ARDUINO COLOUR SORTER

A course project report submitted in partial fulfilment of the requirement

of

SMART SYSTEM DESIGN

by

A. KRUTHI	(2205A21002)
M. SUSMITHA	(2205A21017)
A. JASPER	(2203A51535)
A. NIKITHA	(2203A51534)

Under the guidance of
Mr. Y. Srikanth,
Asst. Prof., Department of ECE

Mrs. K. Manasa,
Asst. Prof., Department of ECE

SR UNIVERSITY
AnanthaSagar (V), Hasanparthy (M), Warangal (U),
Telangana - 506371

ABSTRACT

The Arduino colour sorter project is an innovative system designed to efficiently sort objects based on their colours. By utilizing an Arduino microcontroller and a colour sensor, the project offers a cost-effective and versatile solution for automated sorting applications. It combines programming skills, sensor integration, and mechanical design to create a compact and reliable sorting mechanism.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction:

The Arduino color sorter project aims to develop an automated system for efficient object sorting based on color recognition. This project combines the power of Arduino microcontroller and a color sensor to create a cost-effective and versatile sorting mechanism.

1.2 Objectives:

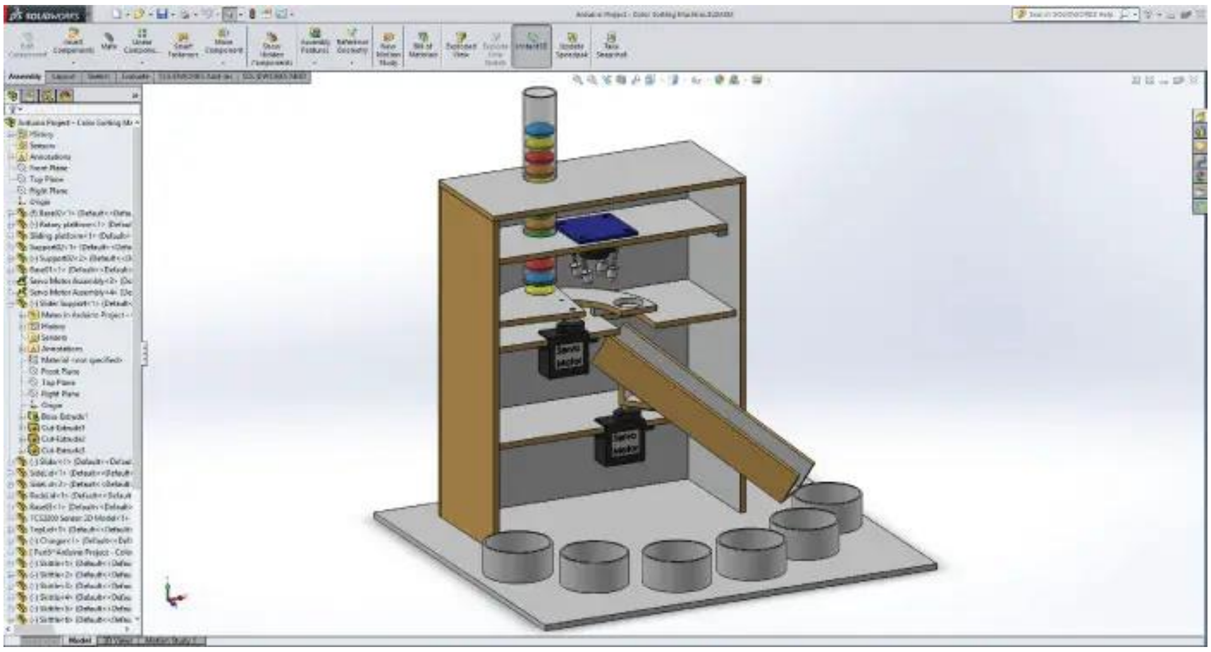
The main objectives of this project are to design a reliable and compact color sorting system, integrate Arduino Uno microcontroller and a color sensor, program the Arduino for color recognition and sorting, and evaluate the performance of the system in terms of accuracy and efficiency.

CHAPTER 2

PROJECT IMPLEMENTATION

2.1 Block Diagram of the Project:

The block diagram illustrates the components used in the project, including Arduino Uno, color sensor, motor drivers, and sorting mechanism, highlighting their interconnections and functions within the system.



2.2 Hardware Description:

All we need for this Arduino project is one color sensor (TCS3200) and two hobbyist servo motors, which makes this project quite simple but yet very fun to build it. In the first place, using the Solidworks 3D modeling software we made the design of the color sorter and here's its working principle:

- Initially, the colored skittles which are held in the charger drop into the platform attached on the top servo motor.
- Then the servo motor rotates and brings the skittle to the color sensor which detects its color.

- After that the bottom servo motor rotates to the particular position and then the top servo motor rotates again till the skittle drop into the guide rail.

2.3 Programming on Arduino:

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. 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.

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.

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.

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. 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.

Here are the RGB values that we got from the sensor for each skittle. Note that these values can vary because the sensor 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. So that’s what the readColor() custom function does and after that using a “switch-case” statement we rotate the bottom servo to the particular position. 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.

CHAPTER 3

CIRCUIT DIAGRAM AND DESCRIPTION

3.1 Working:

The Arduino-based color sorter project is designed to sort objects based on their color using a color sensor and actuator mechanisms controlled by an Arduino board. Here's a general overview of how such a project might work:

1. **Hardware setup:**

- Arduino board: Use an Arduino Uno or any other compatible board as the microcontroller.
- Color sensor: Connect a color sensor module (e.g., TCS3200 or TCS34725) to the Arduino board. The color sensor detects the RGB values of the objects to be sorted.
- Actuators: Connect servo motors or stepper motors to the Arduino board. These actuators will be responsible for moving the sorting mechanism.

2. **Calibration**:

- Calibrate the color sensor to accurately detect colors. This involves adjusting the sensor's sensitivity and calibration values for different colors.
- Set up the sorting mechanism by aligning the positions where the objects will be sorted based on their color.

3. **Code implementation:**

- Write the Arduino code using the Arduino IDE or any other compatible development environment.
- Configure the color sensor and actuators using appropriate libraries.
- Define the pins used for connecting the color sensor and actuators.
- Implement a loop that continuously reads the color values from the sensor.

4. **Sorting algorithm:**
- Based on the color values obtained from the sensor, implement a sorting algorithm that decides the appropriate action to take.
- For example, you might use a simple if-else statement to determine the color range for each sorting bin and move the corresponding actuator to direct the object to the correct bin.

5. **Object sorting:**

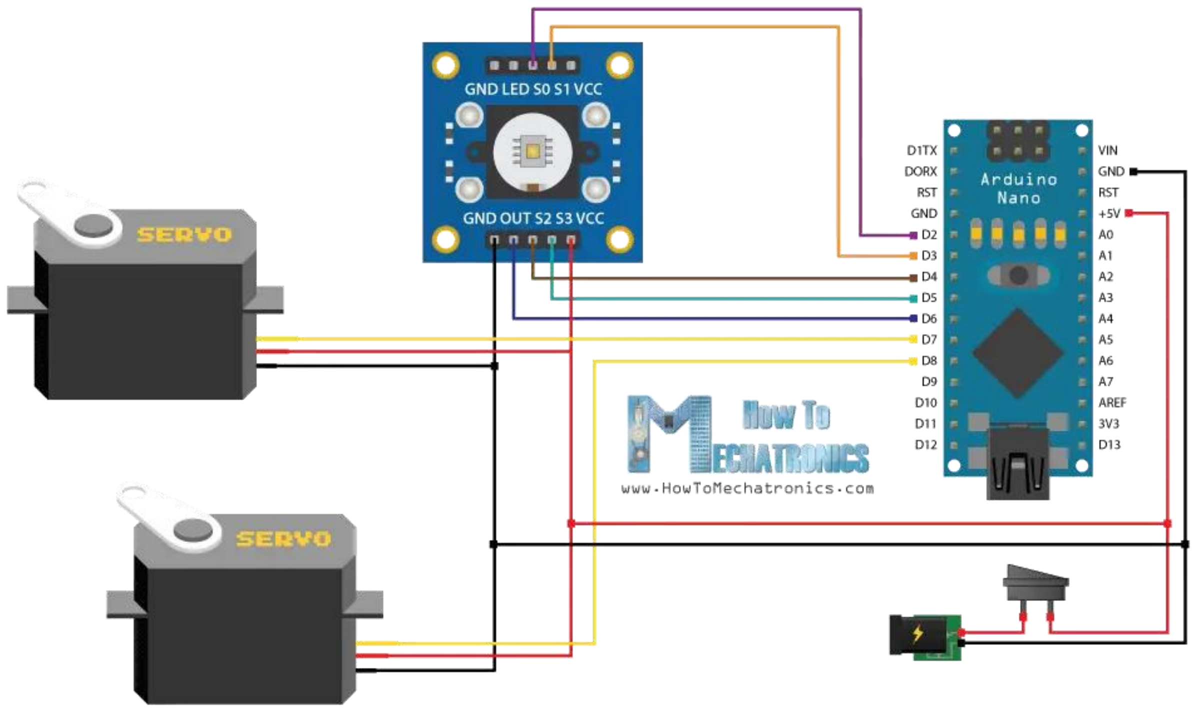
- When an object is detected, the color sensor will provide RGB values.
- Compare the RGB values with predefined color ranges to identify the color of the object.
- Based on the color identification, move the appropriate actuator to divert the object to the correct bin.

6. **Repeat:**
- The loop continues to detect and sort objects as they pass through the system.

7. **Additional features**:

- You can enhance the project by adding features like object counting, error handling, and user interfaces (such as an LCD display or LEDs) to indicate the sorting process.

It's important to note that the specifics of the project may vary depending on the design and components used. This is a general outline of how an Arduino-based color sorter project can work. Feel free to adapt and modify the steps according to your specific requirements and hardware.



3.2 Results:

The results of an Arduino-based gem color sorter project would depend on the accuracy and efficiency of the color detection and sorting mechanism. Here are some potential outcomes and results you can expect from such a project:

1.Accurate color detection:

The color sensor, when properly calibrated, should be able to accurately detect the RGB values of the gems or objects being sorted. This would enable the system to identify the colors and make appropriate sorting decisions based on those values.

2.Efficient sorting:

The sorting mechanism, controlled by the Arduino board, should be able to swiftly and accurately move the gems into their respective bins or containers based on their colors. The efficiency of the sorting process would depend on factors such as the speed of the actuators, the precision of their movements, and the design of the sorting mechanism.

3.Reliable performance:

The Arduino board, when programmed correctly, should provide reliable performance in terms of reading the color values, making sorting decisions, and controlling the actuators. The system should be able to handle a continuous flow of gems and consistently sort them without errors or malfunctions.

4.Sorting accuracy:

The accuracy of the sorting process would depend on the quality of the color sensor, calibration process, and the design of the sorting algorithm. If the color ranges for different bins are appropriately set and the algorithm is robust, the sorting accuracy should be high, with minimal misclassifications or errors.

5.Flexibility and customization:

An Arduino-based color sorter project offers the flexibility to customize the sorting process. You can modify the code and sorting algorithm to accommodate different color ranges or even sort objects based on multiple parameters (e.g., size or shape) in addition to color.

6.Project scalability:

The project can be scaled up or down depending on the size and complexity of the sorting task. With appropriate adjustments and enhancements, the Arduino-based color sorter can handle larger volumes of gems or be adapted to sort different types of objects beyond gems.

It's important to note that the specific results and performance of an Arduino-based color sorter project can vary based on the implementation, components used, and the expertise of the builder. However, with careful calibration, programming, and design considerations, the project has the potential to achieve accurate and efficient gem sorting results.

3.3 Advantages:

1. Affordability:

Arduino boards and components are relatively inexpensive, making the project more affordable compared to professional sorting systems.

2. Customizability:

Arduino-based projects offer a high level of customization. You can modify and adapt the code, sorting algorithm, and hardware to suit your specific requirements and gem sorting needs.

3. Ease of use:

Arduino boards are user-friendly, and the Arduino IDE provides a simple programming environment, making it accessible for beginners and hobbyists to work on the project.

4. Educational value:

Building an Arduino-based gem color sorter project can serve as a valuable educational experience, allowing you to learn and gain hands-on experience with electronics, programming, and automation principles.

5. Scalability:

The project can be scaled up or down depending on the size of the gems and the sorting requirements. You can adjust the hardware and software components to handle different volumes of gems and expand the system if needed.

3.4 Disadvantages:

1.Limited throughput:

Arduino boards are not designed for high-speed or high-throughput applications. The processing power and I/O capabilities of Arduino may limit the sorting speed and overall throughput of the system compared to more advanced sorting solutions.

2.Accuracy limitations:

While Arduino-based color sensors can provide accurate color detection, they may not match the precision and reliability of professional-grade sensors used in industrial sorting systems. The overall accuracy of the sorting process may be lower, leading to occasional misclassifications or errors.

3. Mechanical limitations:

The mechanical components used in the sorting mechanism, such as servos or stepper motors, may not offer the same level of precision, speed, or durability as industrial-grade actuators. This can impact the overall sorting performance and system longevity.

4. Complexity for complex sorting tasks:

Arduino-based color sorters are well-suited for simple sorting tasks based on color. However, if you require more advanced sorting capabilities, such as sorting based on multiple parameters (color, size, shape, etc.), the complexity of the project increases, and it may be more challenging to implement and fine-tune the sorting algorithm.

5. Maintenance and support:

As a DIY project, the maintenance and support for an Arduino-based color sorter may solely rely on your own expertise or the Arduino community. Professional sorting systems often come with dedicated support and maintenance services.

CHAPTER 4
CONCLUSION

4.1 Conclusion:

In conclusion, an Arduino-based gem color sorter project offers an affordable and customizable solution for sorting gems based on color. It provides a valuable educational experience and can be scaled to different sorting requirements. However, it may have limitations in terms of throughput, accuracy, and mechanical components compared to professional sorting systems. Careful consideration of these factors is necessary to determine its suitability for specific applications.

4.2 Future scope:

The future scope of an Arduino-based gem color sorter project includes potential advancements and improvements such as:

1. Integration of advanced color sensing technologies for enhanced accuracy.
2. Implementation of machine learning algorithms for intelligent sorting based on multiple parameters.
3. Integration of conveyor systems for higher throughput.
4. Incorporation of robotic arms for more precise and flexible sorting.
5. Integration of IoT capabilities for remote monitoring and control.
6. Development of user-friendly interfaces and software for easier operation and customization.

APPENDIX

```
/*    Arduino Project - Color Sorting Machine   */
#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;
}
```