Dominic DiPiero

ECET 411

Final Project report

import RPi.GPIO as GPIO  #initializing the GPIO library  
import time     #initializing the time library  
import random    #initializing the random number library  
#importing libraries allows us to use their functions within the scope of the program, the time

#library allows us to use time.sleep, and the random library allows us to use the random number

#generator that is used to turn the green or red light on randomly and the GPIO allows us

#connect components to the board

GPIO.setwarnings(False)  #ignores warnings during code execution  
GPIO.setmode(GPIO.BCM)   #setting how the board numbers its inputs  
  
GAME = True     #sets the game portion of the code to run at the beginning  
GAMEOVER = False   #sets the game over loop to off at the beginning  
servo = 18   #tells the RPi that the servo is connected to pin 18  
A\_G = 26   #tells the RPi that the button for the green LED for player A is at 26  
A\_R = 13    #tells the RPi that the button for the red LED for player A is at 13  
LED\_R = 25   #red led is connected to pin 25  
LED\_G = 6    #green led is connected to pin 6  
B\_G = 24   #tells the RPi that the button for the green LED for player B is at 24  
B\_R = 23   ##tells the RPi that the button for the red LED for player B is at 23

#the above block of code assigns variables to each of the pins where components are connected

#this allows us to easily call components to do their various functions throughout the program

#rather than calling them by what pin they are attached to

GPIO.setup(A\_G,[GPIO.IN](http://gpio.in/))   #this line and the next 6 formerly initializes the component to the pin  
GPIO.setup(A\_R,[GPIO.IN](http://gpio.in/))    #And establishes them as inputs or outputs  
GPIO.setup(B\_G,[GPIO.IN](http://gpio.in/))  
GPIO.setup(B\_R,[GPIO.IN](http://gpio.in/))  
GPIO.setup(LED\_R, GPIO.OUT)  
GPIO.setup(LED\_G,GPIO.OUT)  
GPIO.setup(servo, GPIO.OUT)

#the above block assigns the pins the variables and sets them as inputs or outputs

pwm = GPIO.PWM(servo,50)   #sets the frequency of the servo  
pwm.start(0)    #starts the servo  
x = 70  #starting angle  
wins = 0 #when wins gets to +/- 2, GAME becomes false, GAMEOVER becomes true  
def SetAngle(angle):   #set angle function  
    duty = angle/18+2   #  duty is the angle the servo is to move to  
    GPIO.output(servo,True)   #turns on the servo  
    pwm.ChangeDutyCycle(duty)  #changes the angle of the servo  
    time.sleep(1)    #rest  
    GPIO.output(servo,False)   #turns off the servo  
    pwm.ChangeDutyCycle(0)    #sets duty variable to 0

#the above block is all the things that have to do with getting the servo to work, converting

#angles to points on the servo, as well as the SetAngle function that will be used to change the

#angle of the servo during the game  
  
GPIO.output(LED\_R,False)  #this line and the next make sure the LEDs are off at the start of the program  
GPIO.output(LED\_G,False)  
SetAngle(70)   #sets servo to the middle of both players using set angle function

#the above block gets the components ready to start the game, makes sure the LEDs are off and

#the servo is centered

#SetAngle(25)  #the following lines are commented out and were used to tinker with the angle of  
#time.sleep(2)  #the servo in order to get it to look like angles of 45, 90, 135, 180 respectively  
#SetAngle(70)  
#time.sleep(2)  
#SetAngle(125)  
#time.sleep(2)  
#SetAngle(180)  
  
#0 = pressed, 1 = unpressed  #reminder for me that 0 means a button is pressed

while GAME == True:  #loop will run as long as the game is active  
    time.sleep(4)   #time between random number generation  
    r = random.choice([0,1])   #produces random number, either 0 or 1, stores in variable r  
    print(r)   #puts the random number in the shell

#this block is the random number generator, it waits 4 seconds at the beginning, and after each

#round and then generates either 1 or 0, 0 represents the green LED, 1 representing red. Random

#choice function works by putting the 2 ends of the range you want the random number to be in

# and then assigning it to a variable

    while r == 0:  #green LED is on  
        GPIO.output(LED\_G,True)  #turns green led on  
        GPIO.output(LED\_R,False)  #turns red led off

#this is the conditions when the random number generated is 0, the green LED turns on and one

#of the following two blocks of code are ran, depending on which player presses the correct

#button first  
         
        if GPIO.input(A\_G) == 0:   #start of if statement for if player A presses their button first  
            while True:  
                GPIO.output(LED\_G,False)  #turns green led off  
                x = x- 45  #preparing to change the angle of servo moving it 45 deg towards player A  
                SetAngle(x)  #moves servo  
                print("Player A wins this round")   #prints this statement in shell  
                wins = wins - 1   #adds 1 win in favor of player A  
                print("the score is: ",wins)  #score update  
                break  #exits the while true loop  
            break  #exits the if statement loop, GAME loop continues  
         
        if GPIO.input(B\_G) == 0:   #start of if statement for if player B presses their button first  
            while True:  #while in this loop, do the following  
                GPIO.output(LED\_G,False)   #turn green led off  
                x = x + 45 #preparing to change the angle of servo moving it 45 deg towards player B  
                SetAngle(x)  #moves servo  
                print("Player B wins this round")  #prints that statement in shell  
                wins = wins + 1  #adds one win in favor of player B  
                print("the score is: ",wins)   #score update  
                break  #exits while true loop  
            break  #exits if statement, GAME loop continues

#the two blocks of code are the basically the same loops, the only difference is one is for if

#player A wins, and the other for Player B winning. When either player presses their button

#first, the light goes off, and the servo is moved 45 degrees towards them by updating the angle

#variable x and then running the SetAngle function to change its position to the new angle

#at the end of the loops, a win is either added if player B wins, or subtracted if player A wins

# I will further explain the wins variable in the GAMEOVER loop of the code. After these

#commands have been executed, the loops terminate, and The GAME loop starts again from the

#top, starting with the 4 second rest before another number is generated

    while r == 1:   #Red LED is on  
        GPIO.output(LED\_R,True)  #turns red led on  
        GPIO.output(LED\_G, False)  #ensures that green led off

#this is the loop that runs while the red LED is on, which means that the random number

#generator chose 1

        if GPIO.input(A\_R) == 0:  #if player A presses the correct button first, execute the following loop  
            while True:   #the loop to be executed  
                GPIO.output(LED\_R,False)  #turn red led off  
                print("Player A wins this round")  #states that player A wins this round  
                x = x - 45  #prepares to move servo 45 degrees towards player A  
                SetAngle(x)  #moves servo  
                wins = wins - 1  #adds one win in favor of player A  
                print("the score is: ",wins)  #score update  
                break  #exits the while true loop  
            break  #exits the if statement, GAME loop continues  
         
        if GPIO.input(B\_R) == 0:   #this is the loop for if player B presses the correct button first  
            while True:  #this is the loop to be preformed  
                GPIO.output(LED\_R,False)  #turns red led off  
                print("Player B wins this round")  #states that player B wins this round  
                x = x + 45  #prepares to move servo 45 degrees towards player B  
                SetAngle(x)  #moves servo  
                wins = wins + 1  #adds one win in favor of player B  
                print("the score is: ",wins)  #score update  
                break  #breaks out of while true loop  
            break   #breaks out of if statement, GAME loop continues

#the two blocks of code are the basically the same loops, the only difference is one is for if

#player A wins, and the other for Player B winning. When either player presses their button

#first, the light goes off, and the servo is moved 45 degrees towards them by updating the angle

#variable x and then running the SetAngle function to change its position to the new angle

#at the end of the loops, a win is either added if player B wins, or subtracted if player A wins

#again, after these commands are executed and the score is updated and displayed, the loops

#end and the GAME loop starts again

         
    if wins == 2 or wins == -2 :  #conditions for the end of the game,  
        GAMEOVER = True  #turns on GAMEOVER loop  
        GAME = False   #turns off GAME loop

#the above three lines involve the wins variable. if you notice in the code, when player A wins

#the wins variable is decreased by 1, and if B wins it is increased. in order to win the servo

#must servo face the player (0 for player A, and 180). This means the winning player needs to

#have 2 wins MORE than the other player not just two wins. So, by adding and subtracting from

#the same variable it ensures that when ‘wins’ is at +/- 2, the game loop is stopped, and the

#GAMEOVER loop is started   
  
while GAMEOVER == True:  #this is the game over loop  
    if wins == 2:  #conditions for player B to win the game  
        print("PLAYER B WINS!!!")  #tells player B they are victorious  
        GPIO.output(LED\_G, True)  #turns on green led  
        GPIO.output(LED\_R, True)  #turns on red led  
        time.sleep(0.5)   #keeps LEDs on for half a second  
        GPIO.output(LED\_G, False)  #turns on green led  
        GPIO.output(LED\_R, False)  #turns on red led  
        time.sleep(0.5)   #keeps LEDs off for a half of a second  
        #loop continues until user stops program

#the above block is the block of code that runs when player B has won the game. It prints

# “PLAYER B WINS!!!” and the LEDs begin to flash on and off every .5 seconds

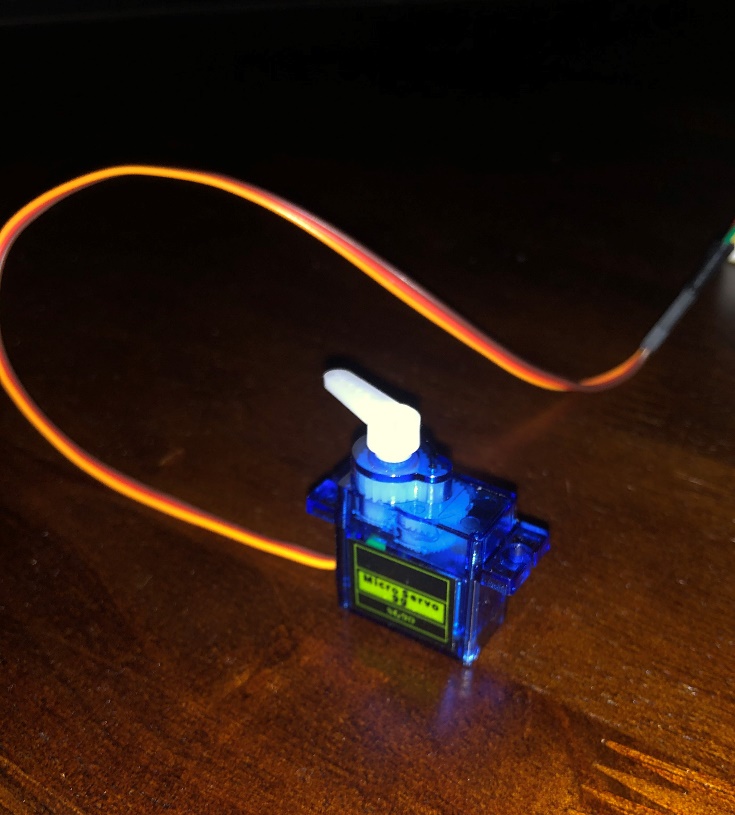
    if wins == -2:   #conditions for player A to win the game  
        print("PLAYER A WINS!!!")  #alerts player A that they are triumphant  
        GPIO.output(LED\_G, True)  #turns on green led  
        GPIO.output(LED\_R, True)  #turns on red led  
        time.sleep(0.5)   #keeps LEDs on for half a second  
        GPIO.output(LED\_G, False)  #turns on green led  
        GPIO.output(LED\_R, False)  #turns on red led  
        time.sleep(0.5)   #keeps LEDs off for a half of a second  
        #loop continues until user stops program

#the above block is the block of code that runs when player A has won the game. It prints

# “PLAYER A WINS!!!” and the LEDs begin to flash on and off every .5 seconds

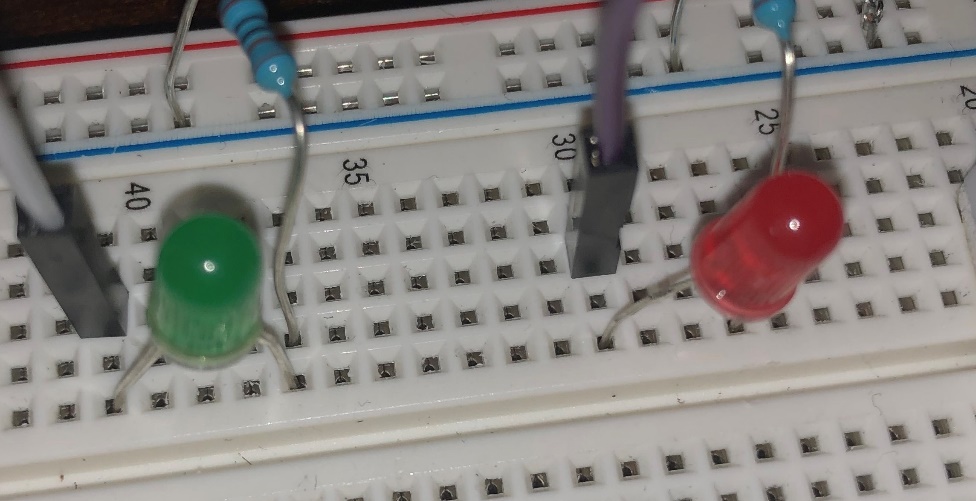
**Pictures of components:**

**SERVO**



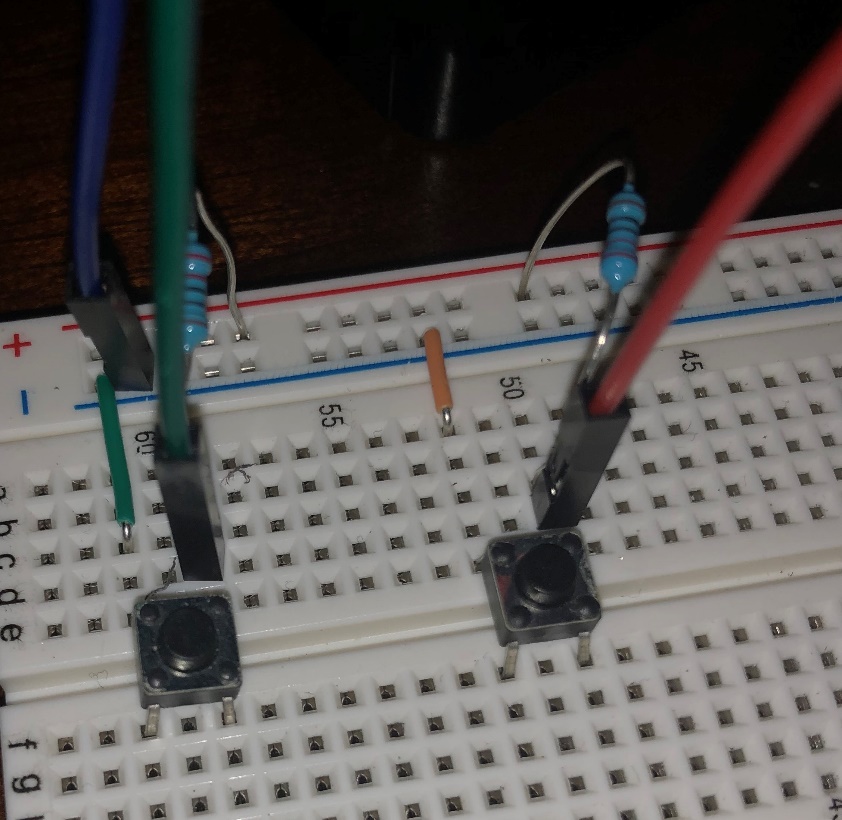
A servo is a small motor that has a range from 0 to 180 degrees, it has 3 pins, a data pin, that reads data such as what angle to turn to, and a ground and a voltage pin.

**LED**



LED (light emitting diode) is a light that serves as the indicator for what button to press during the game. one prong goes to ground, the other goes to the RPi board to receive voltage when the pin is set to high

**Buttons**



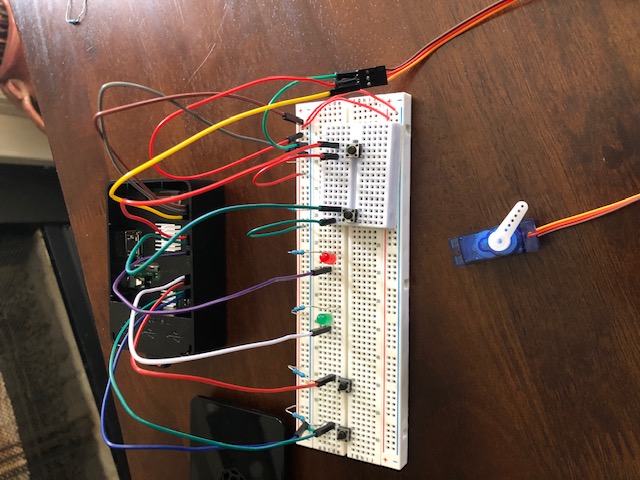
Buttons are used as an output in our game, they send a pulse to the pin while they are pressed, for they are normally open buttons the two prongs on one side of the button are the ones used, one connects to a resistor that connects to the voltage source while also connecting to the board to send data to the RPi, and the other goes to ground

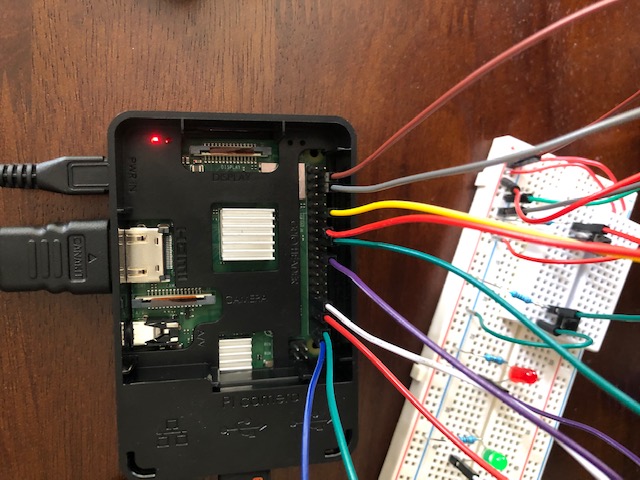
**Resistors**



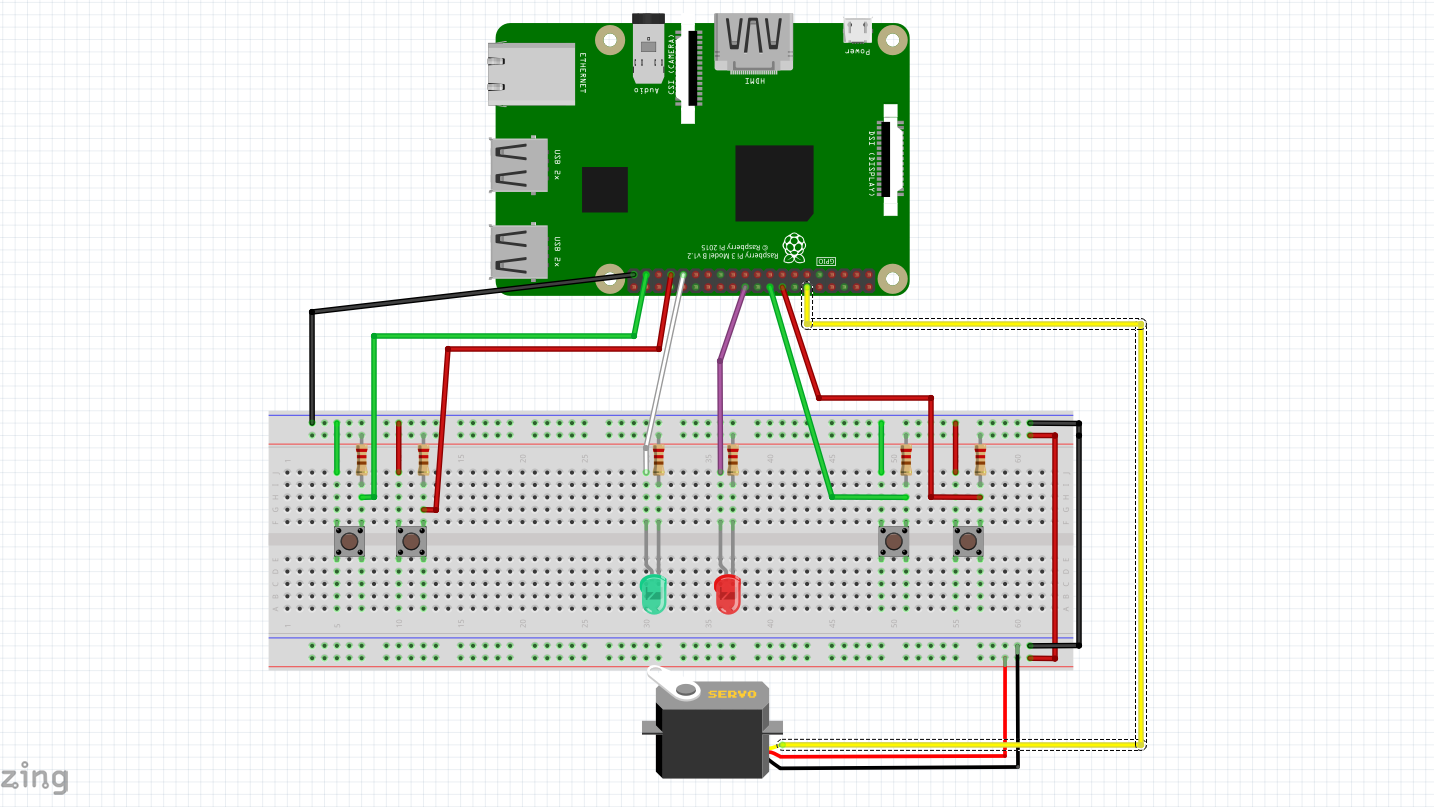
Resistors stop components from receiving too much current and burning out

**Circuit photos**





**Digital Circuit Diagram**



**Connection Schematic**

