Candyland 2.0

Kaitlyn Mangano
Tolga Kaya
ENGR 200 - Computational Methods in Engineering

Abstract

In this project, I utilized an Engineering computation software to create a revised version of the game Candyland by making modifications to it in order to improve the design. For instance, I cut out holes and added photoresistors to the bottom of the game board. Each player receives a push button and an LED at the start of the game. Each time a player passes over a photoresistor the LED's brightness will increase by 20%. By the end of the game, the winners LED will be at a brightness level of 100%.

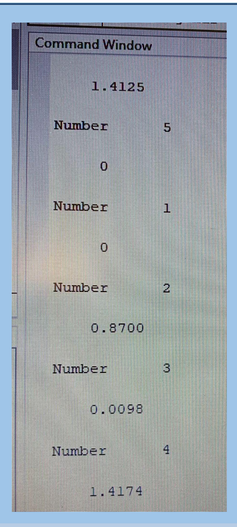
```
clear
a = arduino;
p1 = "A3";
p2 = "A5";
p3 = "A2";
p4 = "A0";
p5 = "A4";
ledY = "D5";
ledR = "D11";
ledG = "D10";
ledB = "D9";
buttonY = "D3";
buttonR = "D12";
buttonG = "D13";
buttonB = "D6";
Start = 1;


leds = [ledY, ledR, ledG, ledB];
buttons = [buttonY, buttonR, buttonG, buttonB];
photoresistor = [p1, p2, p3, p4, p5];
tollerance = [0,0,0,0,0];                % matrix
brightness = [0,0,0,0];
percent = .20;
```

```
for iii = 1:5
    values=readVoltage(a, photoresistor(iii));
    tollerance(iii)=values*percent;
end
while (Start == 1)
    for i = 1:4
        buttonsState = readDigitalPin(a, buttons(i));
        if (buttonsState == 1)
            for ii = 1:5 % read the photoresistor and output the value
                p1Value = readVoltage(a, photoresistor(ii));
                fprintf('Number ');
                disp(ii);
                disp(p1Value);
                if (p1Value < tollerance(ii))
                    writePWMDutyCycle(a, leds(i), brightness(i));
                    if (brightness(i) == 1)
                        brightness(i) = 1;
                    else
                        brightness(i) = brightness(i) + .20;
                    end
                end
            end
        end
    end
end
```

Code

I wrote my code for the project in MATLAB. In the beginning, I declared variables. I specified what analog pins each photoresistor was using. Also, I specified each digital pin that the LEDs and push buttons were using. Next, I created 3 matrices, each containing the variable names declared above. Following this, I had two more matrices. One contained 5 zeros set to tolerance representing the ambient light. Another contained 4 zeros set to brightness representing the LED start off (whether the LED is on or off). Lastly, I created another variable setting percent to .20.

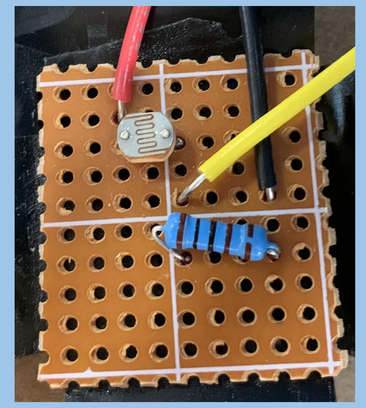
Next, I had a for loop that reads the ambient light and sets the resistor tolerances below that light value. I then have a while loop and inside that a for loop that reads the buttons state. This means whether it is 0 or 1 (is it being pressed or not). If the buttons state is equal to 1 (being pressed), then it will read the values of all the photoresistors and print in the command window of MATLAB what number photoresistor is being read and that value. If the light value is less than the values of the ambient light (a game piece is covering a photoresistor), then increase the brightness of the LED by 20 percent. Once the LEDs brightness gets to 100% stop increasing.



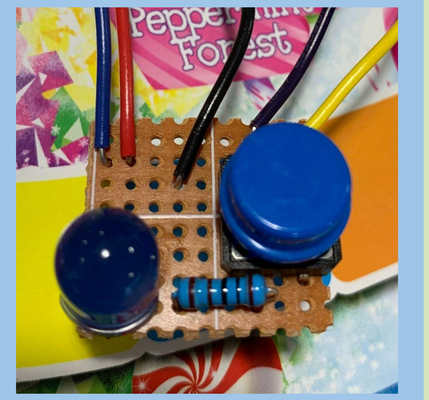
Methods and Materials

The materials used for this project was the Candyland board game, LED's, photoresistors, 1k resistors, Arduino, jumper wires, push button, electrical tape, stripboards, x-acto knife, soldering equipment and a breadboard. The software I used was MATLAB.

First, I cut the game board in half. Next, I used an x-acto knife to cut out 8 stripboards. I made sure to leave enough space for an LED, photoresistor, and resistor. After, I soldered an LED, resistor, and push button onto 4 of the stripboards. I choose to have each LED and button be a different color for each player. Furthermore, I knew it is important to make sure that the 1k resistor is connected to the button. With the 4 stripboards left, I soldered a photoresistor and a 1k resistor onto it, connecting the two in the same row. Next, I cut 35 jumper wires to the proper length of about 12-14 inches so that there is enough room for each player to hold their piece. Then, I soldered a wire onto the power side of the LED and connected the other end to a digital pin on the Arduino, another wire from the ground side of LED to ground on the breadboard, and another wire from the resistor to ground. For the push button, I soldered a wire to one row (including 2 prongs) and connected that to a digital pin, and another wire from the other two prongs to power. I repeat these steps 3 more times. After, I soldered a wire from one side of the photoresistor to power, solder another wire to the other side of the photoresistor and connected that to ground. Lastly, I soldered a wire to the side of the resistor not connected to the photoresistor and connected that to an analog pin. I cut out 5 holes on the game board and used electrical tape to adhere the stripboard with the photoresistor on it to the back of the board so that I could see the photoresistor through the holes.

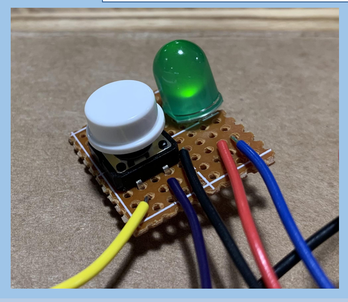


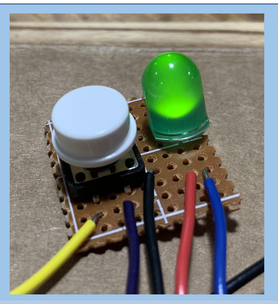
The hardware underneath the board is to the left and the hardware that the players hold is below.



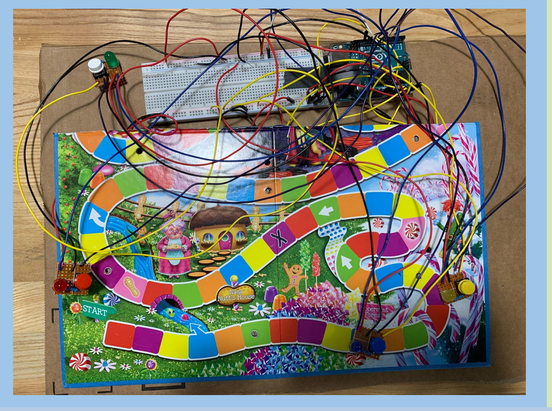
Conclusions

In closing, this project gave me the opportunity to reinforce my MATLAB skills such as practice using loops, matrices, and implementing variables. To further strengthen my project, I will collect data from my game. My plan is to output a graph in MATLAB of each players LED brightness at the end of the game. This would be another way to determine the winner of the game.





The LED on the left is less bright than the LED on the right. The brighter LED has won the game!



Contact

Kaitlyn Mangano
manganok@mail.sacredheart.edu

MATLAB®

Sacred Heart UNIVERSITY | ENGINEERING