ECS511U: Creating Interactive Objects Individual Coursework Document

Awais Amjad 210379897

PART A Code

```
int ledPins[] = { 2, 3, 4, 5, 6, 7, 8, 9 };  // Pins used for LED's
const int ship_size = 3;                      // size of Battleship
int locationCells[ship_size];                 // makes array size of the
ship_size variable
int numberOfHits = 0;                         // number of times the
Battleship has been hit
int winPin = 10;                              // Pin number for the
Green/Win LED
int numberOfGuesses = 5;                      // number of guesses the user
has

void setup() {
  for (int i = 0; i < sizeof(ledPins); i++) {
    pinMode(ledPins[i], OUTPUT);
  }
  pinMode(winPin, OUTPUT);
  randomSeed(analogRead(A3));
  Serial.begin(9600);
  setLocationCells();
}

void loop() {
  String final_result = checkYourself();
  Serial.println(final_result);
  delay(500);
  if (final_result == "kill") {  // if the final result is a kill exit
game, declare user as winner
    numberOfGuesses = 0;
    Serial.println("Press Reset button to restart");
    delay(500);
    exit(1);
  } else if (numberOfGuesses == 0) {  // if the user has ran out of
guesses exit game, declare user as loser
    Serial.println("You lose");
    Serial.println("Press Reset button to restart");
    delay(500);
```

```
    exit(1);
  }
}

char getGuess() {
  char readinput = '^';
  while ((readinput < '1') || (readinput > '8')) {  // makes sure guess is
within bounds
    Serial.println("Enter a guess (1-8):");         //prompts user for
guess within bounds
    Serial.print("Number of guesses remaining:");   //shows remaining
number of guesses
    Serial.println(numberOfGuesses);
    while (Serial.available() == 0) {}
    readinput = Serial.read();  // gets users guess
    Serial.print("You have entered: ");
    Serial.println(readinput);  // diplays what the user entered
    numberOfGuesses--;
    if ((readinput < '1') || (readinput > '8')) {
      Serial.println("Invalid input. Please enter number 1-8");  // if
guess is not within bounds it
    }                                                            // tells
user this
  }
  return readinput;  // returns guess
}

void setLocationCells() {                    //Create random places for
ship
  int startLocation1 = random(1, 6);         //generate random number 1-5
  int startLocation2 = startLocation1 + 1;  // next number for battleship
is 1 after first number
  int startLocation3 = startLocation1 + 2;  // next number for battleship
is 2 after first number
  locationCells[0] = startLocation1;
  locationCells[1] = startLocation2;
  locationCells[2] = startLocation3;
  Serial.println("Best score will be 3 guesses");  // tells the user what
the best score is
}
```

```
String checkYourself() {
  char userGuess = getGuess();
  String outcome = "MISS";
  int guess = (userGuess - '0');  //convert user guess to int
  for (int i = 0; i < ship_size; i++) {
    if (guess == locationCells[0] || guess == locationCells[1] || guess ==
locationCells[2]) {
      numberOfHits++;
      digitalWrite(ledPins[guess - 1], HIGH);
      if (numberOfHits == ship_size) {
        outcome = "KILL. You WON!!!!!!!";
        digitalWrite(winPin, HIGH);
      } else outcome = "HIT";
      break;
    }
    return outcome;
  }
}
```

PART B Code

```
const int ledPins[] = { 2, 3, 4, 5, 6, 7, 8, 9 };  // array of pins used
for LEDs
const int ship_size = 3;                            // size of the ship
int locationCells[ship_size];                       // array with size of
the ship
int numberOfHits = 0;
const int buttonPin1 = 12;
const int buttonPin2 = 13;
int guessButton = 0;
int number;
int winPin = 10;
int numberOfGuesses = 5;
int speakerPin = 11;
int length = 15;         // the number of notes
char hitNotes[] = "aa";  // note played when guess is correct
char missNotes[] = "c";
char winMusic[] = "ccdcfeccdcgf ";  // note played when guess is wrong
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
```

```
int tempo = 300;

void setup() {
  // put your setup code here, to run once:
  for (int i = 0; i < sizeof(ledPins); i++) {
    pinMode(ledPins[i], OUTPUT);
  }
  randomSeed(analogRead(A0));
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(speakerPin, OUTPUT);
  Serial.begin(9600);
}
//declared after cuz of randomSeed(analogRead(A3));
int startLocation1 = random(1, 6);        // start location with random
number between 1 and 5
int startLocation2 = startLocation1 + 1;  // next number for battleship is
1 after first number
int startLocation3 = startLocation1 + 2;  // next number for battleship is
2 after first number
int hit;
void loop() {
  // // GUESS BUTTON
  if (digitalRead(buttonPin1) == LOW) {  // if the button state is
low/button has been pressed then execute code
    guessButton++;
    delay(200);                              // delay when LED changes
    number = constrain(guessButton, 0, 8);  // constrains number of
guesses between 0 and 8
    number = number - 1;
    digitalWrite(ledPins[number], HIGH);  // turns on LED
    delay(50);
    digitalWrite(ledPins[number - 1], LOW);  //turns off previous LED
    if (startLocation1 == ledPins[number - 1] || startLocation2 ==
ledPins[number - 1] || startLocation3 == ledPins[number - 1]) {
      digitalWrite(ledPins[number], HIGH);  //if guess matches battleship
location light LED
      hit++;
    }
```

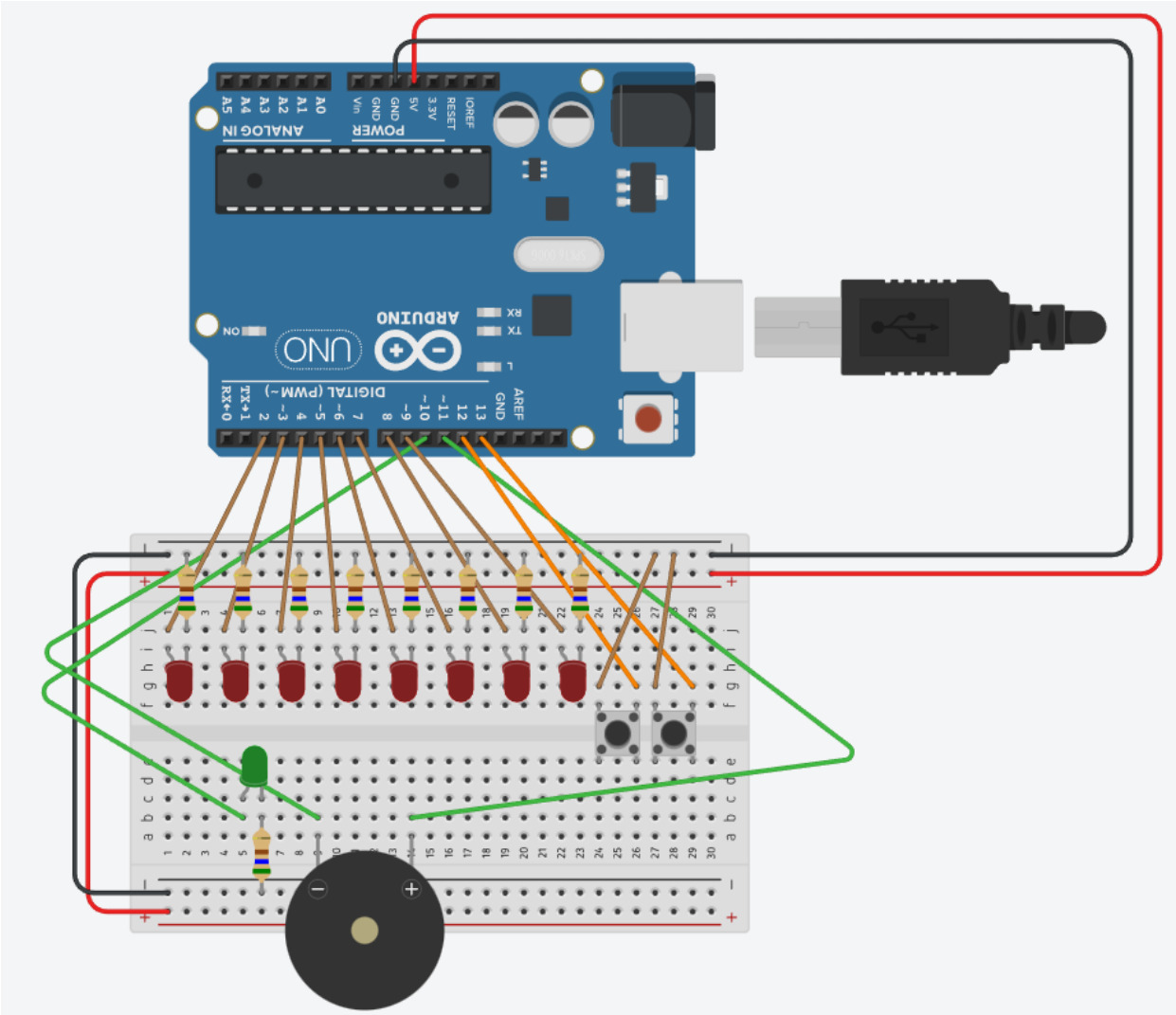
```
    if (guessButton > 8) {  // if the button is at last LED and is
pressed, it will return back to first LED
      digitalWrite(ledPins[number], LOW);
      guessButton = 0;
    }
  }
  // CONFIRM BUTTON
  if (digitalRead(buttonPin2) == LOW) {  // if the button state is
low/button has been pressed then execute code
    if (startLocation1 == ledPins[number - 1] || startLocation2 ==
ledPins[number - 1] || startLocation3 == ledPins[number - 1]) {
      Serial.println("HIT");  // Prints hit when guess is right
      for (int i = 0; i < sizeof(hitNotes); i++) {
        if (hitNotes[i] == ' ') {
          delay(beats[i] * tempo);  // rest
        } else {
          playNote(hitNotes[i], beats[i] * tempo);  // play note when
guess is right
          digitalWrite(ledPins[number - 1], HIGH);  // light up LED when
guess is right
        }
      }
      if (hit == 3) {
        digitalWrite(winPin, HIGH);  // if 3 hits have landed light winPin
        Serial.println("You Won");
        for (int i = 0; i < sizeof(winMusic); i++) {
          playNote(winMusic[i], beats[i] * tempo);  // play music for
winning
        }
      }
    }

    else {
      Serial.println("MISS");  // Prints miss when guess is right
      for (int i = 0; i < sizeof(missNotes); i++) {
        playNote(missNotes[i], beats[i] * tempo);  // play the note for
wrong guess
      }
    }
  }
```

```
}

void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}
void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 900, 1014, 956 };

  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}
```



DESIGN CHOICES

Part A:

1. I decided an array of 1X8 LED's would be best and gave each LED its own digital pin
2. I kept all the LED's red to not confuse the user as it may be assumed different LED's had separate meanings
3. I decided on 5 guesses as this was an optimal number to give the user a chance to make a mistake as well make sure they still had to choose their guesses carefully.
4. I decided on giving the user feedback such as "Guesses remaining", "Best score to achieve" and whether they had hit or missed their target to help them keep track of their progress and give reason to put in effort and retry.

Part B (Includes Part A):

1. I used a piezo as I didn't want the user to only have to look at the screen but also use another one of their senses to understand their progress
2. The piezo also adds a musical element to the game which is fun and positively received by users

3. I decided on buttons for guessing they are common well known items that are very easy to use compared to other items such as potentiometers
4. Using buttons gave me the possibility of using more than 1 which allowed for more functionality

USER MANUAL

Part A:

1. You are prompted to enter a guess between 1 and 8 (including 8) into the Serial Monitor.
2. You are given information on how many guesses you have and what's the best score to achieve
3. There will be 5 attempts to hit the ship with the ship requiring 3 hits to kill. Failure to do so results in a loss
4. If you win, a green LED will light up indicating victory
5. If you lose, it will be printed in the Serial Monitor
6. After either winning or losing, the user will be given to restart the game through the restart button

Part B:

1. Use the left button to move through the LED's and the right button to confirm your choice
2. The LED's restart at original position when gone through the full array by pressing the left button once at the end.
3. If the guess is wrong, a note will be played to signify this as well as "MISS" printed in Serial Monitor
4. If the guess is right, a different note will be played to signify this as well, "HIT" printed in Serial Monitor as well as the LED staying lit
5. If 3 hits are achieved, a short song will play, a green LED will light up and it will be said on the Serial Monitor

SHORTCOMINGS

Part A:

- The same number can be entered 3 times to give a win unlike the real game
- If an invalid input is given as the end input, number of guesses becomes negative
- LED 8 turns on when won due to winPin being turned on. I printed winPin and it is being read as digital pin 10 yet LED pin 8, which is digital pin 9, still turns on. I used tinkerCad to see and it is not an issue there. The wiring looks good so i am unsure what the problem is

Part B:

- There is no maximum amount of tries so the users performance is judged on how many tries they had
- When hit, the LED lights up the previous LED and not the correct LED
- The notes can be very different when played multiple times even of nothing is changed

TESTING

Part A:

1. I entered a value above 8 and below 1 to see the outcome. Outcome was “Invalid Number”. **It worked**
2. Printed random values to see if the random function was working. It was repeating the same value so I looked online on how to fix it, found a solution and implemented it. Repeated the process and found the values were now random and the function was working properly.**It worked**
3. Had my brother test my game to see if there were any problems I couldn't see. He pointed out the spacing between texts was confusing so I changed them to println(). **It worked**

Part B:

1. Tried different notes by tweaking and tuning the numbers to get an appropriate sound. **It worked**
2. Pressing the button repeatedly to see if it repeats. **It worked**
3. To get values working such as the index of the LED, if the battleship had been hit 3 times, if the notes were playing and if values were matching their type, I printed them in the Serial Monitor to get a better understanding. **It worked**