Not Simon Says | 5

Electrical and Computer Engineering 5

Custom Lab 5

Developed by Marcus Higashi, Kevin Shin, Herman Torres

Overview

The goal of this lab is to create a fully functioning game that is a mix of Simon Says and Bop-It! on some breadboards, and it will be run on Arduino. This lab will incorporate many aspects of ECE. This includes using Eagle CAD to create a schematic and PCB design, circuit building, and Arduino programming.

What You Will Need

Materials:

- Buttons
- LEDs
- Resistors
- Jumper wires

Optional Materials (Challenge 6):

- Buttons
- LEDs
- Resistors
- Jumper wires

Machinery:

- Computer
- Arduino MEGA2560

Software:

- Eagle CAD
- Fritzing
- Arduino IDE

Other:

- Fast Reaction Time Recommended, but not necessary

Challenge #1: Schematic and PCB

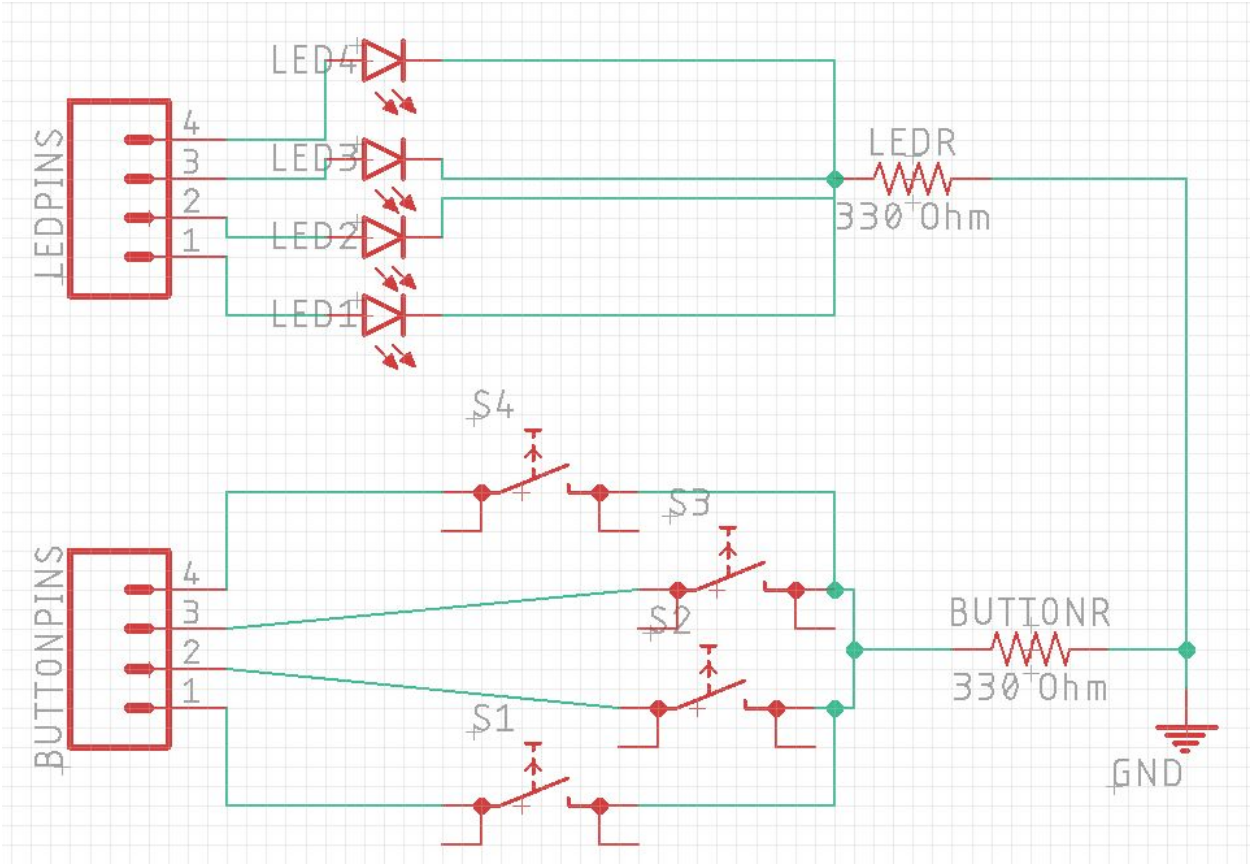
Objective

The objective of this section is to demonstrate understanding of Eagle CAD and plan out how the connections for this project will work.

Components

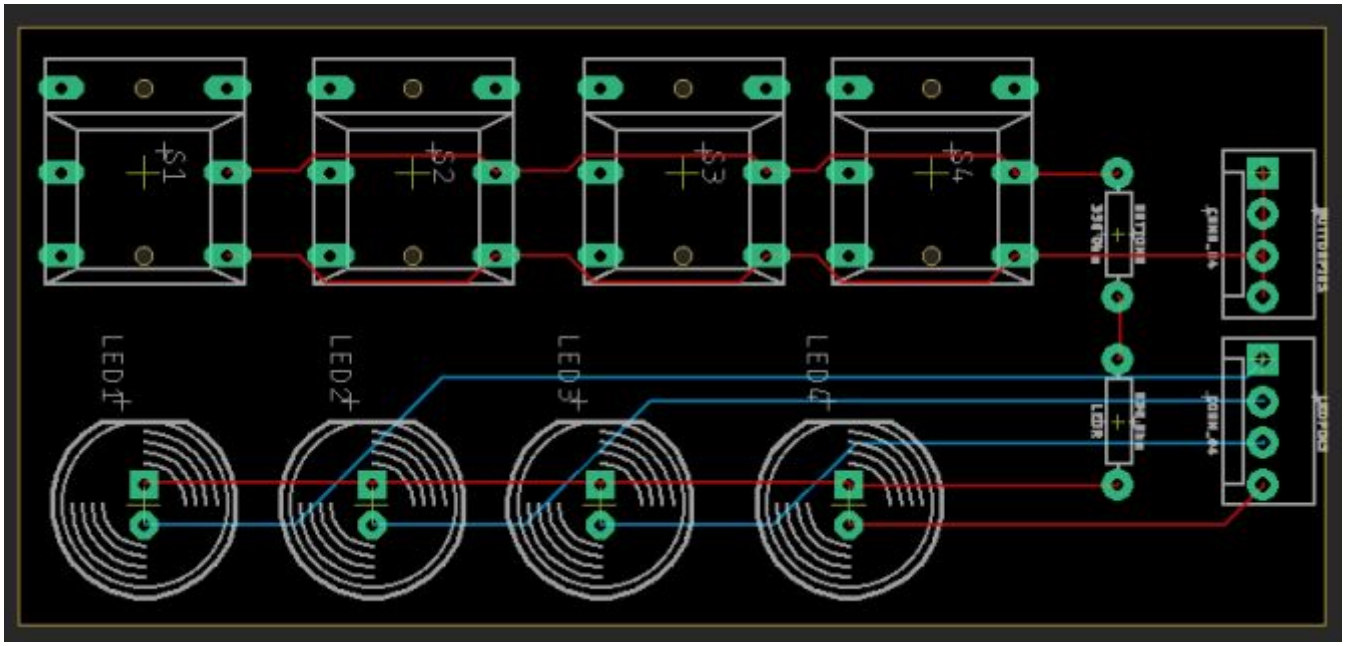
- Eagle CAD

Schematic:



Note: This mainly uses parts from the ECE 5 Tutorial and ECE 5 WS libraries, but the LEDs were generic from the list of other libraries, and the switches were found by searching “push switch” in the “Add Part” menu. Don’t forget to do an **Electrical Rule Check** before moving on to the next step!

PCB:



Note: Your autorouting might look slightly different than this, but as long as all of the parts are there, connected, and pass a **Design Rule Check**.

Task

- Replicate the schematic
- Replicate the PCB.

Challenge #2: Input/Output Circuit

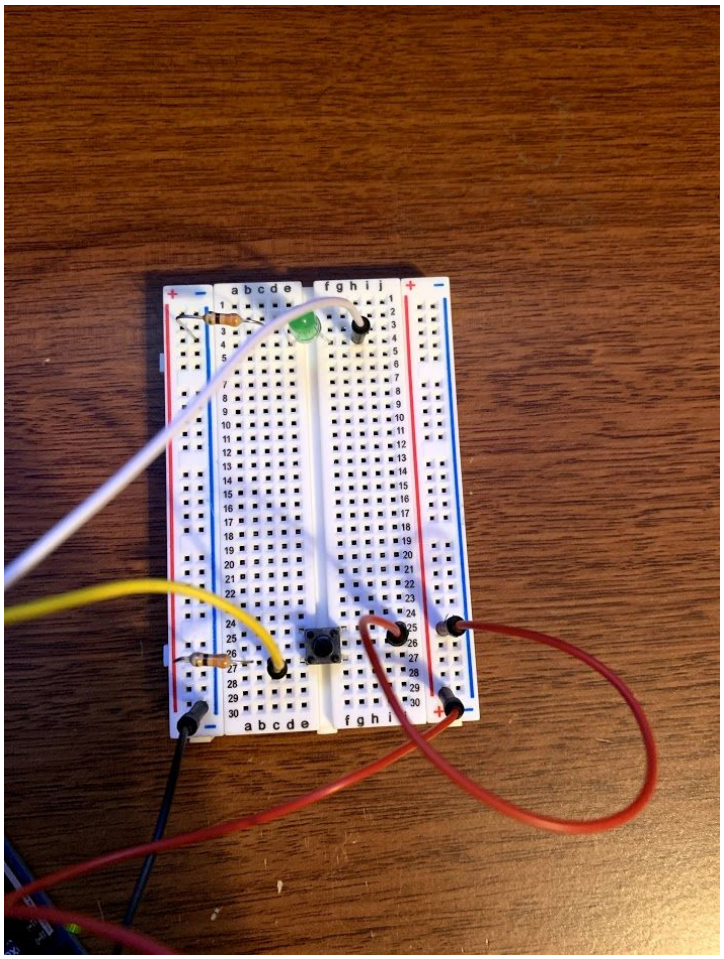
Objective

Create the following circuit. The following diagram uses the push buttons and LEDs, but you may use any inputs and outputs you wish to. Be creative.

Components

- Arduino MEGA
- 4 of the Same Input Devices
- 4 of the Same Output Devices
- Jumper Wires
- 330 Ohm Resistors

Circuit Diagram:



Note: You will need to replicate this design 4 times. To clarify, this means that there will be a total of 4 buttons with each accompanied by an LED.The buttons will be placed on the right of the current one, and the LEDs will be to the right of that LED. Additionally, the color of your wiring and LEDs is dependent on you. (Black wire=ground, Red wire=5V, Yellow wire=button pin, White wire=led pin).

Task

- Create the circuit above with the addition of 3 more buttons and LEDs.

Challenge #3: Coding Not Simon Says

Objective

Code the main foundation of this project. You will have to read and understand the large sections of code and as well fill in the highlighted sections of code. Add the additional codes to the end of the previous coding challenge.

Components

- Arduino Program

Coding Challenge:

```
#define NUM_OF_DEVICES 4

const int ledPins[NUM_OF_DEVICES] = {13,12,11,10};
const int buttonPins[NUM_OF_DEVICES] = {7,6,5,4};
int buttonState[NUM_OF_DEVICES] = {0,0,0,0};
int number,

void setup() {
  Serial.begin(9600);
  for(int i = 0; i < NUM_OF_DEVICES; i++)
  {
    pinMode(ledPins[i], OUTPUT);
    pinMode(buttonPins[i], INPUT);
  }

  game();
}
```

Note: Constants define the sum of input/ output devices being used. PinModes are being initialized and the game function is called. The blank line of code is the defined integer that would be the sum of the players points.

Code add-on:

```
int srandLED(int level)
{
  number = random(0,4);
  digitalWrite(ledPins[number],HIGH);
  delay(ledDuration(level));
  digitalWrite(ledPins[number], LOW);
  delay(500);
  return number;
}
int ledDuration(int level)
{
  if(level == 0)
  {
    return 5000; // Determined by user
  }
  else if (level != 0)
  {
   // return 5000  How much to decrement by
  }
}
```

Note: This srandLED function allows for there to be a random number chosen between 0 to 3. The choosing of this random number would then determine which LED would be turned on. The function ledDuration would allow the led to be turned on and remain on for a certain number of milliseconds determined by the level of the game the user is on.

Code add-on:

```
void game(){
  int gameMode;
  int selection = 0;

    Serial.println("Welcome to the Not Simon Says Game!");
    Serial.println("Please choose the setting you would like to play in:[0] for normal, [1] to exit");
    while(true)
    {

      buttonState[0] = digitalRead(buttonPins[0]);
      buttonState[1] = digitalRead(buttonPins[1]);
      if(buttonState[0] == 1)
      {
        gameMode = 1;
        gameType(gameMode);
        break;
      }
      else if(buttonState[1] == 1)
      {
        Serial.print("Thanks for playing!\n");
        break;
      }

  }

}
```

Note: This game function not only introduces the user to the game, but as well determines what mode the user selects to play the game. The function reads the input of the user's selection of game mode and calls upon the next function named "gameType."

Coding Challenge:

```
void gameType(int gameSetting)
{

  int levels = 0;
  [redacted]
  int number;
  Serial.print("Get ready to start in 5 seconds!\n");
  for(int i = 5; i > 0; i--)
  {
    Serial.print(i);
    Serial.println("...");
    delay(1000);
  }
 if(gameSetting == 1)
 {
   while(true)
   {
     buttonState[0] = 0;
     buttonState[1] = 0;
     buttonState[2] = 0;
     buttonState[3] = 0;
     [redacted]
     [redacted]
     number = srandLED(levels);
     buttonState[0] = digitalRead(buttonPins[0]);
```

Note: This section of code initializes the value of the functions before the next lines of code are run. Following the initialization, the code tells the user that the game is going to begin in 5 seconds, which the for loop is the count down and printing of each number starting from 5. Depending on the gamemode selection made by the user, the next segment of code represents the while loop if the user chooses 1 as their selection. The code within the while loop initializes the buttonstates to 0 and your code. You are to fill in the blanks with code that prints the current score of the user in the serial monitor.

Task

- Compile and finish all blanks in presented code to build a working program for our game.

Challenge #4: Score

Objective

Create code that will present the users score in the serial monitor by filling in the blanks within the presented code.

Components

- Arduino Program
- Arduino Serial Monitor

Score Tracking Code:

```
buttonState[0] = digitalRead(buttonPins[0]);
if(buttonState[0] == HIGH && number == 0)
{
  [          ]
}
else if(buttonState[0] == HIGH && number != 0)
{
  [                                        ]
}

if([      ] == 5) // User chooses
{
  levels++;
  Serial.print("The level is now ");
  Serial.print(levels);
  Serial.print("!\n");
  [            ]
}
```

Note: This section of code reads the buttonPin to determine whether it is on and corresponds with the number then the player score should be increased by one. If the buttonPin reads the button to be off and the number is not corresponding correctly, the game will then end and present the final score. Once the users score reaches 5, the level of the game should be increased by one and the player score should be reset to 0. You are going to repeat this code for 4 of the buttons and number 1, 2, 3.

Task

- Successfully fill in the blank lines of code and repeat the code 3 more times for the other buttons.

Challenge #5: Two Sets Of Output

Objective

Add on to you challenge 3 code and introduce the ability to switch output devices. This will require you to use an additional breadboard with new outputs of your choice.

Components

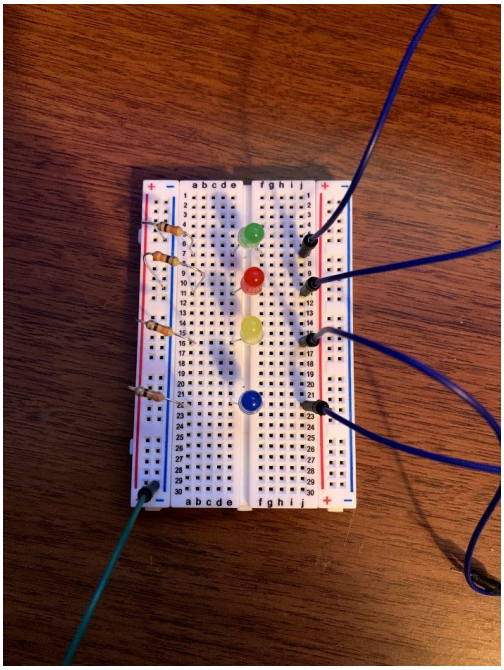
- Arduino program
- Jumper wires
- Leds or outputs of your choice
- 330 ohm resistors

Code:

```
const int ledPins2[NUM_OF_DEVICES] = {9,8,3,2};
pinMode(ledPins2[i], OUTPUT);

a = random(0,2);
if(a == 0)
{
  number = srandLED(levels);
}
else if(a != 0)
{
  number = srandLED1(levels);
}
```

Block of Code to Incorporate another input (May or may not have fill ins)



Note: This code segment allows the program to randomize and choose what output circuit is going to be used. The new pins are intialized and listed.

Task

- Add additional code and circuit to expand the options of the output devices for the game to interact with.

Code

```
const int ledPins[NUM_OF_DEVICES] = {13,12,11,10};
const int ledPins2[NUM_OF_DEVICES] = {9,8,3,2};
const int buttonPins[NUM_OF_DEVICES] = {7,6,5,4};
int buttonState[NUM_OF_DEVICES] = {0,0,0,0};
int number, ______;

int ledDuration(int level)
{
  if(level == 0)
  {
    return 5000; // Determined by user
  }
  else if (level != 0)
  {
   // return 5000  How much to decrement by
  }
}

int srandLED(int level)
{
  number = random(0,4);
  digitalWrite(ledPins[number],HIGH);
  delay(ledDuration(level));
  digitalWrite(ledPins[number], LOW);
  delay(500);
  return number;
}

int srandLED1(int level)
```

```
{
  number = random(0,4);
  digitalWrite(ledPins2[number],HIGH);
  delay(ledDuration(level));
  digitalWrite(ledPins2[number], LOW);
  delay(500);
  return number;
}

void gameType(int gameSetting)
{
  int levels = 0;
  _______
  int number;
  Serial.print("Get ready to start in 5 seconds!\n");
  for(int i = 5; i > 0; i--)
  {
    Serial.print(i);
    Serial.println("...");
    delay(1000);
  }
  if(gameSetting == 1)
  {
    int a;
    while(true)
    {
      buttonState[0] = 0;
      buttonState[1] = 0;
      buttonState[2] = 0;
      buttonState[3] = 0;
       _______________
       _______________
       _______________

       a = random(0,2);
       if(a == 0)
       {
         number = srandLED(levels);
       }
       else if(a != 0)
       {
         number = srandLED1(levels);
       }

      buttonState[0] = digitalRead(buttonPins[0]);
      if(buttonState[0] == HIGH && number == 0)
      {
        ________
      }
```

```
      else if(buttonState[0] == HIGH && number != 0)
      {
        _____________________________
        _________________
        _________________
        break;
      }
      buttonState[1] = digitalRead(buttonPins[1]);
      if(buttonState[1] == HIGH && number == 1)
      {
        score++;
      }
      else if(buttonState[1] == HIGH && number != 1)
      {
        Serial.print("Your final score is ");
        Serial.print(score);
        Serial.print(".\n");
        break;
      }
      buttonState[2] = digitalRead(buttonPins[2]);
      if(buttonState[2] == HIGH && number == 2)
      {
        score++;
      }
      else if(buttonState[2] == HIGH && number != 2)
      {
        Serial.print("Your final score is ");
        Serial.print(score);
        Serial.print(".\n");
        break;
      }
      buttonState[3] = digitalRead(buttonPins[3]);
      if(buttonState[3] == HIGH && number == 3)
      {
        score++;
      }
      else if(buttonState[3] == HIGH && number != 3)
      {
        Serial.print("Your final score is ");
        Serial.print(score);
        Serial.print(".\n");
        break;
      }

      if(buttonState[0] != HIGH && buttonState[1] != HIGH && buttonState[2]
!= HIGH && buttonState[3] != HIGH)
      {
        Serial.print("Too slow!\n");
        break;
```

```
      }

      if(_____ == 5) // User chooses
      {
        levels++;
        Serial.print("The level is now ");
        Serial.print(levels);
        Serial.print("!\n");
        _________
      }


    }
  }
  else if(gameSetting == 2)
  {

  }
}

void game(){
  int gameMode;
  int selection = 0;

    Serial.println("Welcome to the Not Simon Says Game!");
    Serial.println("Please choose the setting you would like to play in:[0]
for normal, [1] to exit");
    while(true)
    {

      buttonState[0] = digitalRead(buttonPins[0]);
      buttonState[1] = digitalRead(buttonPins[1]);
      if(buttonState[0] == 1)
      {
        gameMode = 1;
        gameType(gameMode);
        break;
      }
      else if(buttonState[1] == 1)
      {
        Serial.print("Thanks for playing!\n");
        break;
      }

  }

}
```

```
void setup() {
  Serial.begin(9600);
  for(int i = 0; i < NUM_OF_DEVICES; i++)
  {
    pinMode(ledPins[i], OUTPUT);
    pinMode(ledPins2[i], OUTPUT);
    pinMode(buttonPins[i], INPUT);
  }

  game();
}
```

Final circuit: