Light Stopping Game!

Adithya Shastry

September 27, 2018
CS342:Project 1

Contents

1 Overview **3**

2 Hardware Requirements **3**

3 Hardware Implementation **3**

4 Software Implementation **4**

5 Testing and Possible Further Improvements **7**
5.1 Testing 7
5.2 Further Improvements 7

6 Conclusion **8**

1 Overview

The goal for this project is to create a simplified version of the light stopping game found in arcades. I decided to replicate this game because I have always loved playing this game at the arcade as a kid and being able to physically build the game, however small my version was, meant a lot to me. The game is meant to allow the user to progress to faster and faster speeds as he or she keeps getting the light to stop on the correct bulb(the green bulb). The end to the game is when the player loses, by landing the light on any of the red bulbs. The game uses two inputs(in the form of two buttons) and six outputs(in the form of bulbs). Of the two buttons, one is used to reset the game, and the other used to stop the light.The ATmega micro controller is used as a medium to allow the communication between the physical pins on the board(hardware) and the internally run code(software).This was done through the use of the GPIO registries,as they exist in the memory of the ATmega chip.In the end, this project produced a fully functioning game that is very fun to play.

2 Hardware Requirements

The hardware requirements for the project are as shown in Table 3 below.

Part name	**Amount**
Red Light bulbs	**5**
Green Light bulb	**1**
Buttons	**2**
ATmega Microcontroller	**1**
Resistors(High Capacity)	**7**
Long Bread Board	**1**
Insulated Copper Wires	**20**

Table 1: Hardware Requirements

3 Hardware Implementation

The Hardware implementation is shown in schematic form in Figure 3.There are $6.8K\Omega$ resistors underneath the led lights.

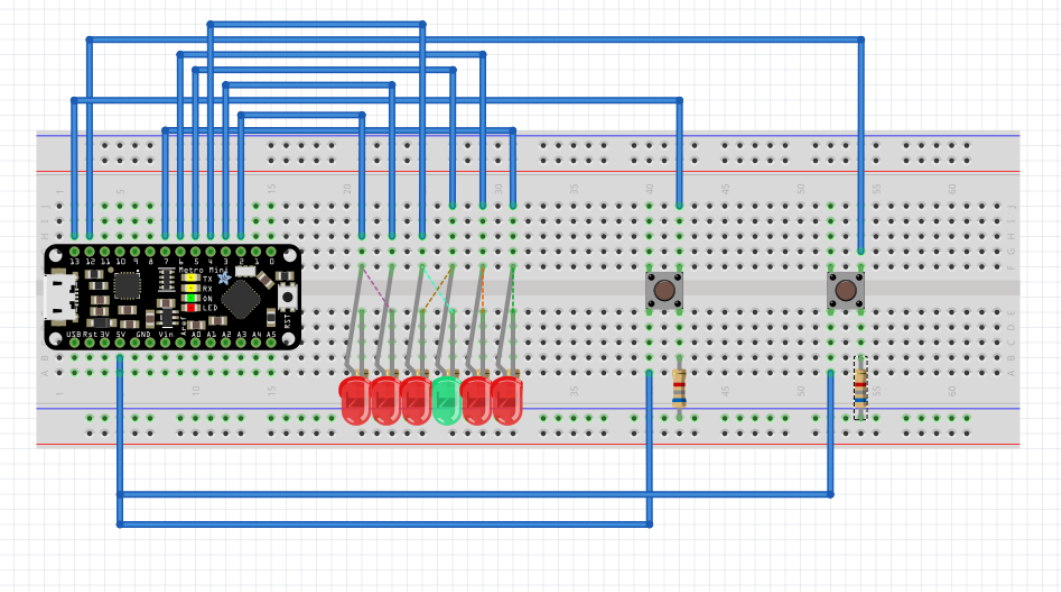

Figure 1: Hardware Implementation

Some tricky parts of the wiring process are the wiring of the resistors and the buttons.In terms of the wiring of the resistors, it is important to make sure the resistor is strong enough to protect the ATmega chip from short circuiting. The best policy here is to use at least a 220Ω resistor, since this will restrict the current to a point where the ATmega chip can handle it. Finally, in terms of the buttons, in order to make sure the button is actually working, connect a light bulb to one of the legs, and connect that bulb to a lane with a resistor going to ground to complete the circuit. This way you will know that the button is actually passing current when it is being pressed.The rest is relatively straight forward.

4 Software Implementation

The Code used for this project is shown below:

```
#include <avr/io.h> // contains uint8_t and register definitions
#include <avr/delay.h>  // contains the delay function
int rest = 300;//The Delay method needs a constant variable
/*
 * Lights are controlled by the following pins
 * PD[2-7] and PB[0]
 * The Green Light is controlled by PB[4]
 *
 * The Start Game button is controlled by PB[5]
 *
 * Skill Stop is controlled by PB[4]
 *
 * Full Stop is controlled by PB[3]
 */
```

```
15 int main(){
16 //This is the main method
17 init();
18 while(true){
19   //Here we want to see if the start game button is pressed
20   bool startGame=PINB& 0b00100000;
21   if(startGame){
22     cycle(100);
23   }
24 }
25 cycle(100);
26
27 return 0;
28 }
29
30
31 //TODO: Make a method that cycles through the lights.
32 int cycle(int cycles){
33   //We will set all the necessary pins to output
34   DDRD=0b11111111;//Setting the lights to output
35   DDRB=0b00000000;
36
37   //This allows us to loop through the
38   //lights one by one in the forward
39   //direction from 0 to 1
40 for(int a=0;a<=cycle;a++){
41   for(int i=2;i<8;i++){
42     //This will go from 0 to 7
43     PORTD=1<<i;
44     int decision=skillStop();
45     if(decision==0){
46       continue;
47     }
48     else if(decision==1){
49       //If the correct button was pressed we want to show that,
50       //decrement the rest time, and continue
51       setRest(200);
52       _delay_ms(2000);
53       continue;
54     }
55     else if (decision==2){
56       _delay_ms(2000);//If the wrong light was chosen, then we want
57       //to stop the
58       return;
59     }
```

```
60      else if(cycles<=0){
61        return 3;
62      }
63    }
64
65    //This will loop in the reverse direction
66    for(int i=7;i>=2;i--){
67      // this will go from 7 to 0
68
69      PORTD=1<<i;
70       int decision=skillStop();
71      if(decision==0){
72        continue;
73      }
74      else if(decision==1){
75        //If the correct button was pressed we want to show that,
76        //decrement the rest time, and continue
77        setRest(80);
78        _delay_ms(2000);
79        continue;
80      }
81      else if (decision==2){
82        _delay_ms(2000);
83        //If the wrong light was chosen,
84        //then we want to stop the
85        return;
86      }
87      else if(cycles<=0){
88        return 3;
89      }
90    }
91
92  }
93  }
94  //TODO skill stop stops the light right at the place
95  int skillStop(){
96    //This method will stop the light in its spot
97    //we want to have a while loop here to make sure the board
98    //registers inputs at any point
99    bool pressed=false;//this will see if the button is pressed
100   int x=0;
101   while(!pressed && x<=rest){
102     _delay_ms(1);
103     pressed=PINB & 0b00010000;
104     x++;
```

```
105     }
106     //This loop will break after the button
107     //has been pressed or if the rest period has expired
108     if(x==0){
109       //We will handle if the rest period has expired first
110       //In this case we want to break
111       return 0;
112     }
113     else if(pressed &&(PORTD & 0b00010000)){
114       //Here we want to handle if the button has been pressed on the
115       //correct spot
116       return 1;//We will handle this in the cycle method
117     }
118     else if(pressed &&!(PORTD & 0b00010000)){
119       //If the button was pressed, but it was the wrong light
120       return 2;
121     }
122     }
123     else{
124       return 0;
125     }
126 }
127
128
129 //Rest Decrement
130
131 void setRest(int decrement){
132   //int decrement will set the decrement factor
133
134   if(rest<=20){
135     rest=20;
136   }
137   else{
138     rest-=20;
139   }
140 }
```

5 Testing and Possible Further Improvements

5.1 Testing

In order to test my little arcade game, I got many of my friends to play the game and try to win. This for the most part worked out well, but I did have some issues during this process

because there were some logical bugs in my code that caused the game not to perform correctly. These were relatively easy fixes once they were observed.

5.2 Further Improvements

One of the things I could not implement was a slow stop button. The idea with this button was to allow the light to slow to a stop instead of stop immediately on the bulb. I wanted to include something like this because the arcade machines usually have a feature like this that the player can use.

6 Conclusion

Overall, this project was a great way to understand and interact with the GPIO, physically and through the code, because it allowed me to create a cool project that people can actually play!