```
/*
M&M Dispenser
ENME 351 Final Project
Author: Ian Michel-Tyler
*/

#include <Servo.h>

Servo red_servo; // Create servo objects for three servos
Servo green_servo;
Servo blue_servo;

int U = 0; // Define global variables
int L = 0;
int D = 0;
int R = 0;
int color; // variable to control which motor actuates

// Initial servo angles found by trial and error
int b_angle_init = 84;
int g_angle_init = 89;
int r_angle_init = 79;

// Interval between actuating to desired angle and initial angle
int dur = 50;

// Define pin for which control script runs
int controlPin = 12;

void setup() {
  Serial.begin(9600);

  // Attach servos to pins
  red_servo.attach(8);
  green_servo.attach(9);
  blue_servo.attach(10);

  // Set servo to starting position
  red_servo.write(r_angle_init);
  green_servo.write(g_angle_init);
  blue_servo.write(b_angle_init);

  // Set button pins and control pin to input
  pinMode(2,INPUT);
  pinMode(3,INPUT);
  pinMode(4,INPUT);
  pinMode(5,INPUT);
  pinMode(controlPin,INPUT);

  // Establish handshake with processing over serial
  // Code adapted from Arduino Serial_Call_Response example at https://www.arduino.cc/en/Tutorial/SerialCallResponse
  establishContact();
}
```

```
void loop() {

  // If the switch is HIGH, the input to the dispenser is the processing game
  if (digitalRead(controlPin) == HIGH){
    // If bytes are available from the serial stream, do:
    if (Serial.available() > 0){
      // Accept information before sending
      color = Serial.read();
      // Read button pins and write to serial
      U = digitalRead(2);
      L = digitalRead(3);
      D = digitalRead(4);
      R = digitalRead(5);
      Serial.write(U);Serial.write(L);Serial.write(D);Serial.write(R);
      // Delay ample time to allow processing to execute
      delay(50);
  }
  // If the code received is red, actuate red servo to dispense red M&M
  if (color == 1){
      red_servo.write(r_angle_init+15);
      delay(dur);
      red_servo.write(r_angle_init);
  }
  // Same but green
  else if(color == 2){
      green_servo.write(g_angle_init+15);
      delay(dur);
      green_servo.write(g_angle_init);
  }
  // Same but blue
  else if(color == 3){
      blue_servo.write(b_angle_init+15);
      delay(dur);
      blue_servo.write(b_angle_init);
  }
}
  // If switch is LOW, input becomes python script running rudimentary color classifier
  else if(digitalRead(controlPin) == LOW){
    if (Serial.available() > 0) {
      color = Serial.read();
      // For some reason I decided to use characters for this one
      if(color == 'R'){
        red_servo.write(r_angle_init+15);
        delay(dur);
        red_servo.write(r_angle_init);
      }
      else if(color == 'G'){
        green_servo.write(g_angle_init+15);
        delay(dur);
        green_servo.write(g_angle_init);
      }
      else if(color == 'B'){
        blue_servo.write(b_angle_init+15);
        delay(dur);
        blue_servo.write(b_angle_init);
      }
      else{
        delay(50);
      }
    }
    else{
      delay(50);
    }
    delay(50);
  }
}
```

```
// Code adapted from Arduino Serial_Call_Response example at https://www.arduino.cc/en/Tutorial/SerialCallResponse
void establishContact() {
  while (Serial.available() <= 0) {
    Serial.print('A');   // send a capital A until reciprocated
    delay(300);
}
}
```

```
/*
M&M Dispenser
ENME 351 Final Project
Author: Ian Michel-Tyler
*/

import processing.serial.*;
Serial myPort;

int dot_size = 20;
int x_min = 10;
int y_min = 10;
int x_max = 790;
int y_max = 790;
int x_now = (10 + 20*floor(random(39)));
int y_now = (10 + 20*floor(random(39)));
int prize_x = 20*floor(random(40));
int prize_y = 20*floor(random(40));
int prize_id = ceil(random(3));
int score = 0;
int px_adjust;
int py_adjust;
int x_prev;
int y_prev;
boolean firstContact = false;
int serialCount = 0;
int serialInArray[] = new int[4];
boolean reset = false;

// Function to reset board after prize is captured
void reset_board(){

  // Reset the reset variable
  reset = false;
  // Flash screen
  background(225);
  // Redraw player
  fill(0,225,225);
  ellipse(x_now,y_now,dot_size,dot_size);

  // Create random location and color for prize
  prize_id = ceil(random(3));
  prize_x = 20*floor(random(40));
  prize_y = 20*floor(random(40));

  // Draw prize
  if (prize_id == 1){
    fill(255,0,0);
    rect(prize_x,prize_y,dot_size,dot_size);
  }
  else if (prize_id == 2){
    fill(0,255,0);
    rect(prize_x,prize_y,dot_size,dot_size);
  }
  else if (prize_id == 3){
    fill(0,0,255);
    rect(prize_x,prize_y,dot_size,dot_size);
  }
```

```
  // Because the rectangle object uses top left corner location and ellipse uses center location as parameters, this variable "translates" location
  py_adjust = prize_y+10;
  px_adjust = prize_x+10;
}

void setup(){

  // Initializes background and size
  background(225);
  size(800,800);
  stroke(225);

  // Draws cyan circle at random player starting point
  fill(0,225,225);
  ellipse(x_now,y_now,dot_size,dot_size);

  // Initialize prize location and color
  if (prize_id == 1){
    fill(255,0,0);
    rect(prize_x,prize_y,dot_size,dot_size);
  }
  else if (prize_id == 2){
    fill(0,255,0);
    rect(prize_x,prize_y,dot_size,dot_size);
  }
  else if (prize_id == 3){
    fill(0,0,255);
    rect(prize_x,prize_y,dot_size,dot_size);
  }

  py_adjust = prize_y+10;
  px_adjust = prize_x+10;

  // Create serial object using COM3 and clear buffer
  printArray(Serial.list());
  myPort = new Serial(this, Serial.list()[0], 9600);
  myPort.clear();

}
```

```
void draw() {

  // Display score in top right corner
  fill(0);
  textSize(20);
  text("Score = " + str(score),650,50);

  // Cover previous circle
  fill(225);
  stroke(225);
  rect(x_prev,y_prev,dot_size,dot_size);

  // Draw circle at new player location
  fill(0,225,225);
  ellipse(x_now,y_now,dot_size,dot_size);

  // If board needs reseting
  if (reset == true){
    reset_board(); // Calls reset function to create new prize object
  }
}

// Establish handshake with arduino over serial
// Code adapted from Arduino Serial_Call_Response example at https://www.arduino.cc/en/Tutorial/SerialCallResponse
```

```
// Establish handshake with arduino over serial
// Code adapted from Arduino Serial_Call_Response example at https://www.arduino.cc/en/Tutorial/SerialCallResponse
void serialEvent(Serial myPort) {

    // read incoming byte from buffer
    int inByte = myPort.read();

    // If the byte is the first 'A' read from the arduino then clear buffer and send confirmation receipt to serial. Processing starts listening.
    if (firstContact == false) {
      if (inByte == 'A') {
        myPort.clear();
        firstContact = true;
        myPort.write('A');
      }
    }
    // After handshake established, incoming bytes representing game controls are stored in an array.
    // When all four are received they are configured to the game inputs.
    else {
      serialInArray[serialCount] = inByte;
      serialCount++;

      if (serialCount > 3){
        int U = serialInArray[0];
        int L = serialInArray[1];
        int D = serialInArray[2];
        int R = serialInArray[3];

        // Store previous coordinates for later use. Probably a smarter way to do the later computations without this variable.
        x_prev = x_now-10;
        y_prev = y_now-10;

        // Compute new coordinates of player after move
        x_now = x_now + 20 * (R-L);
        y_now = y_now + 20 * (D-U);

        // Boundary conditions
        if (x_now < x_min){
          x_now = x_max;
        }

        if (x_now > x_max){
          x_now = x_min;
        }
```

```
    if (y_now < y_min){
      y_now = y_max;
    }

    if (y_now > y_max){
      y_now = y_min;
    }

    // If the player gets the prize ---> send prize color code to arduino for dispensing
    if ((x_now == px_adjust) && (y_now == py_adjust)){
      // Cast variable from int to byte for serial transfer
      myPort.write(byte(prize_id));
      // Make sure new prize is generated
      reset = true;
      // Keep track of score
      score++;
    }

    // Otherwise ready for next move
    else{
      myPort.write('A');
    }
    // Prepare for new set of four bytes
    serialCount = 0;
  }

 }
}
```

```python
import serial
import numpy as np
import cv2
import struct
import time

cap = cv2.VideoCapture(0) # Creates camera object using webcam at port 0 using opencv library

Lower = np.array([[100, 60, 15], [40, 80, 50], [50, 50, 100]]) # BGR x BGR Lower Values
Upper = np.array([[255, 120, 40], [90, 255, 90], [110, 80, 255]]) # BGR x BGR Upper Values

ser = serial.Serial('COM3', 9600, timeout = 1) # creates serial object at COM3 using pyserial library

while True:
    val, image = cap.read() # captures image for iteration of computation

    color = np.ceil(cv2.mean(image))
    color = color[0:3]

    if (color[0] < Upper[0,0] and color[0] > Lower[0,0] and
        color[1] < Upper[0,1] and color[1] > Lower[0,1] and
        color[2] < Upper[0,2] and color[2] > Lower[0,2]):

        serial_color = 'B'                                  # If color is in RGB range defined for blue then color code ='B'

    elif (color[0] < Upper[1,0] and color[0] > Lower[1,0] and
        color[1] < Upper[1,1] and color[1] > Lower[1,1] and
        color[2] < Upper[1,2] and color[2] > Lower[1,2]):

        serial_color = 'G'                                  # If color is in RGB range defined for green then color code ='G'

    elif (color[0] < Upper[2,0] and color[0] > Lower[2,0] and
        color[1] < Upper[2,1] and color[1] > Lower[2,1] and
        color[2] < Upper[2,2] and color[2] > Lower[2,2]):

        serial_color = 'R'                                  # If color is in RGB range defined for red then color code ='R'

    else:
        serial_color = 'N'                                  # Writes unused letter for colors out of defined ranges.


    print(serial_color)
    print(color)
    ser.write(serial_color.encode()) # Writes color code cast as byte to serial port
    time.sleep(5) # delay 5 seconds

    if cv2.waitKey(1) == 27: # esc will terminate program and unlink serial port and camera instance
        break
        ser.close()
        cv2.destroyAllWindows() # Doesn't do anything when running from jupyter
```