimes, the microcontroller finishes processing the code long before the motion is finished. In order to slow down the processing of the code, we had to add delay() function between two digitalRead() so that the motion can be completed before the code is processed.

SUMMARY AND CONCLUSION

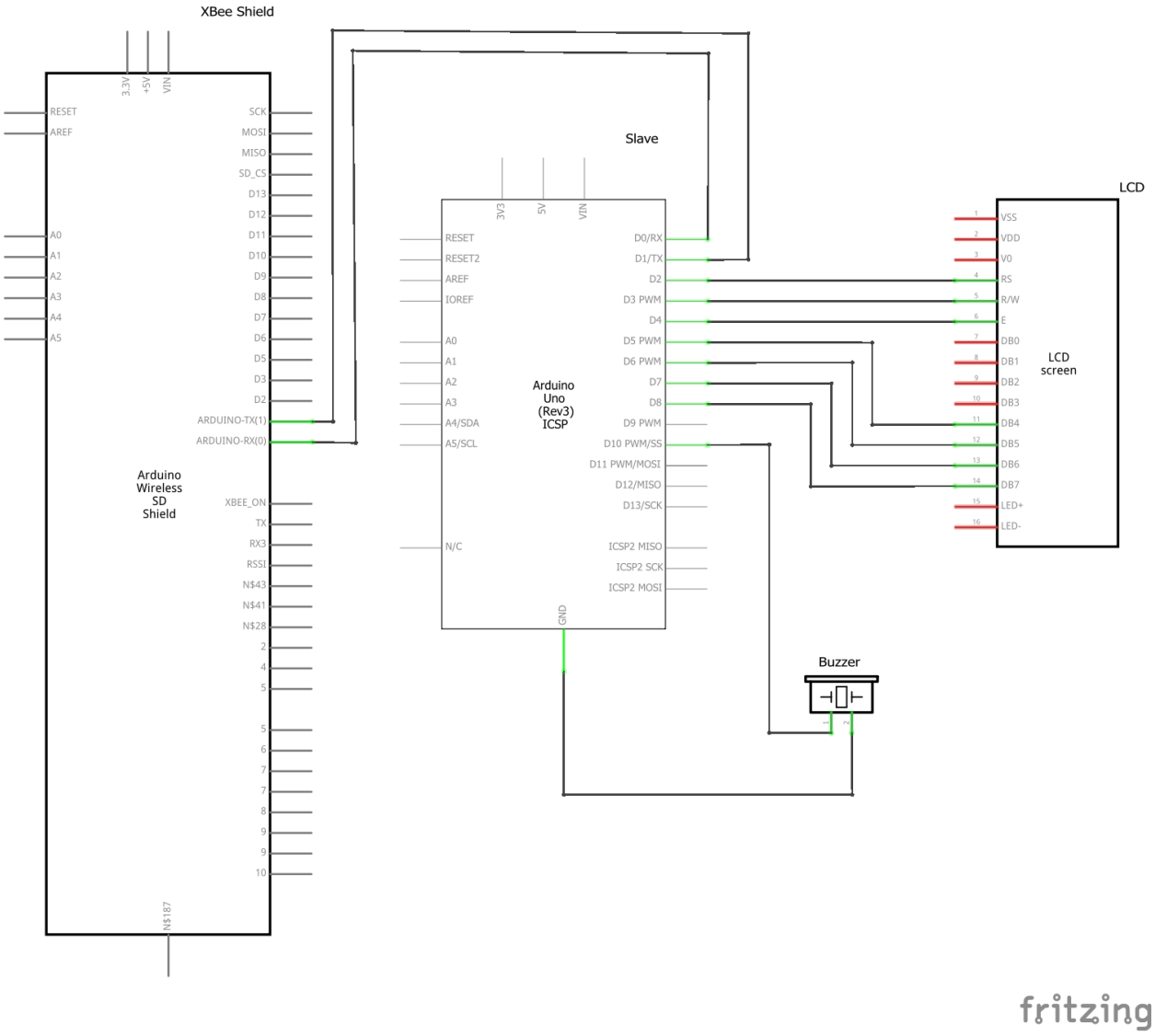
In this final project, we created our own design problem and solved it using all of the tools that we have learned throughout the labs this quarter. We implemented the game Wheel of Fortune using the Arduino. The features we were required to implement for this project were the Xbee wireless communication interface, serial monitor, LCDs, and two new features that had not been used in previous labs. In this project, new features included the tilt sensor, buzzer, and potentiometer. This project was a three week process over the end of autumn quarter.

Appendices

Schematics for master device:



Schematics for slave device:



Code for Slave device:

```
// EE 299 Lab 4
// Feifan Qiao, Mitchell Szeto, Bert Zhao
// This is the code for slave device. On receiving transmission from the master device,
// the slave device spins the wheel and determins who much each guess is worth.
// It also displays the amount of money each player have.
// Last modified: 12/11/2018

#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7, 8); // bus 1
```

```
#define SCREEN_LENGTH 16
String wheelDisplay = "100|200|300|400|500|600|700|800|900|1000|1100|1200|1300|1400|1500|";
#define BUZZER 10  // one of the physical buzzer is broken, remember to check

int state = 0;
int wheelVal = 0;
int strength = 0;
int player = 0;
int score = 0;
int p1Total = 0;
int p2Total = 0;
String guessWord = "";

void setup() {
  Serial.begin(9600);
  pinMode(BUZZER, OUTPUT);
  lcd.begin(16, 2);
  lcd.display();
}

void loop() {
  if (state == 0) {
    state = readWheelInput();
  } else if (state == 1) {
    spinWheel(wheelVal, strength);
    state = 0;
  }
}

// receives and reads the wheel value strength from the master
int readWheelInput() {
   if (Serial.available() == 4) { // 2 times 2 bytes
      wheelVal = readInt();
      strength = readInt();
      Serial.println(wheelVal);
      Serial.println(strength);
      return 1;
   } else {
      return 0;
   }
}
```

```
// spins the wheel and displays it on the lcd
void spinWheel(int num, int strength) {
  String number = String(num);
  String printStr = "";
  int i = 0;
  int count = 0;
  while (printStr.substring(0, number.length()) != number || strength > count) {
    printStr = displayWheelPos(i);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(printStr);
    lcd.setCursor(0, 1);
    lcd.print('^');
    delay(50 / strength);
    i++;
    if (i == wheelDisplay.length()) {
      i = 0;
      count += 10;
    }
  }
  tone(BUZZER, 261);
  delay(100);
  noTone(BUZZER);
}

// shows the spinning of the wheel
String displayWheelPos(int num) {
  if (num + SCREEN_LENGTH < wheelDisplay.length()) {
    return wheelDisplay.substring(num, num + SCREEN_LENGTH);
  } else {
    return wheelDisplay.substring(num, wheelDisplay.length()) + wheelDisplay.substring(0,
SCREEN_LENGTH);
  }
}

// reads bytes from the master devices
// and converts them to integers
int readInt() {
  byte b1 = Serial.read();
  byte b2 = Serial.read();
  int number = b1 | b2 << 8;
  return number;
}
```

Code for master device:

```
// EE299 Lab 4
// Feifan Qiao, Mitchell Szeto, Bert Zhao
// This is the code for the master device. Pushbutton, tilt sensor, rotary potentiometer
// and a lcd are connected to the device. It prompts users for a word input, and waits for
// guesses from player 1 and player 2. The strength of the spin and the resistance of the wheel
// are sent to the slave device as two integers. The game is over when the all of the characters
// in the word are guessed.

#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7, 8);  // bus 1

// Analog/digital port connections
int buttonPin = 10;          // please avoid using pin 8 when possible, might have interference with lcd
int tiltPin = 9;
int buttonState = 0;
int tiltState = 0;
int rotationSensor = 1;

// Wheel and spin properties
#define WHEEL_SIZE 15
int wheel[WHEEL_SIZE];
unsigned int spinStrength = 0;
#define INT_MAX 65535
#define MAX_SPIN 5000
bool nextRoll = true;
int spinNumber = 0;
int divFactor = 0;

// inputting characters and words through lcd
// and rotation sensor
int cursorIndex = 0;
int prevCursorIndex = -1;
int page = 1;                // 1 for the first screen and 2 for second screen, initialized to 1
const int asciiStart = 97;
const int asciiEnd = 122;
const int cursorStep = 40;      // cursor moves for every increment of 40 out of 1024 rotation sensor values
unsigned long firstTime = 0;
int wordLength = 0;
int nClick = 0;
```

```
String phrase;
String updatedPhrase;

struct Player {
  char guess;
  int guessValue;
  int state;
  int score;
};
struct Player p1 = {0, 100, 0, 0};
struct Player p2 = {0, 100, 0, 0};
/*
 * Key for the player state
 * 0 = player shakes tilt sensor to spin the wheel
 * 1 = player waits while the wheel scrolls
 * 2 = player enters a letter to guess
 * 3 = checks the guess with the phrase
 * 4 = checks the game state and determines if game is over
 * 5 = clears the serial buffer
*/

int gameState = 0;
/*
 * Key for the game state
 * 0 = game start and setup
 * 1 = player 1's turn
 * 2 = player 2's trun
 * 3 = game if complete
 * 4 = hold state
*/

int spinNum = 0; // Number of spins the wheel goes through

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
  pinMode(tiltPin, INPUT);
  makeWheel();
  lcd.begin(16, 2);
//  printWheel(); for debugging
  lcdDisplay("Welcome to the Wheel of Fourtune");
  lcdDisplay("Enter in phrase");
}
```

```
void loop() {
//  Serial.print("gameState = "); for debugging
//  Serial.println(gameState);
   divFactor = (analogRead(rotationSensor)/10) + 1;
//  Serial.println(divFactor); for debugging
  if (gameState == 0) {
    inputWord();
    lcdDisplay(phrase);
    createUpdatedPhrase();
    gameState = 1;
    p1.state = 0;
    lcdDisplay("Player 1's turn");
    lcdDisplay("The state of the word:");
    lcdDisplay(updatedPhrase);
    delay(2000);  // so that button state read is not tangled together
  } else if (gameState == 1) {
    // player 1's turn
    // spin wheel and guess letter or whole phrase
    // count money
    // send player 1's stats to slave LCD
    // if all words guessed gamestate = 4
    // else gamestate = 3 back to player 2
    if (p1.state == 0) { // player 1 spins the wheel
      spinNum = spin();
      if (buttonState) {
        p1.state = 1;
      }
    } else if (p1.state == 1) { // Scrolls through the array
      p1.guessValue = scrollArray();
      lcdDisplay(p1.guessValue);
      sendInt(p1.guessValue);  // sending data to slave
      sendInt(spinStrength);   // sending data to slave
      lcdDisplay("Player 1 guess a letter");
      p1.state = 2;
    } else if (p1.state == 2) { // Player enters a guess
      char temp = pickCharacter();
      delay(100);  // please don't remove, otherwise the cursor on lcd updates too fast and is not going to
show
      if (temp != NULL) {
        p1.guess = temp;
        lcdDisplay("You guessed:    ");
        lcdDisplay(String((char)p1.guess));
```

```
//        Serial.print("available = ");
//        Serial.println(Serial.available());
          spinNum = 0;
          p1.state = 3;
        }
      } else if (p1.state == 3) { // Checks the guess and updates the word and score
        int correct = checkGuess(p1.guess);
        p1.score += (correct * p1.guessValue);
        //sendInt(1);           // sending data to slave
        //sendInt(p1.score);    // sending data to slave
        //writeString(phrase);   // sending data to slave
        lcdDisplay("Player 1's score is: ");
        lcdDisplay(p1.score);
        p1.state = 4;
      } else if (p1.state == 4) { // Checks the game state and moves to player 2's turn or the game is over
        updateGuess(p1.guess);
        lcdDisplay("The state of the word:");
        lcdDisplay(updatedPhrase);
        bool game = gameOver();
        if (game) {
          gameState = 3;
        } else {
          lcdDisplay("Player 2's turn");
          spinStrength = 0;
          gameState = 2;
          p2.state = 0;
        }
      }
    } else if (gameState == 2) {
      // player 2's turn
      // spin wheel and guess letter or whole phrase
      // count money
      // send player 1's stats to slave LCD
      // if all words guessed gamestate = 4
      // else gamestate = 2 back to player 1
      if (p2.state == 0) {
        spinNum = spin();
        if (buttonState) {
          p2.state = 1;
        }
      } else if (p2.state == 1) {
        p2.guessValue = scrollArray();
        lcdDisplay(p2.guessValue);
```

```
      sendInt(p2.guessValue);  // sending data to slave
      sendInt(spinStrength);   // sending data to slave
      lcdDisplay("Player 2 guess a letter");
      p2.state = 2;
    } else if (p2.state == 2) {
      char temp = pickCharacter();
      delay(100);  // please don't remove, otherwise the cursor on lcd updates too fast and is not going to
show
      if (temp != NULL) {
        p2.guess = temp;
        lcdDisplay("You guessed:    ");
        lcdDisplay(String((char)p2.guess));
//        Serial.print("available = ");
//        Serial.println(Serial.available());
        spinNum = 0;
        p2.state = 3;
      }
    } else if (p2.state == 3) {
      int correct = checkGuess(p2.guess);
      p2.score += (correct * p2.guessValue);
      //sendInt(1);           // sending data to slave
      //sendInt(p1.score);    // sending data to slave
      //writeString(phrase);   // sending data to slave
      lcdDisplay("Player 2's score is: ");
      lcdDisplay(p2.score);
      p2.state = 4;
    } else if (p2.state == 4) {
      updateGuess(p2.guess);
      lcdDisplay("The state of the word:");
      lcdDisplay(updatedPhrase);
      bool game = gameOver();
      if (game) {
        gameState = 3;
      } else {
        lcdDisplay("Player 1's turn");
        spinStrength = 0;
        gameState = 1;
        p1.state = 0;
      }
    }
  } else if (gameState == 3) {
    // game over
    // declare winner and money won
```

```
    // start new game?
    lcdDisplay("The game is over");
    lcdDisplay("Player 1's final score: ");
    lcdDisplay(p1.score);
    lcdDisplay("Player 2's final score: ");
    lcdDisplay(p2.score);
    //sendInt(p1.score);    // sending data to slave
    //sendInt(p2.score);    // sending data to slave
    gameState = 4;
  } else if (gameState == 4) {
    // holding state
  }
}

// Creates the array with the values of the spins
void makeWheel() {
  for (int i = 0; i < WHEEL_SIZE; i++) {
    wheel[i] = (i + 1) * 100;
  }
}

// Prints the values in the wheel array, used for debugging
void printWheel() {
  for (int i = 0; i < WHEEL_SIZE; i++) {
    lcdDisplay(wheel[i]);
  }
}

// Spins the wheel based on the number of tilt shakes
int spin() {
  buttonState = digitalRead(buttonPin);
  tiltState = digitalRead(tiltPin);
  if (spinStrength < MAX_SPIN) {
      if (nextRoll && tiltState) {
      nextRoll = false;
      spinStrength++;
    } else if (!tiltState) {
      nextRoll = true;
    }
  }
  if (buttonState) {
    spinNumber = wheelSlots();
//    Serial.println(spinStrength);
```

```
//    Serial.println(spinNumber);
  }
  return spinNumber;
}

// Determines the number of slots the wheel spins through
int wheelSlots() {
  return spinStrength / divFactor;
}

// Scrolls through the array based on the spin number and
// prints the values to the serial monitor
int scrollArray() {
  int i = 0;
  int j = 0;
  while (i < spinNum) {
    // Serial.println(wheel[j]);
    i++;
    j++;
    delay(100);
    if (j >= WHEEL_SIZE) {
      j = 0;
    }
  }
  lcdDisplay(wheel[j]);
  return wheel[j];
}

// Checks the phrase with the guess to determine and return the number of occurances of
// the guess. Returns 0 if the letter has already been guessed
int checkGuess(char guess) {
  int i = 0;
  int result = 0;
  while (i < updatedPhrase.length()) {
    if (updatedPhrase.charAt(i) == guess) {
      return 0;
    }
    i++;
  }
  i = 0;
  while (i < phrase.length()) {
    if (phrase.charAt(i) == guess) {
      result++;
```

```
    }
    i++;
  }
  return result;
}

// Creates a string of asterisks with the same number of characters
// as the string that is to be guessed
void createUpdatedPhrase() {
  // Serial.println(phrase.length());
  for (int i = 0; i < phrase.length(); i++) {
    updatedPhrase.concat("*");
  }
}

// Updates the updatedPhrase with the acctual letter if it was guessed correctly
void updateGuess(char guess) {
  for (int i = 0; i < phrase.length(); i++) {
    if (phrase.charAt(i) == guess) {
      updatedPhrase.setCharAt(i, guess);
    }
  }
}

// Determines if the game is over based on the number of unguessed words
bool gameOver() {
  int i = 0;
  int unguessedLetters = 0;
  while (i < updatedPhrase.length()) {
    if (updatedPhrase.charAt(i) == '*') {
      unguessedLetters++;
    }
    i++;
  }
  if (unguessedLetters) {
    return false;
  } else {
    return true;
  }
}

// word length is limited to 16 characters
// user scrolls through characters
```

```
// single click for selecting current character
// double click to finish inputting word
void inputWord() {
  do {
    int rotationVal = analogRead(rotationSensor) / cursorStep;
    int displayVal = rotationVal + asciiStart;
    lcd.setCursor(wordLength, 0);
    lcd.print((char)displayVal);
    lcd.setCursor(wordLength, 1);
    lcd.print("^");
    if (wordLength != 0) {
      lcd.setCursor(wordLength - 1, 1);
      lcd.print(" ");
    }

    nClick = buttonDoubleClick();
    if (nClick == 1) {
      phrase += String((char)displayVal);
      wordLength++;
    }
  } while (nClick != 2);
  // clears the cursor after finishing
  lcd.setCursor(wordLength, 1);
  lcd.print(" ");
}

// displays the alphabet on the screen for the player to choose
// returns the character chosen by the player
char pickCharacter() {
  if (prevCursorIndex != -1) {
    if (page == 1) {
      lcd.setCursor(prevCursorIndex, 1);
    } else {
      lcd.setCursor(prevCursorIndex - 16, 1);
    }
    lcd.print(" ");
  }

  if (page == 1) {
    for (int i = asciiStart; i < asciiStart + 16; i++) {
      lcd.setCursor(i - asciiStart, 0);
      lcd.print((char)i);
    }
```

```
  } else {
    for (int i = asciiStart + 16; i <= asciiEnd + 6; i++) {
      lcd.setCursor(i - asciiStart - 16, 0);
      if (i > asciiEnd) {  // clears the any character displayed beyond 'z'
        lcd.print(" ");
      } else {
      lcd.print((char)i);
      }
    }
  }

  cursorIndex = analogRead(rotationSensor) / cursorStep;
  prevCursorIndex = cursorIndex;

  if (cursorIndex <= 15) {
    page = 1;
    lcd.setCursor(cursorIndex, 1);
    lcd.print("^");
  } else {
    page = 2;
    lcd.setCursor(cursorIndex - 16, 1);
    lcd.print("^");
  }

  if (digitalRead(buttonPin) == HIGH) {
    return cursorIndex + asciiStart;
  } else {
    return NULL;
  }
}

// returns 2 if button is pressed twice within 2 seconds
// returns 1 if button is pressed once within 2 seconds
// returns 0 if button is not pressed
int buttonDoubleClick() {
  if (digitalRead(buttonPin) == HIGH) {
    firstTime = millis();
    delay(200);
    while (millis() - firstTime <= 2000) {
      if (digitalRead(buttonPin) == HIGH) {
        return 2;
      }
    }
```

```
    return 1;
  }
  return 0;
}
// displays the given string on the display
void lcdDisplay(String input) {
  lcd.home();
  if (input.length() <= 16) {
    lcd.print(input);
  } else {
    lcd.print(input.substring(0, 16));
    lcd.setCursor(0, 1);
    lcd.print(input.substring(16, input.length()));
  }
  delay(2000);
  lcd.clear();
}

// displays the given integer on the display
void lcdDisplay(int input) {
  lcd.home();
  lcd.print(input);
  delay(2000);
  lcd.clear();
}

// writes integer to the slave device
void sendInt(int input) {
  byte buf[2];
  buf[0] = input & 255;
  buf[1] = (input >> 8)  & 255;
  Serial.write(buf, sizeof(buf));
}

// writes string to the slave device
void writeString(String stringData) { // Used to serially push out a String with Serial.write()

  for (int i = 0; i < stringData.length(); i++)
  {
    //Serial.println(stringData[i]);
    Serial.write(stringData[i]);   // Push each char 1 by 1 on each loop pass
  }
```

}