COLOR BASED PRODUCT SORTING

A MINI PROJECT REPORT IN
ET3491-EMBEDDED SYSTEMS AND IOT DESIGN

B.E Electronics And Communication Engineering

Submitted by
Vishnakumar M
3rd Year-6th Semester
820421106062

ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE
Kovilvenni-614 403, Thiruvarur District.

Team Members:
820421106062-Vishnakumar.M
820421106063-Viswa.S
820421106065-Yogesh.B

TABLE OF CONTENT:

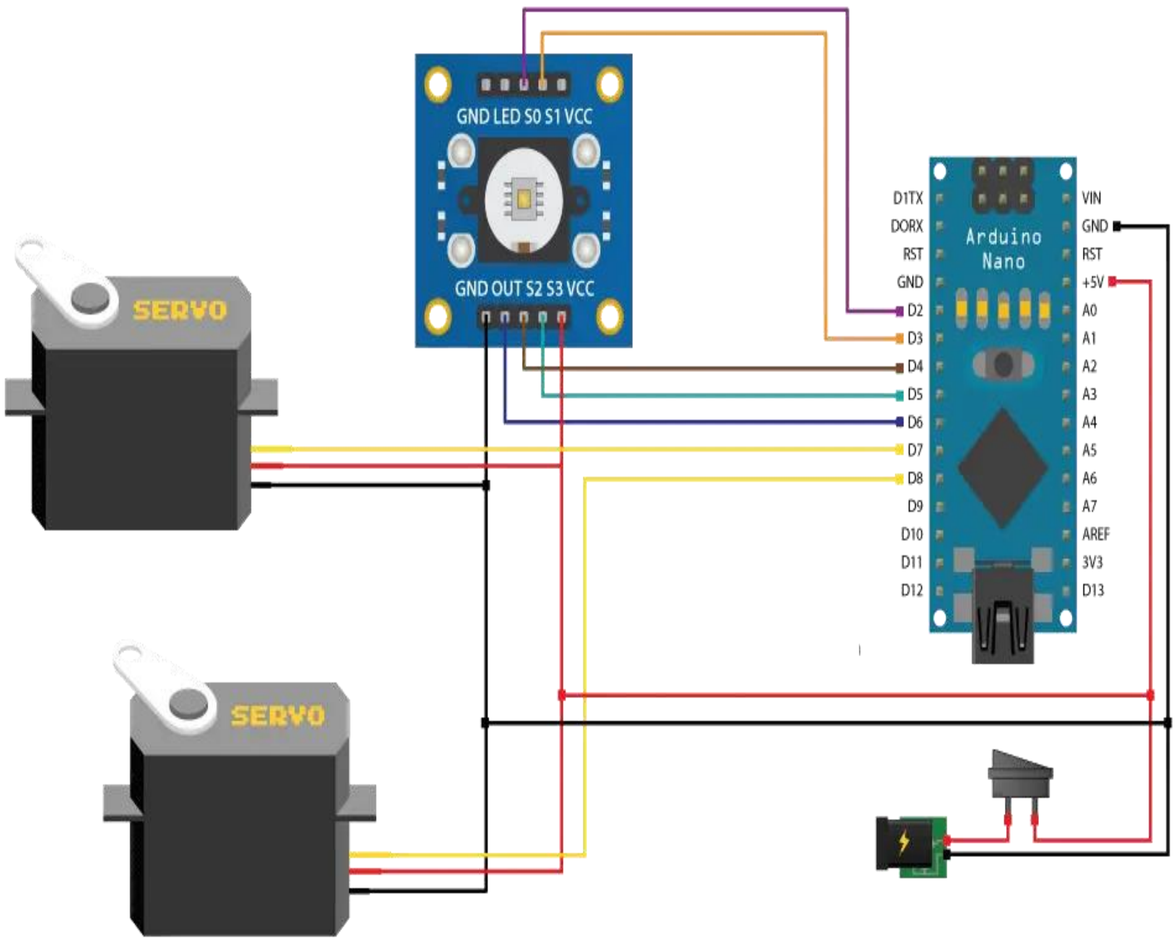
OBJECTIVE:

The objective of this project is to develop a robust and efficient color-based product sorting system using IoT technology. The system will utilize advanced color sensing techniques to accurately identify and classify products based on their color attributes. By integrating IoT devices such as sensors, microcontrollers, and communication modules, the project aims to achieve real-time monitoring, data processing, and decision-making capabilities. The ultimate goal is to enhance productivity and accuracy in product sorting processes, leading to improved operational efficiency and customer satisfaction.

INTRODUCTION:

A color sorting is simply to sort the things according to their color. It can be easily done by seeing it but when there are too many things to be sorted and it is a repetitive task then automatic color sorting machines are very useful. These machines have color sensor to sense the color of any objects and after detecting the color servo motor grab the thing and put it into respective box. They can be used in different application areas where color identification, color distinction and color sorting is important. Some of the application areas include Agriculture Industry (Grain Sorting on the basis of color), Food Industry, Diamond and Mining Industry, Recycling etc. The applications are not limited to this and can be further applied to different industries.

CIRCUIT DIAGRAM:



PROCEDURE:

Here are the steps involved in the working principle of the color sorter project using Arduino, TCS3200 color sensor, and two hobbyist servo motors:

1. Initial Setup Colored skittles are placed in the charger, ready to be sorted.

2. Skittle Placement: The top servo motor positions the platform to receive a skittle from the charger.

3. Color Detection: The top servo motor rotates, bringing the skittle to the TCS3200 color sensor for color detection.

4. Color Recognition: The color sensor detects the color of the skittle, determining its category.

5. Sorting Process: Based on the color detected, the bottom servo motor rotates to the designated position for that color.

6. Final Placement: The top servo motor rotates again, guiding the skittle to drop into the appropriate guide rail or container based on its color category

COMPONENTS & IT'S DESCRIPTION:

1.Color sensor: TCS230 TCS3200 Color

White light is a mixture of three basic colors known as primary colors. They are red, blue and green. These colors have different wavelengths. Combinations of these colors at different proportions create different types of colors. When the white light falls on any surface, some of the wavelengths of the light are absorbed by the surface while some are reflected back based on the properties of the surface material. Colour of the material is detected when these reflected wavelengths fall on the human eye. A material reflecting wavelengths of red light appears as red. The component used to detect colors is the Color sensor. A color sensor detects the color of the material. This sensor can categorize the color as red, blue or green. These sensors are also equipped with filters to reject the unwanted IR light and UV light.

To detect the color of material three main types of equipment are required. A light source to illuminate the material surface, a surface whose color has to be detected and the the receivers which can measure the reflected wavelengths.

We use **TC230/TCS3200 color sensor** in this project – shown in the figure below .

The TCS3200 color sensor can detect a wide variety of colors based on their wavelength. This sensor is specially useful for color recognition projects such as color matching, color sorting, test strip reading and much more. It uses a TAOS TCS3200 RGB sensor chip to detect color. It also contains four white LEDs that light up the object in front of it.

TCS230 TCS3200 COLOR SENSOR

Specifications:

Here's the sensor specifications:

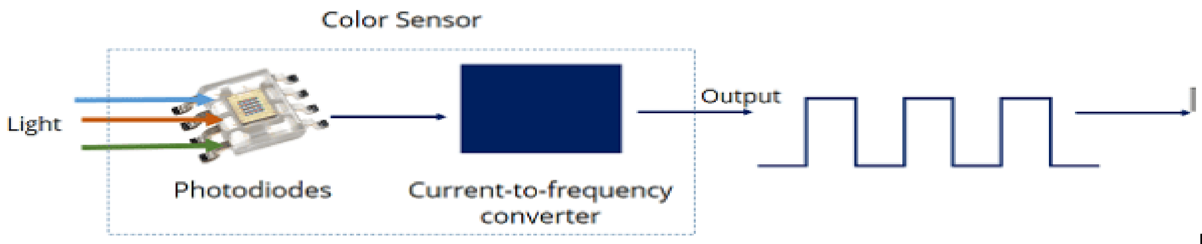
- Power: 2.7V to 5.5V
- Size: 28.4 x 28.4mm (1.12 x 1.12")
- Interface: digital TTL
- High-resolution conversion of light intensity to frequency
- Programmable color and full-scale output frequency
- Communicates directly to microcontroller

How does the TCS3200 sensor work?
The TCS3200 has an array of photodiodes with 4 different filters. A photodiode is simply a semiconductor device that converts light into current. The sensor has:

- 16 photodiodes with red filter – sensitive to red wavelength
- 16 photodiodes with green filter – sensitive to green wavelength
- 16 photodiodes with blue filter – sensitive to blue wavelength
- 16 photodiodes without filter

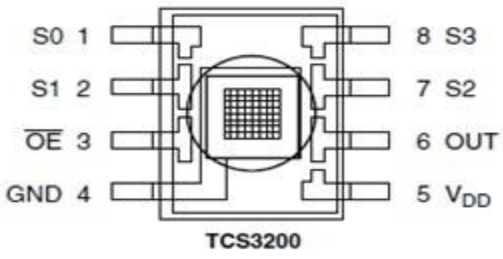
If you take a closer look at the TCS3200 chip you can see the different filters.

By selectively choosing the photodiode filter's readings, you're able to detect the intensity of the different colors. The sensor has a current-to-frequency converter that converts the photodiodes' readings into a square wave with a frequency that is proportional to the light intensity of the chosen color. This frequency is then, read by the Arduino – this is shown in the figure below.



Pinout details

Here's the sensor pinout:



Pin Name	I/O	Description
GND (4)		Power supply ground
OE (3)	I	Enable for output frequency (active low)
OUT (6)	O	Output frequency
S0, S1（1，2）	I	Output frequency scaling selection inputs
S2, S3（7，8）	I	Photodiode type selection inputs
VDD（5）		Voltage supply

Filter selection

To select the color read by the photodiode, you use the control pins S2 and S3. As the photodiodes are connected in parallel, setting the S2 and S3 LOW and HIGH in different combinations allows you to select different photodidodes. Take a look at the table below:

Photodiode type	S2	S3
Red	LOW	LOW
Blue	LOW	HIGH
No filter (clear)	HIGH	LOW
Green	HIGH	HIGH

Frequency scaling

Pins S0 and S1 are used for scaling the output frequency. It can be scaled to the following preset values: 100%, 20% or 2%. Scaling the output frequency is useful to optimize the sensor readings for various frequency counters or microcontrollers. Take a look at the table below:

Output frequency scaling	**S0**	**S1**
Power down	L	L
2%	L	H
20%	H	L
100%	H	H

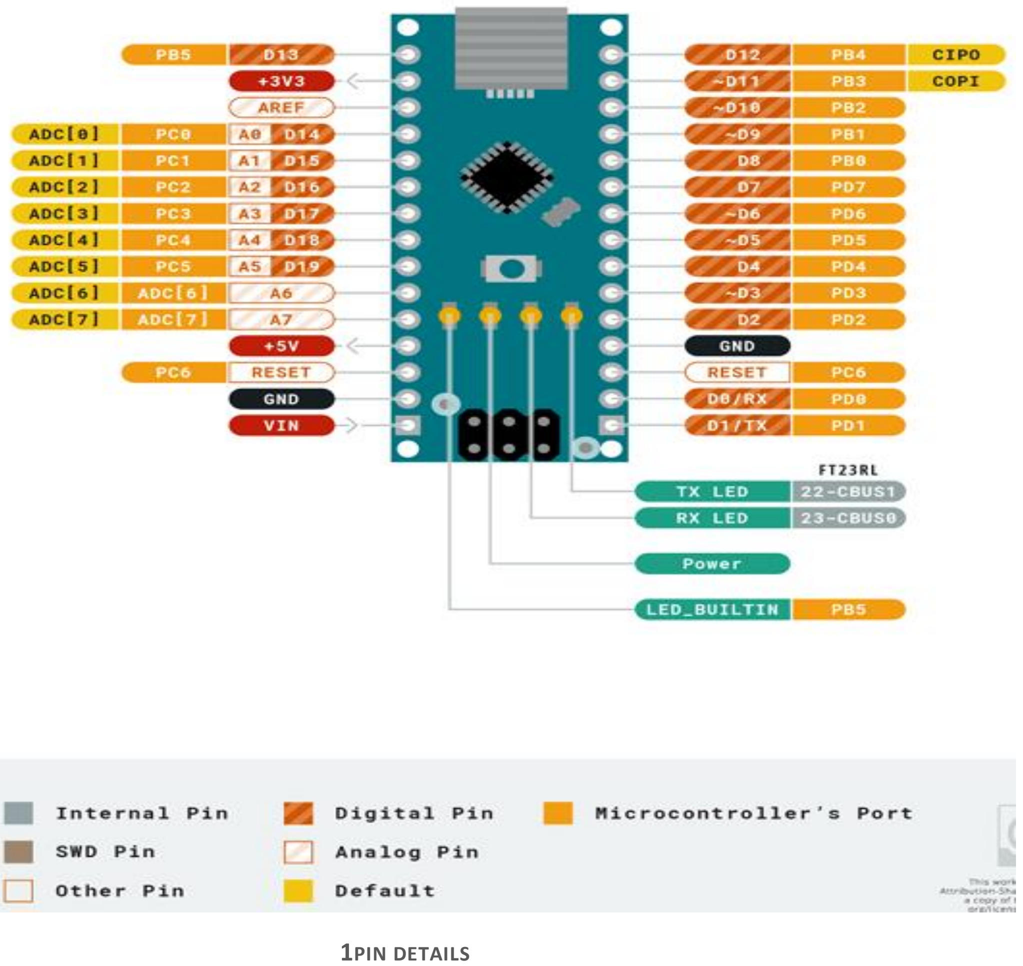
2.ARDUINO NANO :

Arduino Nano is a commonly used prototype development board that was released in 2008. This board has similar features to the Arduino Uno board available in the market, but this design with a small form factor makes it suitable for breadboard usage.Arduino Nano boards are equipped with the same microcontroller available in the Arduino Uno board, Atmega328p. Only missing thing is DC jack is replaced with a USB B connecter. Moreover, in Addition to Arduino Uno, Nano has Additional Analog pins that are not in the Uno.

Specifications:

Microcontroller	Atmega 328p(High-performance low-power 8-bit processor)
Operating	voltage 5 V
Input voltage (VIN)	6-20 V
Power consumption	19 mA
Flash memory	32 KB, of which the bootloader uses 2 KB
Internal SRAM	2 KB
Clock speed	16 Mhz

EEPROM	1 KB
DC current per I/O pin	40 mA (20 mA recommended)
Digital I/O pins	22
PWM outputs	6 (D3, D5, D6, D9, D10, D11)
Analog input pins	8 (ADC 10 bit)
I2C	A4 (SDA), A5 (SCL)
Master/Slave SPI Serial Interface	D10 (SS), D11 (MOSI), D12 (MISO), D13 (SCK)
LED_BUILTIN	D13



1PIN DETAILS

Digital Pins are marked as D2-D13, TX(D0), RX(D1) these pins on the Arduino Nano used for input and output by using the function such as pinMode(), digitalWrite() and digitalRead(). These pins operate at 5V and Provide a maximum current of 40Miliamphere. All the digital and Analog pins are connected with an internal Pull resistor of 50K ohms. By default its disconnected, for those who want to use it can enable it through the program by providing pinMode(PIN, INPUT_PULLUP);

In Arduino Nano, you have Analog pins ranging from A0 to A7, in which A6 and A7 can only be used for Analog inputs. Pins and its function will be explained in the below table

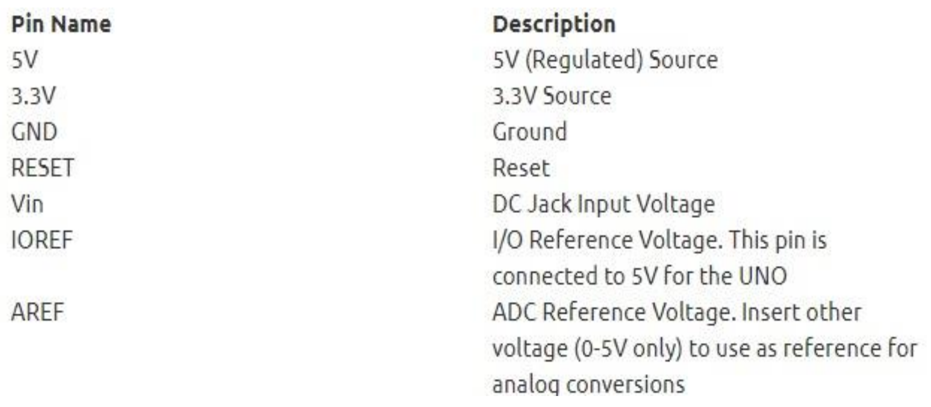
Pin number	Pin name	Type	Function Assigned to the pin
1	D1/TX	Digital Pin	Serial communication (TX)
2	D0/RX	Digital Pin	Serial communication (RX)
3	RESET	Other pin	Reset (active LOW)
4	GND	Ground	
5	D2	Digital Pin	External interrupt
6	~D3	Digital Pin	External interrupt 8-bit PWM output
7	D4	Digital Pin	
8	~D5	Digital Pin	8-bit PWM output
9	~D6	Digital Pin	8-bit PWM output
10	D7	Digital Pin	
11	D8	Digital Pin	
12	~D9	Digital Pin	8-bit PWM output
13	~D10	Digital Pin	SPI communication (SS-slave select), 8-bit PWM output
14	~D11	Digital Pin	SPI communication (MOSI-Master out slave in), 8-bit PWM output
15	D12	Digital Pin	SPI communication (MISO-Master in slave out)
16	D13	Digital Pin	SPI communication (SCK) Connected to a built-in LED
17		Power	
18	AREF	Analog Pin	Reference voltage for the analog inputs(0-5v)
19	D14 A0	Digital Pin Analog Pin	
20	D15 A1	Digital Pin Analog Pin	

21	D16, A2	Digital Pin Analog Pin	
22	D17, A3	Digital Pin Analog Pin	
23	D18, A4	Digital Pin Analog Pin	I2C communication (SDA)
24	D19, A5	Digital Pin Analog Pin	I2C communication (SCL)
25	D20 A6	Digital Pin Analog Pin	Cannot be used as a digital pin
26	D21 A7	Digital Pin Analog Pin	Cannot be used as a digital pin

Communication pins:

The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). ATmega328 also supports I2C (TWI) and SPI communication. Arduino Nano Provide SDA and SCL are available on A4 and A5 *pins*. SDA -> A4; SCL -> A5. Support I2C (TWI) communication using the Wire library. SPI connections are available on SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Although provided by the underlying hardware, these pins support SPI communication, which is not currently included in the Arduino language.

Power pin:



Pin Name	Description
5V	5V (Regulated) Source
3.3V	3.3V Source
GND	Ground
RESET	Reset
Vin	DC Jack Input Voltage
IOREF	I/O Reference Voltage. This pin is connected to 5V for the UNO
AREF	ADC Reference Voltage. Insert other voltage (0-5V only) to use as reference for analog conversions

How to Power Arduino Nano:

- Mini-B USB connection
- 7-15V unregulated external power supply (pin 30)
- 5V regulated external power supply (pin 27)

MiniB USB **Connector:** The most popular way to power the Arduino Nano board is **to use a** USB cable. **You** can use a MiniB USB cable connected to the USB port of your laptop, PC, or USB **5V power** adapter. **This** cable is also used to program Arduino Nano

VIN pin: You can also power the Arduino Nano with an **external unregulated** 6 **–** 20V power supply connected to the VIN pin (pin 30). **This** pin can also be used to power the microcontroller with a battery **Pin + 5V:** It is also possible to use **an** external **stabilized 5V power** supply connected to the pin **+ 5V (pin** 27). **However,** this method is not recommended as it bypasses voltage regulators. **If** you want to power the board in this way, you **need** to make sure that the voltage level is stable and not **over** 5V

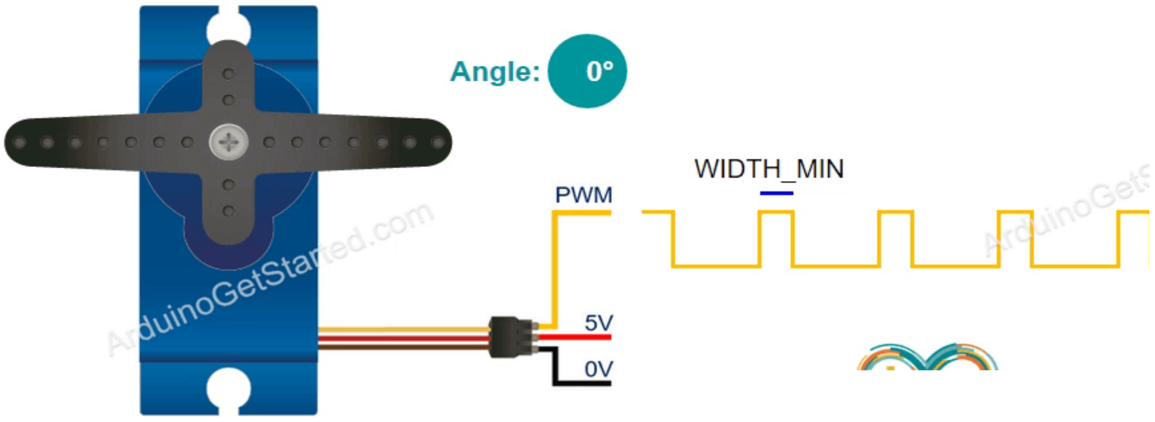
3.SERVO MOTOR:

Wire Configuration:

Wire Number	Wire Colour	Description
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange	PWM signal is given in through this wire to drive the motor

Steps:

1. it used to control the angular positions of the object
2. It can rotate its handle between 0 deg to 180 deg or 0 deg to 360 deg
3. If PWM's width=WIDTH_MIN,the servomotor rotates to 0 deg,
4. If PWM's width=WIDTH_MAX,the servomotor rotates to 180 deg,
5. If PWM's width is between WIDTH_MIN and WIDTH_MAX, the servomotor rotates to angle between 0 deg and 180 deg in proportion.



4.JUMPER WIRES:

Jumper wires are simply **wires that have connector pins at each end**, allowing them to be used to connect two points to each other without soldering.

SOURCE CODE:

For Arduino Color Sorter

```
#include <Servo.h>
#define S0 2
#define S1 3
#define S2 4
#define S3 5
#define sensorOut 6

Servo topServo;
Servo bottomServo;

int frequency = 0;
int color=0;
void setup() {
 pinMode(S0, OUTPUT);
 pinMode(S1, OUTPUT);
 pinMode(S2, OUTPUT);
 pinMode(S3, OUTPUT);
 pinMode(sensorOut, INPUT);

 // Setting frequency-scaling to 20%
 digitalWrite(S0, HIGH);
 digitalWrite(S1, LOW);

 topServo.attach(7);
 bottomServo.attach(8);

 Serial.begin(9600);
}

void loop() {

 topServo.write(115);
 delay(500);

 for(int i = 115; i > 65; i--) {
  topServo.write(i);
  delay(2);
 }
 delay(500);

 color = readColor();
 delay(10);

 switch (color) {
  case 1:
  bottomServo.write(50);
  break;

  case 2:
  bottomServo.write(75);
  break;

  case 3:
  bottomServo.write(100);
  break;

  case 4:
  bottomServo.write(125);
  break;

  case 5:
  bottomServo.write(150);
  break;

  case 6:
  bottomServo.write(175);
  break;

  case 0:
  break;
 }
 delay(300);

 for(int i = 65; i > 29; i--) {
  topServo.write(i);
  delay(2);
 }
 delay(200);

 for(int i = 29; i < 115; i++) {
  topServo.write(i);
  delay(2);
 }
 color=0;
}

// Custom Function - readColor()
int readColor() {
 // Setting red filtered photodiodes to be read
 digitalWrite(S2, LOW);
 digitalWrite(S3, LOW);
 // Reading the output frequency
 frequency = pulseIn(sensorOut, LOW);
 int R = frequency;
 // Printing the value on the serial monitor
 Serial.print("R= ");//printing name
 Serial.print(frequency);//printing RED color frequency
 Serial.print("  ");
 delay(50);

 // Setting Green filtered photodiodes to be read
 digitalWrite(S2, HIGH);
 digitalWrite(S3, HIGH);
 // Reading the output frequency
 frequency = pulseIn(sensorOut, LOW);
 int G = frequency;
 // Printing the value on the serial monitor
 Serial.print("G= ");//printing name
 Serial.print(frequency);//printing RED color frequency
 Serial.print("  ");
 delay(50);

 // Setting Blue filtered photodiodes to be read
 digitalWrite(S2, LOW);
 digitalWrite(S3, HIGH);
 // Reading the output frequency
 frequency = pulseIn(sensorOut, LOW);
 int B = frequency;
 // Printing the value on the serial monitor
 Serial.print("B= ");//printing name
 Serial.print(frequency);//printing RED color frequency
 Serial.println("  ");
 delay(50);

 if(R<45 & R>32 & G<65 & G>55){
  color = 1; // Red
 }
 if(G<55 & G>43 & B<47 &B>35){
  color = 2; // Orange
 }
 if(R<53 & R>40 & G<53 & G>40){
  color = 3; // Green
 }
 if(R<38 & R>24 & G<44 & G>30){
  color = 4; // Yellow
 }
 if(R<56 & R>46 & G<65 & G>55){
  color = 5; // Brown
 }
 if (G<58 & G>45 & B<40 &B>26){
  color = 6; // Blue
 }
 return color;
}
```

At this point, first we need to program the Arduino and then finish the assembly. Here's the Arduino Code:

DESCRIPTION OF THE CODE:

1. So, we need to include the "Servo.h" library, define the pins to which the color sensor will be connected, create the servo objects and declare some variables needed for the program.
2. In the setup section we need to define the pins as Outputs and Inputs, set the frequency-scaling for the color sensor, define the servo pins and start the serial communication for printing the results of the color read on the serial monitor.
3. In the loop section, our program starts with moving the top servo motor to the position of the skittle charger. (Note that this value of 115 suits to my parts and my servo motor, so you should adjust this value as well as the following values for the servo motors according to your build).
4. Next using the "for" loop we will rotate and bring the skittle to the position of the color sensor. We are using a "for" loop so that we can control the speed of the rotation by changing the delay time in loop.
5. Next, after half a second delay, using the custom made function, readColor() we will read the color of the skittle. Here's the code of the custom function. Using the four control pins and the frequency output pin of the color sensor we read color of the skittle.
6. The sensor reads 3 different values for each skittle, Red, Green and Blue and according to these values we tell what the actual color is.
7. Here are the RGB values that I got from the sensor for each skittle.(Note that these values can vary because the sensors isn't always accurate). Therefore, using these "if" statements we allow the sensor an error of around +-5 of the tested value for the particular color. So for example if we have a Red skittle, the first "if" statement will be true and the variable "color" will get the value 1.
8. The readColor() custom function does and after that using a "switch-case" statement we rotate the bottom servo to the particular position.

9. At the end we further rotate the top servo motor until the skittle drops into the guide rail and again send it back to the initial position so that the process can repeated.

CONCLUSION:

The color sorter project utilizing Arduino Nano demonstrates an efficient and versatile solution for automated color-based sorting tasks. By integrating a color sensor, servo motors, and Arduino Nano, the system can accurately identify and sort objects based on their color characteristics.The Arduino Nano's compact size and robust functionality make it well-suited for embedded systems like this color sorter. The color sensor's ability to differentiate between red, green, and blue components enables precise color recognition. The servo motors, controlled by Arduino Nano, facilitate the physical sorting process by moving objects to designated locations based on their color classification.This project showcases the power of microcontroller-based systems in real-time automation, offering potential applications in industries such as manufacturing, agriculture, and recycling. With further enhancements and optimizations, such as machine learning algorithms for advanced color detection or conveyor belt integration for continuous sorting, this color sorter using Arduino Nano can be tailored to specific industrial or educational needs.