Automated Ball Color Sorting System

1st Andrey Fritz L. Solijon
Department of Computer Engineering and Mechatronics
Pagadian City, Philippines
andreyfritz.solijon@g.msuiit.edu.ph

2nd Chris Immanuel I. Arcasa
Department of Computer Engineering and Mechatronics
Iligan City, Philippines
chrisimmanuel.arcasa@g.msuiit.edu.ph

3rd Thaddeus Rosales
Department of Computer Engineering and Mechatronics
Butuan City, Philippines
thaddeus.rosales@g.msuiit.edu.ph

***Abstract*—This paper presents the design and implementation of an Automated Ball Color Sorting System utilizing an Arduino microcontroller running FreeRTOS. The system is engineered to detect and sort colored balls into appropriate bins based on real-time color identification. A TCS34725 color sensor captures the color data, which is processed and used to actuate servo motors that control the sorting mechanism. The implementation leverages FreeRTOS to manage concurrent tasks effectively, including sensor reading, color processing, and servo control. Inter-task communication is handled using FreeRTOS queues, ensuring reliable and deterministic behavior. The Sensor Reading Task collects color data and transmits it to the Color Processing Task, which interprets the color and sends corresponding commands to the Servo Control Task. This modular approach promotes system scalability and facilitates support for additional colors. The project showcases practical applications of real-time systems concepts such as multitasking, scheduling, and hardware interfacing. Experimental results demonstrate the system's capability to accurately detect and sort colored balls with minimal latency and consistent performance, validating the effectiveness of the proposed architecture.**

***Keywords*—FreeRTOS, Arduino, Color Sorting, Automation**

I. Introduction

A. Background

In industrial automation, sorting systems play a crucial role in streamlining production lines, improving efficiency, and reducing human error. From packaging lines to recycling plants, automated sorting mechanisms are widely used to classify and direct items based on parameters such as size, weight, material, or color. A common example is the use of color sorting systems in quality control or material separation processes, where speed and accuracy are vital to ensure throughput and consistency. In such real-time applications, timing is critical, as delays or missed detections can lead to incorrect sorting, system bottlenecks, or overall degradation in performance.

One way to ensure high-speed and responsive operation in these systems is through multitasking—the ability to handle multiple operations concurrently. In embedded systems, multitasking allows sensors, processors, and actuators to work in parallel, rather than waiting for sequential instructions to complete. This leads to better responsiveness and more deterministic behavior, which is essential in time-sensitive industrial environments.

However, achieving multitasking on platforms such as the Arduino can be challenging. The standard Arduino programming model follows a sequential, blocking execution flow, which makes it difficult to implement concurrent behaviors. While some workarounds using timers and interrupts exist, they can quickly become complex and unmanageable for larger systems.

To address these limitations, FreeRTOS (Free Real-Time Operating System) can be integrated with the Arduino platform. FreeRTOS enables real-time multitasking by allowing multiple tasks to run independently, with precise control over task priorities, timing, and inter-task communication. It provides an effective abstraction for writing scalable, responsive embedded applications and is particularly well-suited for applications involving sensor reading, data processing, and actuator control.

In this project, we integrate FreeRTOS into an Automated Ball Color Sorting System, evaluate its performance compared to a traditional Arduino implementation, and justify its significance in systems requiring precise timing and concurrency. The system is designed to identify the color of balls using a TCS34725 sensor and sort them into respective bins using servo motors. With FreeRTOS managing three core tasks—sensor reading, color processing, and servo control—the system demonstrates how real-time task scheduling can improve sorting speed, responsiveness, and modularity in embedded automation projects.

B. Goals

The primary objective of this project is to design and implement a real-time automated ball color sorting system that demonstrates the practical application of multitasking in embedded systems using FreeRTOS. The specific goals of the project are as follows:

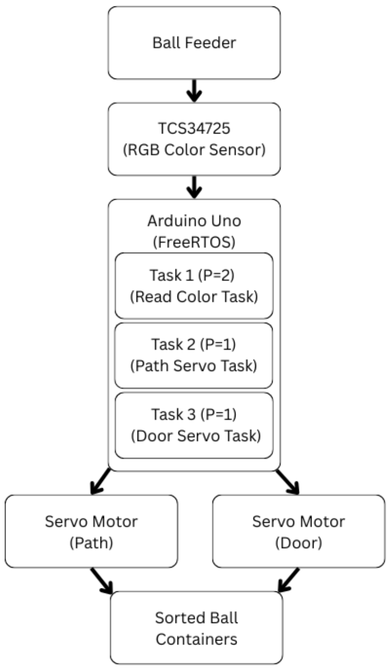
1. Develop a Functional Sorting Mechanism: Design and build a system that can detect the color of incoming balls using a TCS34725 color sensor and sort them into the correct bins using servo-actuated paths and gates.
2. Implement Real-Time Multitasking with FreeRTOS: Integrate FreeRTOS with the Arduino platform to manage multiple concurrent tasks such as color sensing, data processing, and motor actuation in a deterministic and time-efficient manner.
3. Demonstrate Inter-Task Communication: Utilize FreeRTOS features such as queues to enable reliable communication between tasks, ensuring timely and accurate transfer of sensor data and control commands.
4. Highlight the Importance of Timing and Concurrency in Industrial Applications: Showcase how real-time operating systems can enhance the performance and reliability of automation systems

in industrial contexts where precise timing and multitasking are essential.

5. Design a Modular and Scalable Architecture: Ensure that the system is easily extendable to accommodate additional colors, sensors, or actuators, demonstrating its flexibility for future industrial applications.

II. Design and Implementation

A. System Block Diagram



Ball Feeder
TCS34725 (RGB Color Sensor)
Arduino Uno (FreeRTOS)
Task 1 (P=2) (Read Color Task)
Task 2 (P=1) (Path Servo Task)
Task 3 (P=1) (Door Servo Task)
Servo Motor (Path)
Servo Motor (Door)
Sorted Ball Containers

Fig. 1. System Block Diagram

Figure 1 illustrates the architecture of the Automated Ball Color Sorting System managed by FreeRTOS on an Arduino Uno. The process begins with a ball feeder delivering balls to a TCS34725 RGB color sensor, which identifies each ball's color. This color data is then processed by the Arduino Uno, where FreeRTOS schedules three concurrent tasks based on their assigned priorities. Task 1, with the lowest priority (P=2), is dedicated to reading color data from the sensor. Tasks 2 (Path Servo Task) and 3 (Door Servo Task) are assigned a higher priority (P=1), ensuring that the control of the path and door servo motors, responsible for the physical sorting of the balls, takes precedence. Consequently, after the color is read, the higher-priority tasks quickly actuate the respective servo motors to guide each ball into the correct sorted ball containers.

B. Project Setup

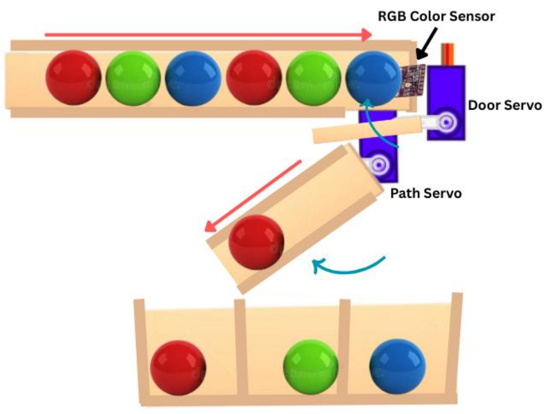


Fig. 2. System Concept

Figure 2 shows the setup of the operation flow of the Automated Ball Color Sorting System. Balls are loaded into an inclined ball feeder, allowing them to roll freely toward the TCS34725 RGB color sensor. Upon reaching the sensor, the ball's color is detected and processed by an Arduino Uno running FreeRTOS. The system uses three tasks to manage real-time operation: the Color Reading Task (Task 1, Priority 2), the Path Servo Control Task (Task 2, Priority 1), and the Door Servo Control Task (Task 3, Priority 1). After identifying the ball's color, the path servo motor rotates to align with the designated path leading to the appropriate bin. Once positioned, the door servo momentarily opens to release the ball, which then drops and follows the guided path to the correct container.

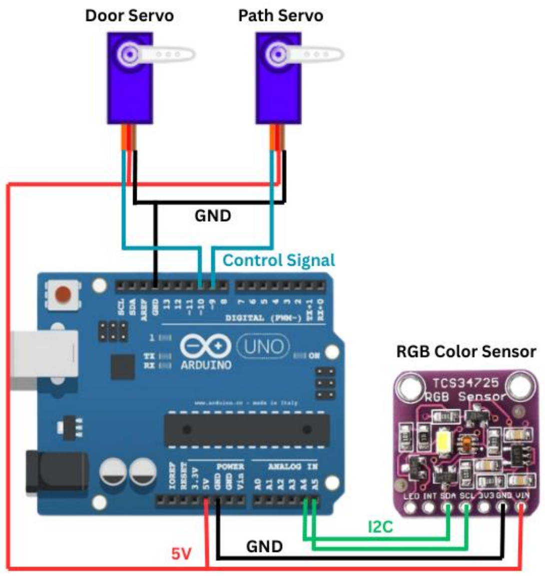


Fig. 3. Schematic Diagram

Figure 3 is the schematic diagram of the Automated Ball Color Sorting System that shows the electrical connections between the Arduino Uno, the TCS34725 color sensor, and the two servo motors (path and door). Both servo motors and the color sensor share the Arduino's 5V and GND power lines. The control signal for the path servo is connected to digital pin 9, while the door servo is controlled through digital pin 10. The TCS34725 color sensor communicates with the Arduino via the I²C protocol, with its SDA and SCL lines connected to analog pins A4 and A5, respectively. This setup enables synchronized power and communication across components in support of the automated sorting system.

III. Results and Discussion

A. Arduino Implementation

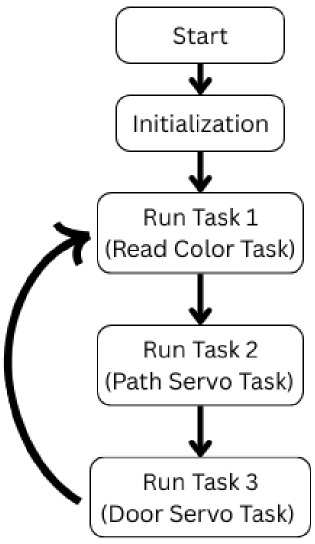


Fig. 4. Arduino Flow

Figure 4 illustrates the sequential flow of the Arduino-based implementation. The system successfully performed color sorting, with the tasks executed in a strict order to ensure proper operation. The Path Servo Task waits for the Read Color Task to complete before executing, and the Door Servo Task similarly waits for the Path Servo Task to finish. This sequential execution ensures reliable ball sorting, but it also highlights a limitation in multitasking capability compared to a real-time operating system like FreeRTOS.

B. FreeRTOS Implementation

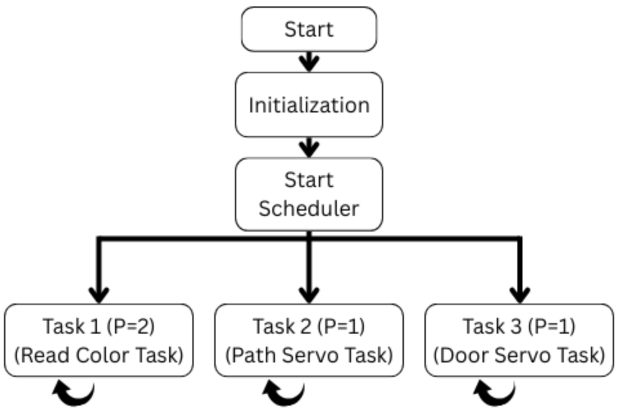


Fig. 5. FreeRTOS Flow

Figure 5 illustrates the flow of the FreeRTOS-based implementation of the system. The FreeRTOS implementation successfully performed multitasking, with the tasks sharing mutexes to synchronize concurrent operations. Tasks were properly synchronized using non-blocking delays and mutexes, ensuring consistent and reliable performance. Whenever a color is detected, Task 1 (the Read Color Task) uses a vTaskDelay to allow the two lower-priority tasks to run concurrently, optimizing the timing and overall speed of the sorting system.

C. Documentation

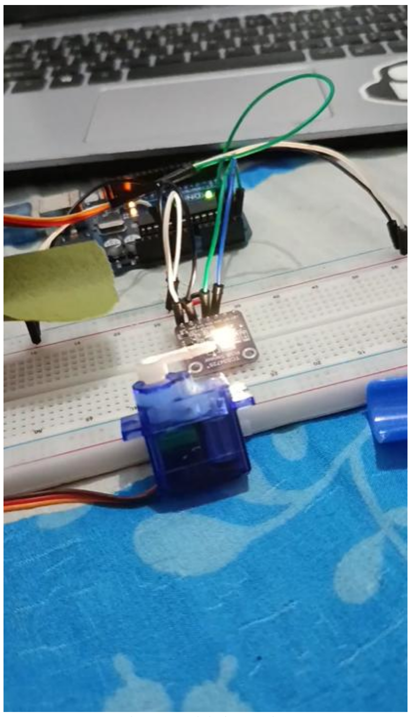


Fig. 5. Initial Testing

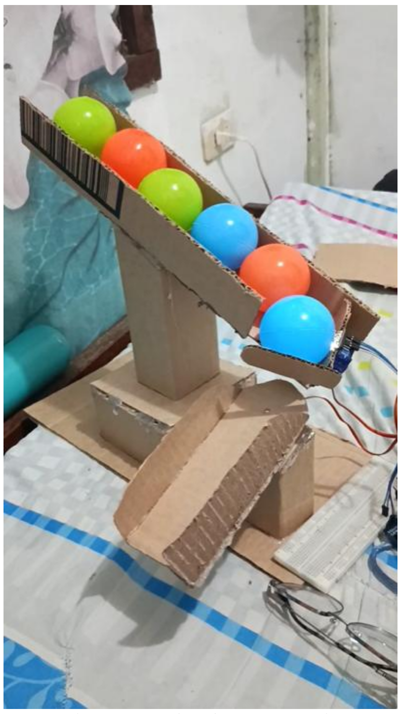


Fig. 6. Completed System Design

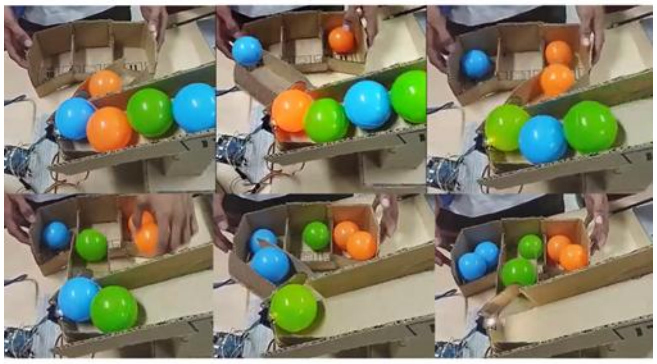


Fig. 7. Demonstration of the System

From what we've observed from our results, the FreeRTOS was successfully implemented and the system was successful in sorting the colored balls and storing them in their respective containers with occasional errors.

IV. CONCLUSION

Both the Arduino and FreeRTOS implementations of the automated ball color sorting system were successful; however, they exhibit significant differences in their approach to task execution and system performance. Ideally, the path and door servo motors should operate independently to maximize speed and responsiveness. The Arduino implementation, due to its sequential nature, prevents such concurrency—requiring the door servo to wait for the path servo to complete its action before proceeding. Although achieving concurrency in Arduino is possible, it typically requires extensive and complex coding.

In contrast, FreeRTOS offers robust APIs that facilitate the parallel execution of tasks through time slicing. This significantly enhances system speed and efficiency.

Justification:

In our automated ball color sorting system, the adoption of FreeRTOS offers several key advantages over the traditional Arduino implementation:

1. Modularity and Scalability – FreeRTOS allows the system logic to be modularized into separate tasks, making the codebase cleaner and easier to scale. This modular approach also enables future expansion, such as integrating an LCD display or adding remote control features.
2. Non-Blocking Behavior – While Arduino's use of delay() blocks the entire processor, FreeRTOS utilizes vTaskDelay(), which allows lower-priority tasks to run during wait periods. This improves system speed and responsiveness by reducing idle time.
3. Fault Isolation – If one of the servo motors malfunctions, FreeRTOS ensures that the Read Color Task continues to log data for debugging purposes, instead of the system becoming stuck. In Arduino, it is more challenging to isolate and manage such failures.
4. Deterministic Behavior – FreeRTOS guarantees that the highest-priority tasks are always executed first. This ensures that the Read Color Task, which has the highest priority, consistently captures color data without missing fast-moving balls.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(referenc*