

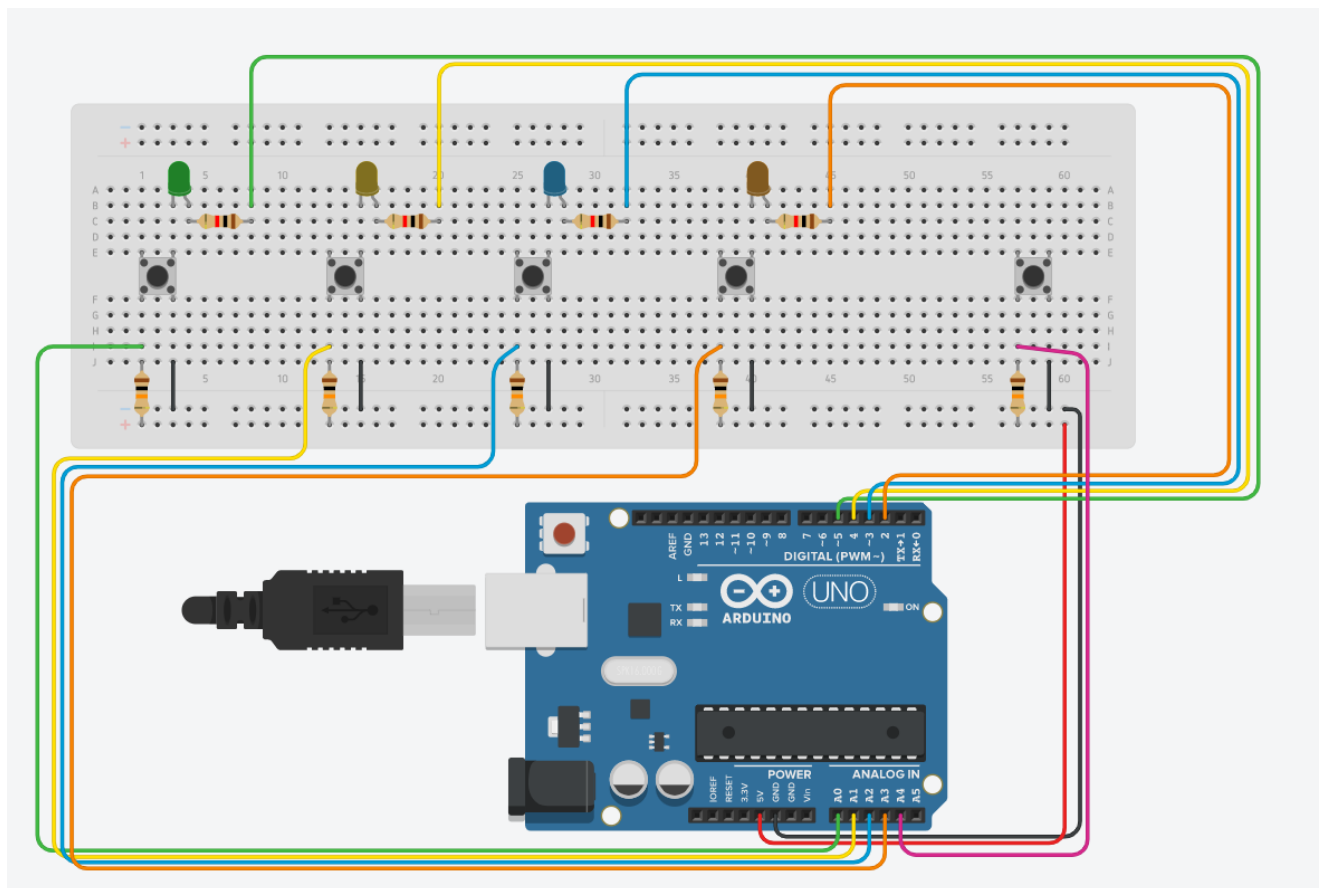
## Intro

This assignment is intended to help gain familiarity with digital input and output while using a microcontroller. So, being able to enter a command of some sort, in this case, pressing a button, and then getting a calculated and planned reaction from the device. In this case, the user will be able to press a button and see a direct response to the button being pressed in the form of a memory game where the microcontroller will flash a number of lights in a certain pattern. The user is to mimic those lights by pressing the appropriate buttons in the appropriate sequence. With each proper sequence of buttons pressed, the game gets harder.

## Demo

<https://youtu.be/CwPJSWyannc>

## Schematic



### **Explanation**

The game starts by pressing the start button, the button on the far-right side of the breadboard. This initializes the game and random sequence the game will follow when prompting the user with the blink of a random light. From here the user will press the button directly under the light that flashed. If done correctly, all lights will flash and the sequence will begin again but one extra light will blink after the first. Again the user must press each button in the proper order that the lights blinked. This cycle will repeat a few times or until the user presses a button that does not match up with the sequence of lights. In the event that the sequence is not followed, the lights will all flash once and the game will need to be restarted by pressing the right button.


## Code

```
Text 1 (Arduino Uno R3)
1  /*
2  Rico Garcia
3  Lab Assignment 4 Simple Simon
4
5  Press start button on right of bread board to begin game.
6  One light will light and give the user time to mimic the
7  action by pressing the button below the light that
8  briefly lit up. If the user presses the correct button all the
9  lights will flash and a new sequence of lights will blink. User
10 must press the buttons in the correct pattern to progress. If
11 the wrong pattern is pressed, the lights will blink and the user
12 will have to press the start button to begin the game again.
13
14 */
15
16 const int MAX_LEVEL
17     = 100;
18 int sequence[MAX_LEVEL];
19 int your_sequence[MAX_LEVEL];
20 int level
21     = 1;
22
23 int velocity = 1000;
24
25 void setup() {
26 pinMode(A0, INPUT);
27 pinMode(A1,
28     INPUT);
29 pinMode(A2, INPUT);
30 pinMode(A3, INPUT);
31
32 pinMode(2, OUTPUT);
33 pinMode(3,
34     OUTPUT);
35 pinMode(4, OUTPUT);
36 pinMode(5, OUTPUT);
37
38 digitalWrite(2, LOW);
39 digitalWrite(3,
40     LOW);
41 digitalWrite(4, LOW);
42 digitalWrite(5, LOW);
43 }
44
45 void loop()
46 {
47 // begin game
48 if
49     (level == 1)
```

```
Text [dropdown] [download] [print] [font-size] 1 (Ar...
48  if
49      (level == 1)
50  generate_sequence();
51
52  // if button pressed or user is correct
53  if (digitalRead(A4)
54      == LOW || level != 1)
55  {
56  show_sequence();
57  //demo pattern & wait for user's response
58  get_sequence();
59  }
60  }
61
62  void
63      show_sequence()
64  {
65  digitalWrite(2, LOW);
66  digitalWrite(3, LOW);
67  digitalWrite(4,
68      LOW);
69  digitalWrite(5, LOW);
70
71  for (int i = 0; i < level; i++)
72  {
73  digitalWrite(sequence[i],
74      HIGH);
75  delay(velocity);
76  digitalWrite(sequence[i], LOW);
77  delay(200);
78  }
79  }
80
81  void
82      get_sequence()
83  {
84  // correct user input
85  int flag = 0;
86
87  for
88      (int i = 0; i < level; i++)
89  {
90  flag = 0;
91  while(flag == 0)
92  {
93  if (digitalRead(A0)
94      == LOW)
95  {
96  digitalWrite(5, HIGH);
97  }
```

```
94     -- LOW)
95     {
96     digitalWrite(5, HIGH);
97     your_sequence[i] = 5;
98     flag = 1;
99     delay(200);
100    if
101      (your_sequence[i] != sequence[i])
102      {
103      wrong_sequence();
104      return;
105      }
106    digitalWrite(5,
107      LOW);
108    }
109
110    if (digitalRead(A1) == LOW)
111    {
112    digitalWrite(4, HIGH);
113    your_sequence[i]
114      = 4;
115    flag = 1;
116    delay(200);
117    if (your_sequence[i] != sequence[i])
118    {
119    wrong_sequence();
120    return;
121    }
122    digitalWrite(4,
123      LOW);
124    }
125
126    if (digitalRead(A2) == LOW)
127    {
128    digitalWrite(3, HIGH);
129    your_sequence[i]
130      = 3;
131    flag = 1;
132    delay(200);
133    if (your_sequence[i] != sequence[i])
134    {
135    wrong_sequence();
136    return;
137    }
138    digitalWrite(3,
139      LOW);
140    }
141
142    if (digitalRead(A3) == LOW)
143    {
```

```
142 if (digitalRead(A3) == LOW)
143 {
144   digitalWrite(2, HIGH);
145   your_sequence[i]
146     = 2;
147   flag = 1;
148   delay(200);
149   if (your_sequence[i] != sequence[i])
150   {
151     wrong_sequence();
152     return;
153   }
154   digitalWrite(2,
155     LOW);
156 }
157
158 }
159 }
160 right_sequence();
161 }
162
163 // generate random pattern each time
164 void generate_sequence()
165 {
166   randomSeed(millis());
167
168   for (int i = 0; i < MAX_LEVEL; i++)
169   {
170     sequence[i]
171       = random(2, 6);
172   }
173 }
174 void wrong_sequence()
175 {
176   for (int i = 0; i < 3;
177     i++)
178   {
179     digitalWrite(2, HIGH);
180     digitalWrite(3, HIGH);
181     digitalWrite(4,
182       HIGH);
183     digitalWrite(5, HIGH);
184     delay(250);
185     digitalWrite(2, LOW);
186     digitalWrite(3,
187       LOW);
188     digitalWrite(4, LOW);
189     digitalWrite(5, LOW);
190     delay(250);
```

 Serial Monitor

```
177     i++)
178     {
179     digitalWrite(2, HIGH);
180     digitalWrite(3, HIGH);
181     digitalWrite(4,
182     HIGH);
183     digitalWrite(5, HIGH);
184     delay(250);
185     digitalWrite(2, LOW);
186     digitalWrite(3,
187     LOW);
188     digitalWrite(4, LOW);
189     digitalWrite(5, LOW);
190     delay(250);
191     }
192     level
193     = 1;
194     velocity = 1000;
195     }
196
197     void right_sequence()
198     {
199     digitalWrite(2,
200     LOW);
201     digitalWrite(3, LOW);
202     digitalWrite(4, LOW);
203     digitalWrite(5, LOW);
204     delay(250);
205
206     digitalWrite(2,
207     HIGH);
208     digitalWrite(3, HIGH);
209     digitalWrite(4, HIGH);
210     digitalWrite(5, HIGH);
211     delay(500);
212     digitalWrite(2,
213     LOW);
214     digitalWrite(3, LOW);
215     digitalWrite(4, LOW);
216     digitalWrite(5, LOW);
217     delay(500);
218
219     // increase difficulty of patterns
220     if
221     (level < MAX_LEVEL);
222     level++;
223
224     velocity -= 50;
225     }
```