# End-Term Project report on

## OBJECT SORTER ROBOT BASED ON SHAPE AND COLOR USING RASPBERRY PI

**Submitted by**

**Abhishek Sahoo (2241019026)**

**Jaydev Sahoo (2241019078)**

**Archit Kumar Sahoo (2241019040)**

**Debadutta Barik (2241002036)**

**Sunaina Pradhan (2241016436)**

**Bastav Kumar Swain (2241019590)**

**B. Tech. (CSE(IOT) 6th Semester (Section–2241037)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
Institute of Technical Education and Research

**SIKSHA 'O' ANUSANDHAN**
**DEEMED TO BE UNIVERSITY**

Bhubaneswar, Odisha, India.
(May 2025)

# Abstract

The **Object Sorter Robot Based on Shape and Colour Using Raspberry Pi** is an intelligent automation system designed to identify and categorize objects based on their physical characteristics—specifically, shape and colour. The primary goal of the project is to minimize human intervention in industrial and domestic sorting applications by using computer vision and robotic actuation. The system leverages a Raspberry Pi as the central controller, interfaced with a camera module for real-time image processing and a robotic arm or conveyor mechanism to physically sort the objects.

The camera captures images of incoming objects, and the image processing module, powered by OpenCV, extracts shape and colour features using contour detection and colour space analysis (such as HSV). The Raspberry Pi processes this data and classifies the object based on predefined parameters. Depending on the classification, the robot then actuates motors to place the object into the appropriate bin or zone. This integration of hardware and software creates an efficient, low-cost, and customizable solution suitable for small-scale sorting tasks in educational, manufacturing, or recycling settings.

This project demonstrates the practical application of embedded systems, computer vision, and automation in solving real-world problems. It also promotes skill development in areas such as Python programming, robotics, image processing, and electronics. With potential enhancements like machine learning integration or wireless communication, the system can be scaled for more advanced industrial needs, making it a strong foundation for future developments in smart automation.

# Contents

# Chapter 01: Introduction

## 1.1. Introduction

In today's fast-paced world, automation plays a crucial role in enhancing efficiency, reducing human effort, and minimizing errors in repetitive tasks. Object sorting is one such task that is widely needed in industries such as manufacturing, packaging, recycling, and agriculture. Traditionally, sorting has been performed manually or with complex and expensive machinery. However, with the advancement in embedded systems and image processing, it is now possible to develop low-cost and intelligent robotic systems that can perform object sorting efficiently.

The Object Sorter Robot is designed to detect and classify objects based on two primary parameters: **shape** and **colour**. By using a **Raspberry Pi** as the central processing unit, along with a camera module and motor-driven actuators, the system captures real-time images of the objects and processes them using Python and OpenCV libraries. The shape is identified through contour detection and edge analysis, while colour detection is performed using HSV (Hue, Saturation, Value) filtering. Based on these characteristics, the system categorizes and sorts the objects into designated bins or sections.

This project is a perfect blend of hardware and software integration, demonstrating practical knowledge in fields such as robotics, embedded systems, image processing, and automation. It not only showcases how affordable components like the Raspberry Pi can be used for industrial applications but also opens doors for innovation in smart factories, waste management, and educational robotics. The implementation of such a system can significantly improve speed, accuracy, and scalability in object classification and sorting processes.

## 1.2. Background

The rapid advancement of automation and robotics has significantly transformed modern industries, especially in areas involving repetitive tasks such as sorting and categorization. Traditionally, these tasks were performed manually or with the help of mechanical systems that lacked intelligence and adaptability. While large-scale industries have adopted expensive automated sorting systems using PLCs and advanced robotics, there remains a strong need

for affordable, intelligent, and compact solutions, especially for small-scale businesses, academic purposes, and research-based applications.

With the introduction of powerful single-board computers like the **Raspberry Pi**, developers and researchers now have the ability to implement intelligent systems using compact, low-cost hardware. The Raspberry Pi, coupled with open-source libraries such as **OpenCV**, has made it possible to integrate computer vision into robotic applications. This enables robots to "see" and make decisions based on visual input. Shape and colour detection, which were once considered complex tasks, can now be accomplished with relative ease using image processing algorithms and a camera module.

This background forms the basis for the development of the Object Sorter Robot. By harnessing the computational power of the Raspberry Pi and the flexibility of Python programming, the project aims to create an efficient sorting robot capable of recognizing and classifying objects based on visual characteristics. This not only serves as a cost-effective solution for automation but also provides a practical platform for learning and experimentation in the fields of embedded systems, machine vision, and robotics.

## 1.3 Project Objectives

The primary objective of this project is to design and develop an intelligent robot that can automatically identify, classify, and sort objects based on their **shape** and **colour** using image processing techniques. The system aims to reduce manual effort and human error in sorting tasks by leveraging the computational power of the **Raspberry Pi** and integrating it with a camera module and motor-driven actuators. By capturing real-time images of objects and processing them through **OpenCV**, the robot can accurately detect object features and direct them to the correct bins or zones using predefined sorting logic.

A secondary objective is to create a **cost-effective, flexible, and scalable** solution that can be easily deployed in educational setups, research labs, small-scale industries, and waste management systems. The project also serves to enhance understanding and practical experience in fields such as embedded systems, robotics, computer vision, and automation. Additionally, it lays the groundwork for future improvements such as machine learning-based classification, wireless control, and cloud integration, thereby extending its usability in smart industrial applications.

## 1.4 Scope

The scope of this project encompasses the development of a vision-based robotic system capable of sorting objects based on their **visual attributes—specifically shape and colour**. Using a camera module connected to a **Raspberry Pi**, the system captures real-time images of incoming objects and processes them using **image processing techniques** implemented with OpenCV. The processed data is used to classify each object, after which a robotic mechanism (such as a servo-based arm or conveyor system) sorts the item into appropriate bins. This system provides a low-cost and reliable automation solution for environments such as **recycling units, small manufacturing lines, packaging units, and educational labs**.

Furthermore, the project's modular design allows for future expansion in functionality and scalability. Features like **machine learning-based classification**, sorting by size or texture, and remote monitoring via IoT can be integrated to increase system intelligence and adaptability. The system also serves as a strong learning platform for students and researchers in the fields of **robotics, computer vision, embedded systems, and industrial automation**, promoting innovation in smart automation technologies that can benefit both academic and real-world applications.

## 1.5 Project Management

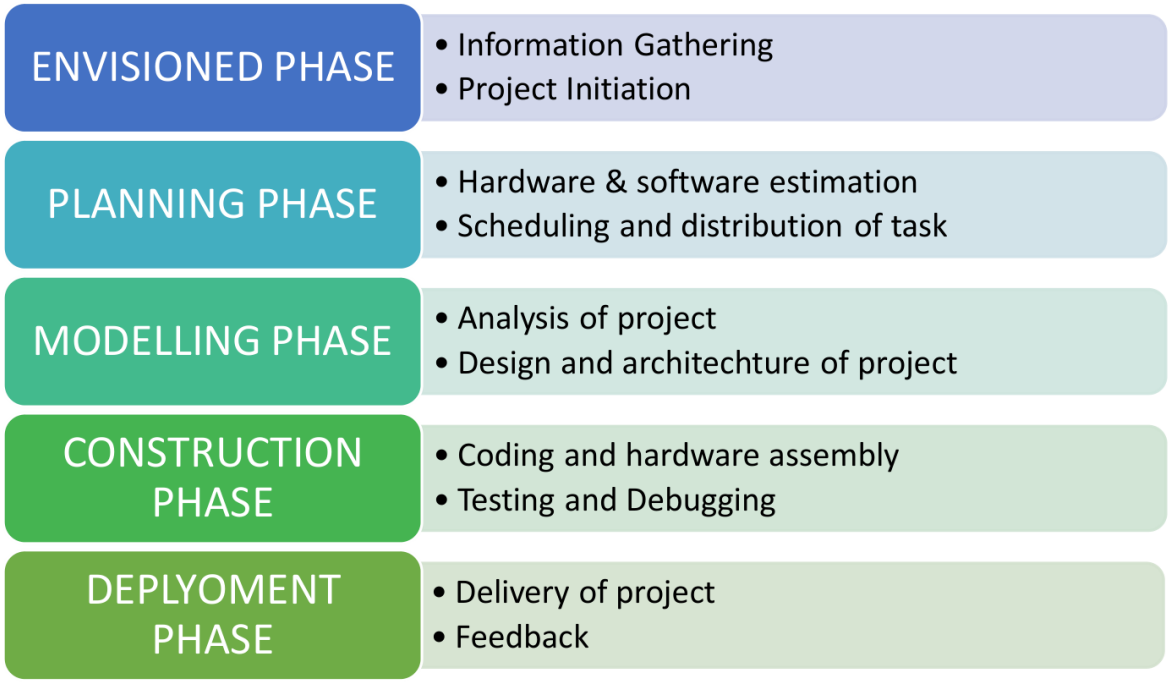



**Figure 1. Model of phases in project management.**

## 1.6 Overview and Benefits

The **Object Sorter Robot Based on Shape and Color Using Raspberry Pi** offers several practical benefits, including increased **efficiency, accuracy, and consistency** in sorting tasks while significantly **reducing human effort and error**. Its low-cost, compact design makes it ideal for small industries, educational institutions, and research projects where budget-friendly automation is needed. By utilizing **real-time image processing**, the system ensures fast and intelligent classification of objects, enhancing productivity. Additionally, it provides a hands-on platform for learning advanced technologies like **computer vision, embedded systems, and robotics**, encouraging innovation and technical skill development in the field of smart automation.

The object sorter robot offers several practical benefits across various domains, particularly in automation and smart manufacturing. By automating the sorting process based on shape and color, it significantly reduces the need for manual labor, increasing efficiency and accuracy in industrial environments such as packaging, recycling, and quality control. The use of Raspberry Pi ensures cost-effectiveness while still offering powerful processing capabilities for real-time image analysis. Additionally, the system's modularity allows for easy upgrades or customization, enabling it to adapt to different object types or sorting criteria as needed. Overall, the project demonstrates how intelligent automation combined with IoT can streamline workflows, reduce human error, and contribute to the development of smarter, more efficient systems.

## 1.7 Organization of the Report

The report is organised into the following chapters. Each chapter is unique on its own and is described with necessary theory to comprehend it.

Chapter 2 deals with background survey and review, Chapter 3 has the description of the theoretical aspects that has been acquired to commence the project work.

# Chapter 02: Background Review & Survey

## 2.1 Related Works

Several research and engineering efforts have focused on automating the object sorting process using embedded systems, especially with platforms like Raspberry Pi. A notable stream of work involves using image processing techniques combined with machine learning or rule-based logic to detect object features such as shape, color, and size. For instance, researchers have developed systems that use OpenCV with Raspberry Pi cameras to capture real-time images, which are then analyzed to determine an object's geometric shape and dominant color. These works typically involve RGB thresholding, edge detection (like Canny), and contour analysis for shape recognition, followed by actuation through motors or servo mechanisms for sorting.

Earlier studies also highlight the use of robotic arms or conveyor-based sorting systems where objects are placed in specific bins after classification. Some projects incorporate additional sensors like IR or ultrasonic to improve object detection accuracy and trigger the camera capture at optimal times. Comparisons have been made between using microcontrollers like Arduino and single-board computers like Raspberry Pi, showing that the latter provides more flexibility and computational power for image-based processing tasks. These prior works collectively demonstrate the feasibility and efficiency of Raspberry Pi-based object sorting robots in industrial automation and educational robotics.

Previous studies have implemented object sorting robots using Raspberry Pi and OpenCV for color and shape detection. These systems utilized contour analysis and HSV/RGB filtering for classification, with servo motors or conveyors for sorting. Many projects also explored IoT integration for real-time monitoring, demonstrating effective automation in applications like waste management, packaging lines, and educational robotics.

# Chapter 03: Theoretical Aspects

## 3.1 Internet of Things (IoT)

Integrating the Internet of Things (IoT) into the object sorter robot enhances its functionality, connectivity, and monitoring capabilities. In this project, Raspberry Pi serves as the central IoT-enabled hub, connecting sensors, camera modules, actuators, and cloud platforms. With IoT integration, the robot can not only detect and sort objects based on shape and color but also transmit real-time data such as object count, category, and sorting efficiency to a remote dashboard or mobile app. This feature allows for remote supervision, diagnostics, and performance analysis, which is particularly useful in industrial automation, quality control, and smart manufacturing setups.

Moreover, IoT enables the robot to participate in a networked ecosystem, where it can communicate with other machines or systems. For example, if the robot is deployed in a warehouse or assembly line, it can send alerts when bins are full or when unusual objects are detected. Data collected over time can be stored on cloud platforms like ThingSpeak, Firebase, or AWS IoT Core for predictive maintenance, analytics, and future optimization. This connectivity transforms the object sorter from a standalone machine into an intelligent, data-driven system that aligns with the vision of Industry 4.0 and smart factories.

## 3.2 Features of IoT

### a) Intelligence

In the context of the object sorter robot, **intelligence** refers to the ability of the system to process data and make decisions autonomously. Using image processing techniques on the Raspberry Pi, the robot intelligently identifies the shape and color of each object. This is achieved through algorithms such as contour detection, edge detection (for shape), and HSV filtering (for color), which mimic human visual understanding in a controlled environment.

Additionally, the system can be programmed to improve decision-making by incorporating logic for handling unexpected inputs—such as objects that don't match predefined criteria. As an IoT-enabled system, the robot can also use cloud-based intelligence (via APIs or ML models

hosted remotely) for advanced decision-making and pattern recognition, which adds a layer of smart adaptability to different sorting scenarios.

**b) Connectivity**

**Connectivity** is the backbone of any IoT system, and in this project, it allows the object sorter robot to communicate with external systems and users. The Raspberry Pi, equipped with Wi-Fi or Ethernet capabilities, enables real-time data transmission to cloud platforms like Firebase, AWS IoT, or Thing Speak. This connectivity allows for remote monitoring of sorting activity, object logs, and system status.

Furthermore, the robot can be integrated into a broader smart factory environment, where it communicates with other machines or control systems using MQTT or HTTP protocols. For example, when a bin is full or sorting criteria are changed, the system can send alerts or adjust behavior accordingly—all enabled by robust IoT connectivity.

**c) Dynamic Nature**

The **dynamic nature** of IoT systems means they respond to changing environments, and this robot exhibits that through real-time interaction with moving objects and environmental conditions. Whether the lighting changes, objects vary slightly in shape, or a sensor malfunctions, the robot is designed to adapt by recalibrating its detection thresholds or triggering fallback mechanisms.

Moreover, dynamic updates to the sorting logic can be done wirelessly. For example, the robot's shape or color recognition parameters can be updated remotely based on data insights, allowing it to adapt to new object types or categories without manual intervention—enhancing its usefulness in evolving industrial contexts.

**d) Enormous Scale**

IoT systems are often deployed at a massive scale, and while this project represents a single robot, its **scalability** makes it suitable for large deployments. Multiple units can be installed across a manufacturing line, each connected and managed from a central IoT dashboard. These units could share data and coordinate sorting tasks, increasing efficiency and reducing redundancy.

Additionally, the data generated from these systems can scale significantly over time. Information about object counts, types, frequency, and system performance can be stored and analyzed on the cloud, helping organizations .

### e) Sensing

**Sensing** is a critical component of this project. The object sorter uses a camera module (as the primary sensor) to visually sense and interpret object properties like color and shape. The system may also include IR sensors or proximity sensors to detect object presence, ensuring precise timing for capturing images and initiating the sorting process.

These sensors provide real-time data to the Raspberry Pi, allowing the robot to perceive its environment. This real-world data is the foundation for decision-making. Without accurate sensing, the robot would misclassify or miss objects entirely. Thus, robust sensor integration enables the robot to function reliably and efficiently in real-time scenarios.

### f) Heterogeneity

IoT environments consist of diverse devices, platforms, and communication protocols—a concept known as **heterogeneity**. In this project, heterogeneity is evident through the integration of multiple components like the Raspberry Pi (computing unit), camera (visual sensor), servo motors (actuators), and cloud platforms (remote data storage and analysis). These devices operate on different standards but work seamlessly together.

The project may also involve different programming languages and tools—Python for image processing, HTML/JS for dashboard visualization, and cloud APIs for communication. Managing this heterogeneity effectively ensures the system is robust, modular, and adaptable to future changes or upgrades in components or technologies.

### g) Security

As with any IoT-enabled system, **security** is paramount. Since the object sorter robot connects to the internet and transmits data, it must be protected from unauthorized access or malicious interference. Secure communication protocols (like HTTPS or MQTT with TLS) can be implemented to encrypt data during transmission.

Moreover, user authentication, access control, and secure firmware updates ensure that the system is not easily tampered with. Since the robot may operate in sensitive or industrial environments.

## Advantages of IoT

## a) Communication

IoT facilitates seamless **communication between devices**. In this project, the Raspberry Pi communicates with sensors, actuators (like servo motors), and cloud platforms for real-time data exchange. This allows the robot to send object data, sorting status, and alerts to a remote dashboard, enabling efficient monitoring and decision-making even from a distance.

### b) Automation and Control

The object sorter robot embodies IoT-enabled **automation**, eliminating the need for manual intervention in the sorting process. It autonomously detects the color and shape of objects, sorts them accordingly, and can even update its behavior based on remote instructions. This improves consistency, accuracy, and reduces human errors in repetitive tasks.

### c) Information

Through IoT integration, the robot collects and transmits **real-time information** such as object counts, types sorted, time of sorting, and system health. This data can be visualized on dashboards and stored for historical analysis, which is beneficial for auditing and improving sorting strategies in industrial applications.

d) **Monitoring**

IoT allows for **remote monitoring** of the robot's performance. Users can track operations through a web or mobile interface and get notifications for system errors or full bins. This feature enhances reliability and ensures timely maintenance or troubleshooting without being physically present near the machine.

### e) Efficiency

With continuous operation and minimal downtime, the IoT-based sorter greatly increases **efficiency** in workflows. Automated data handling, remote updates, and error detection reduce time and resource waste. The system can adapt to new object types or sorting rules dynamically, making it scalable and future-ready for evolving tasks.

## 3.3 Disadvantages of IoT

### a) Compatibility

**Compatibility issues** may arise when integrating diverse sensors, modules, or cloud platforms with Raspberry Pi. Ensuring that all components work together smoothly requires careful selection and sometimes custom configuration, especially when scaling the system or switching hardware interfaces.

### b) Complexity

IoT systems introduce **complex architecture** involving hardware, software, cloud integration, and communication protocols. This complexity increases the development time and demands knowledge in multiple domains like network

### c) Privacy/Security

Since the system transmits data over networks, **privacy and security** become concerns. Unauthorized access can lead to manipulation of the sorting process or theft of sensitive data. Without proper encryption, authentication, and firewall setups, the system remains vulnerable to cyberattacks or data breaches.

### d) Safety

If not properly designed, IoT-based systems may lead to **safety hazards**. For instance, malfunctioning actuators or sensor failures can result in mechanical errors or damage to the robot and surrounding equipment. Implementing safety protocols and fail-safe mechanisms is crucial to prevent such risks.

## 3.5 Application areas of IoT

This project fits into several **IoT application areas**, including **industrial automation**, **smart manufacturing**, **waste management**, and **education**. In industries, the robot can automate sorting on assembly lines, improving efficiency and reducing labor costs. In recycling facilities, it can be trained to separate plastics or metals based on shape and color. It also serves as an excellent educational model for students to understand real-world applications of IoT, robotics, and machine vision.

## 3.6 IOT Technologies and Protocols

### a) Bluetooth

Bluetooth can be used in the robot for short-range wireless communication, such as pairing with a mobile app for local control or setup without internet. It's low-power and ideal for small-scale, personal or educational projects.

### b) Zigbee

Zigbee is a **low-power, mesh-network protocol** suitable for communicating between multiple sorting robots across a warehouse. It allows devices to form a network, transmitting data over larger areas without relying on Wi-Fi.

### c) Z-Wave

Z-Wave, like Zigbee, is used for **low-bandwidth, short-distance communication**, mainly in home automation. In this project, it could be used for integrating the sorter into smart home or building systems where multiple IoT devices interact securely.

### d) Wi-Fi

Wi-Fi is the **primary communication method** for this project. The Raspberry Pi connects to cloud servers or dashboards via Wi-Fi, enabling real-time data transfer, remote updates, and user monitoring from any internet-enabled device.

### e) Cellular

Cellular networks (e.g., 4G/5G) can be employed if Wi-Fi is unavailable, especially in **remote or industrial field deployments**. A USB cellular dongle or GSM module with the Raspberry Pi ensures connectivity in off-grid areas.

### f) NFC

Near Field Communication (NFC) could be integrated for **user authentication** or task assignment. For instance, scanning an NFC tag could instruct the robot to switch to a specific sorting mode or log data under a user profile.

**g) LoRaWAN**

LoRaWAN is ideal for **long-range, low-power communication** in remote environments. It can be used to transmit sorting statistics or alerts over several kilometers with minimal power consumption, useful in agriculture or rural deployment scenarios.

## 3.7 Project Layout

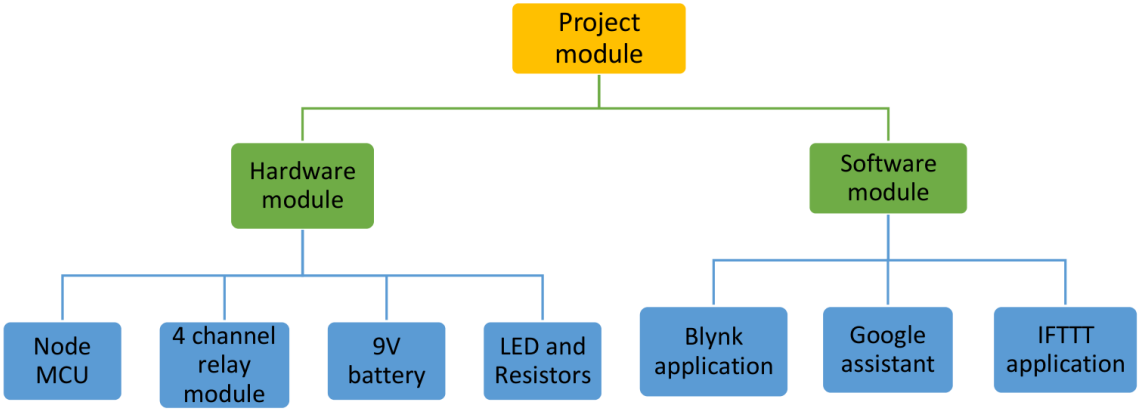



**Figure 2. Layout of project module**

### 3.7.1 Brief Description

The Object Sorter Robot Based on Shape and Colour Using Raspberry Pi is an intelligent, IoT-enabled system designed to automatically identify, classify, and sort objects based on their physical attributes such as shape and color. Utilizing a camera module, image processing techniques, and machine learning algorithms, the Raspberry Pi serves as the central controller that processes visual input, makes sorting decisions, and actuates motors to place objects in designated bins. This project aims to enhance automation in sorting tasks commonly found in manufacturing, recycling, and packaging industries, offering improved accuracy, efficiency, and reduced human intervention through real-time data handling and connectivity.

# Chapter 04: Hardware Requirements

## 4.1 Raspberry Pi 4 Model – 4GB

The Raspberry Pi 4 Model B with 4GB RAM is the central processing unit of the object sorter robot. It is responsible for image acquisition, processing, shape and color recognition, and decision-making tasks. With its quad-core processor and sufficient memory, it can efficiently run Python scripts, OpenCV libraries, and machine learning models necessary for real-time object classification. Its GPIO pins also enable communication with external components like servo motors and sensors.

### 4.1.1 Features

The **Raspberry Pi 4 Model B (4GB RAM)** is a powerful single-board computer designed for embedded systems, IoT projects, and educational purposes. It offers significantly improved performance over previous versions and is well-suited for tasks such as image processing and real-time control in robotics projects like the Object Sorter Robot. Below are its key features:

- **Quad-Core Processor**: Equipped with a Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC clocked at 1.5GHz, it delivers enhanced computing power for multitasking and running complex algorithms.
- **4GB LPDDR4-3200 SDRAM**: The 4GB RAM provides ample memory for running applications involving computer vision, Python scripts, and real-time data handling.
- **Dual Monitor Support**: It supports dual-display output via two micro- HDMI ports (up to 4K resolution), useful for development and debugging.
- **Connectivity**: Comes with Gigabit Ethernet, Bluetooth 5.0, and dual-band 2.4GHz and 5GHz IEEE 802.11ac Wi-Fi for seamless IoT integration and remote access.
- **GPIO Pins**: A 40-pin GPIO header enables direct interfacing with various sensors, servo motors, and external electronics.
- **USB Ports**: Four USB ports (2 × USB 3.0 and 2 × USB 2.0) allow connection to peripherals like keyboards, cameras, and external storage devices.

### 4.1.2 Pin Configuration

The Raspberry Pi 4 Model B features a 40-pin GPIO (General Purpose Input/Output) header that provides versatile interfacing capabilities for external components such as sensors, motors, and modules. Out of the 40 pins, 26 are GPIO pins that can be programmed as input or output, while the remaining pins include power supply pins (3.3V, 5V, and GND), I2C pins (SDA, SCL), SPI pins (MOSI, MISO, SCLK, CE0, CE1), UART (TXD, RXD), and PWM functionalities. These pins allow the Raspberry Pi to communicate with other hardware in real-time, making it ideal for embedded systems like the Object Sorter Robot. Each pin is assigned a specific function based on the Broadcom (BCM) numbering or physical layout, and users can configure them using Python or other supported languages for custom input/output tasks.

## 4.2 Raspberry Pi 5 Case

The Raspberry Pi 5 case is used to securely house and protect the Raspberry Pi board and associated components. It helps with thermal management by offering ventilation or built-in fan support, preventing overheating during prolonged operation. Additionally, it ensures that the Raspberry Pi is shielded from dust, accidental damage, and electrostatic discharge.

## 4.3 Robotic Arm

The robotic arm is a key mechanical component of the sorting system. Controlled by servo motors, it physically picks and places objects into the correct bins based on their classification. The arm’s range of motion and precision are essential for accurate and efficient sorting, especially when dealing with varied object sizes and shapes.

## 4.4 Servo Motor

Servo motors are used to provide precise angular movement required by the robotic arm for picking and placing tasks. Each joint in the robotic arm is typically controlled by a dedicated servo, enabling smooth and controlled motion. Their compact size and high torque make them ideal for robotics applications where precision is crucial.

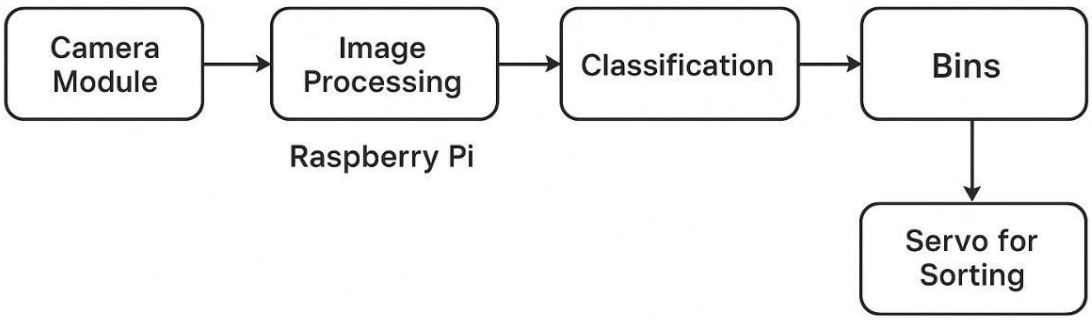
## 4.5 Servo Motor Driver

The servo motor driver is an interface that allows the Raspberry Pi to control multiple servo motors simultaneously. It manages the power supply and control signals required for each servo, ensuring stable and coordinated movement of the robotic arm during sorting operations. This driver is essential for reducing GPIO usage and preventing electrical overload on the Pi.

## 4.6 Sorting Bins

Sorting bins are the final physical destination for the classified objects. Each bin corresponds to a specific category—such as a certain shape or color—and receives objects as directed by the robotic arm. Proper placement and labeling of these bins ensure that the system maintains high sorting accuracy and supports easy collection or disposal afterward.

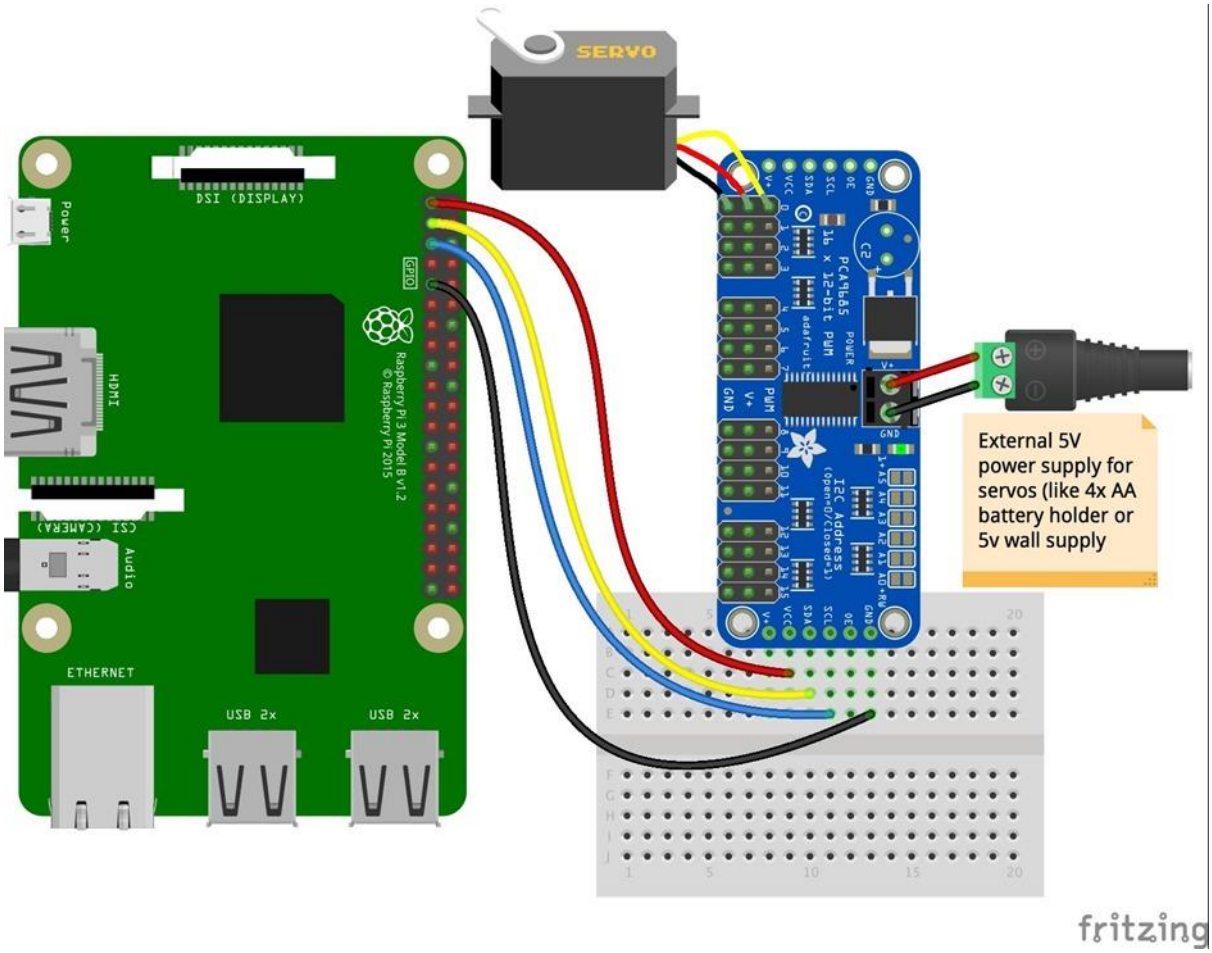
### 4.6.1 Block diagram of the proposed system



### 4.6.2 Working of the system

The **Object Sorter Robot** works by capturing images of objects using the **Raspberry Pi Camera Module**, which are then processed by the **Raspberry Pi 4** using **computer vision techniques** (via OpenCV or similar libraries) to identify the **shape and color** of each object. Based on the classification results, the **Raspberry Pi** sends control signals to the **servo motor driver**, which in turn moves the **robotic arm** powered by **servo motors** to pick up the object. The arm then places the object into the appropriate **sorting bin**. This entire process is performed in real-time, with the system intelligently coordinating between image processing and robotic actuation to achieve accurate and efficient sorting without human intervention.

### 4.6.3 Circuit Diagram



## 4.7 Components Required

**Table 1. Component listing.**

| Sl. No. | Components and Specification | Quantity |
|---|---|---|
| 1. | Raspberry Pi 4 Model 4GB | 1 |
| 2. | Raspberry Pi Camera Module 3 | 1 |
| 3. | Robotic Arm | 1 |
| 4. | Servo Motor | 4 |
| 5. | Servo Motor Driver | 1 |
| 6. | Sorting Bins | 3 |
| 7. | MicroSD Card | 1 |

# Chapter 05: Software Requirements

## 5.1 Thonny IDE

Thonny is a beginner-friendly Python IDE that comes pre-installed with Raspberry Pi OS. It provides a simple and clean interface to write, test, and debug Python scripts—making it ideal for projects involving image processing and hardware control. Features like step-by-step debugging, syntax error highlighting, and an integrated shell make it easy to interact with GPIO pins, camera modules, and libraries like OpenCV, RPi.GPIO, and numpy.

### 5.1.2 Raspberry Pi OS (formerly Raspbian)

- The official operating system for Raspberry Pi. It provides a Linux environment with pre-installed tools necessary for development.
- Required for running Python scripts and managing hardware components.

### 5.1.3 Python (v3.x)

- Primary programming language used in the project.
- Comes pre-installed with Raspberry Pi OS.

### 5.1.4 OpenCV ( Open Source Computer Vision Library)

- Used for image processing, object detection, and shape & color recognition.
- Enables real-time video analysis and image segmentation.

### 5.1.5 NumPy

- A Python library for numerical operations.
- Often used with OpenCV for array manipulation during image processing.

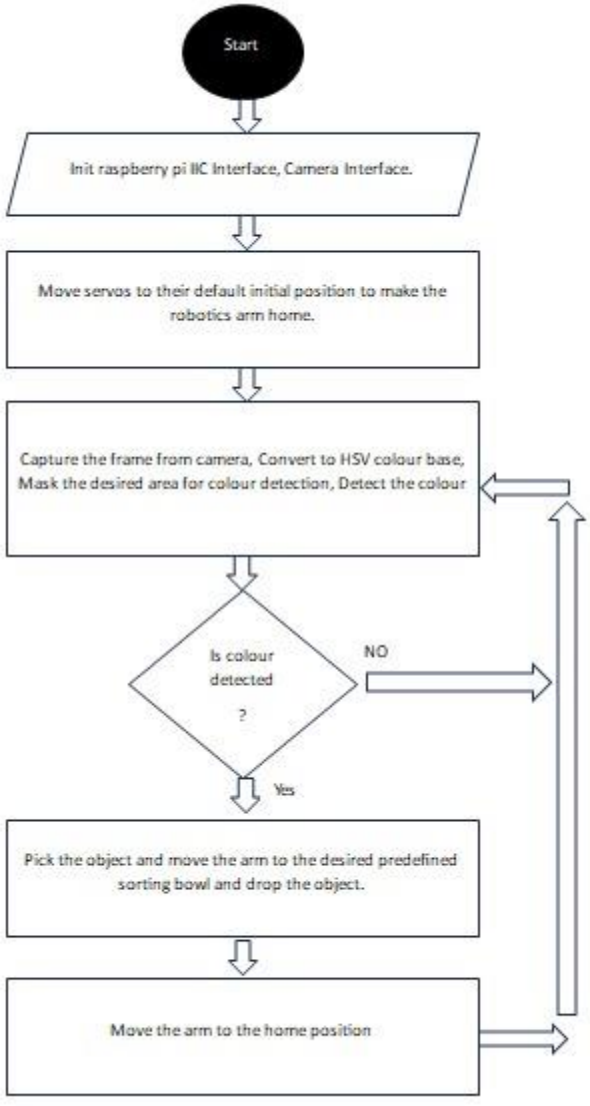
### 5.1.6 PCA9685 Library (Adafruit_PCA9685)

- If a servo motor driver like PCA9685 is used, this library allows PWM control of multiple servos via I2C.

### 5.1.7 Module Drivers / Picamera2

- ❖ Drivers or libraries required to interface with the Raspberry Pi Camera Module 3.
- ❖ picamera2 is commonly used for capturing images and streaming video from the camera.

## 5.2 Logic and Flowchart



# Chapter 06: Project development & Testing Aspects

## 6.1 Project Development

- The development of the **Object Sorter Robot** was carried out in multiple phases, starting from hardware assembly to the implementation of the image processing and sorting algorithm. Initially, the hardware components such as the **Raspberry Pi 4 (4GB)**, **Pi Camera Module**, **servo motors**, **robotic arm**, **motor driver**, and **sorting bins** were interconnected. The Raspberry Pi Camera was calibrated and mounted in a suitable position to capture a clear view of incoming objects.
- Once the hardware setup was complete, the software development began using **Thonny IDE** and **Python**. Libraries such as **OpenCV** and **NumPy** were used to detect and classify the shape and color of objects. The classification logic was implemented using color thresholding and contour analysis. Based on the classification output, the robotic arm was instructed to move via **PWM signals** to the appropriate bin. Each part of the system—from capturing the image to moving the object—was implemented in modular code to ensure maintainability and easy debugging.

### 6.2 Testing Aspects

- Testing was conducted in a phased manner to ensure the correct functioning of both hardware and software. Initial unit testing involved verifying the connectivity and functionality of individual components like the Raspberry Pi camera, servo motors, and robotic arm. GPIO pins were tested using Python libraries like **gpiozero** and **RPi.GPIO** to ensure accurate signal transmission.

- For software testing, sample objects of different shapes (circle, square, triangle) and colors (red, green, blue, etc.) were used under varying lighting conditions to assess the accuracy of the OpenCV-based detection algorithms. Test cases were also created to simulate edge cases like partially visible objects, overlapping items, or noise in the background.

## 6.3 Result

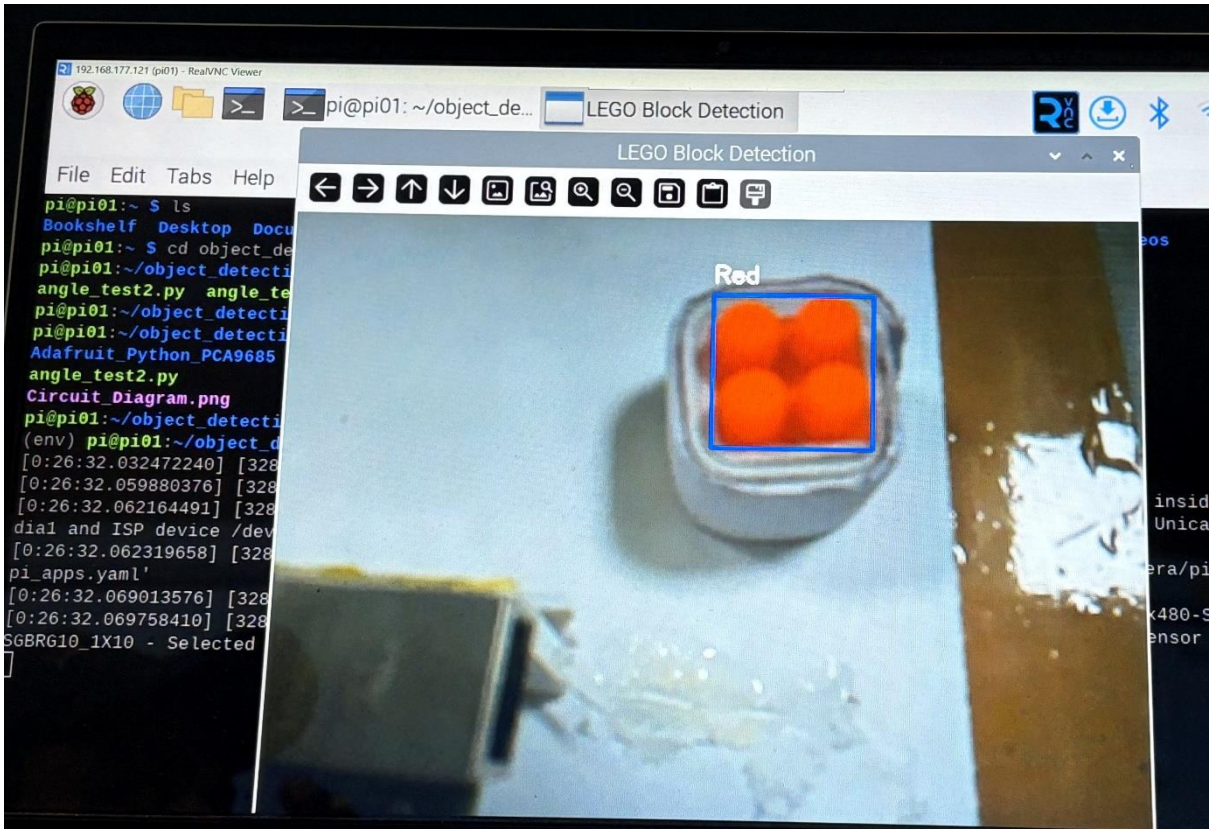



**Fig 6.1 Red Color Detected**

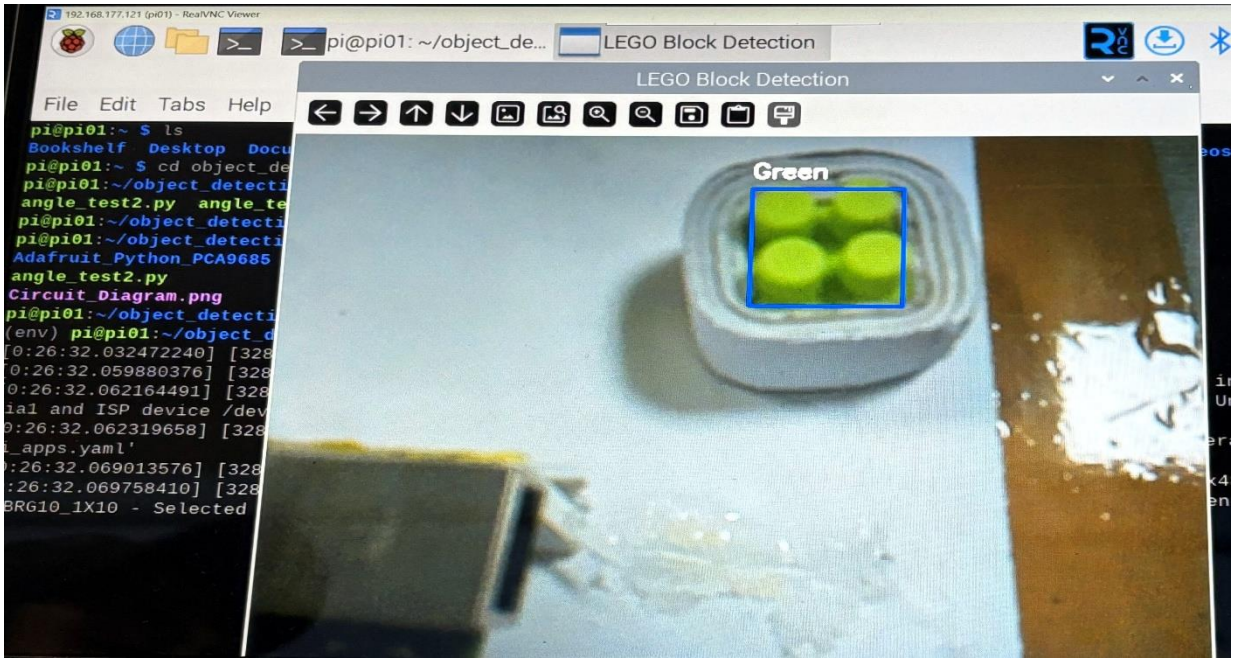



**Fig 6.2 Green color Detected**

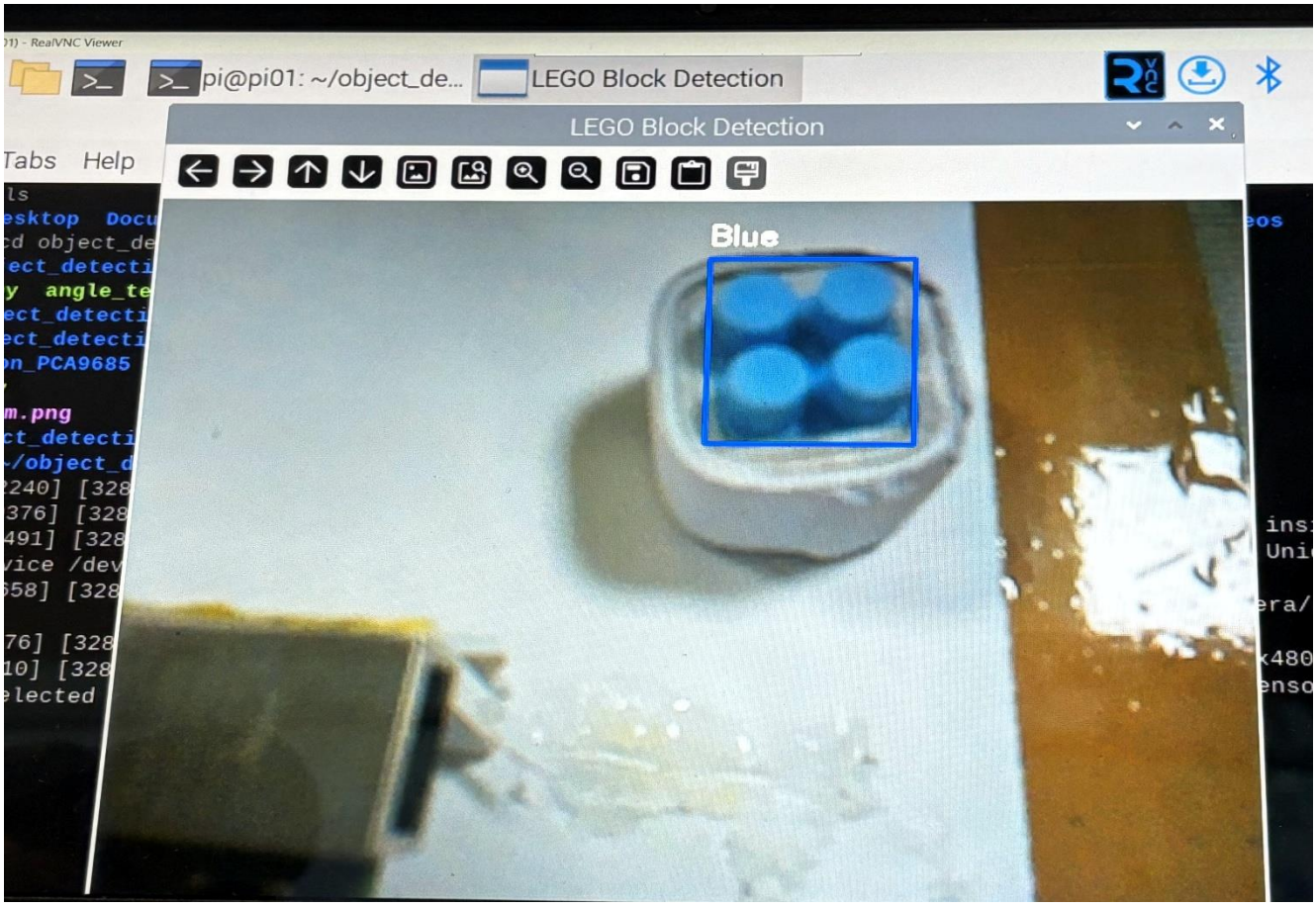



**Fig 6.3 Blue color detected**

**Fig 6.4 Model Architecture Diagram**

# Chapter 07: Conclusion & Future Scope

## 7.1 Conclusion

The project "Object Sorter Robot Based on Shape and Colour Using Raspberry Pi" successfully demonstrates an automated system capable of identifying and sorting objects based on their shape and colour. Utilizing a Raspberry Pi as the central processing unit, along with a camera for image processing and machine learning algorithms for classification, the robot can accurately differentiate objects. The system's efficiency is enhanced by integrating motors for movement and actuators for sorting the objects into designated categories. This project showcases the potential of combining Raspberry Pi with computer vision to create practical solutions for automation in various industries, such as inventory management or assembly lines, highlighting its ability to perform complex tasks with simplicity and reliability.

## 7.1 Limitations

The limitations of the "Object Sorter Robot Based on Shape and Colour Using Raspberry Pi" include its reliance on specific lighting conditions, as the system's accuracy can be affected by variations in ambient light, leading to errors in object detection and classification. The camera's resolution and processing power of the Raspberry Pi may also limit its ability to accurately identify small or highly similar objects, resulting in sorting mistakes. Additionally, the system's sorting speed and efficiency could be compromised when handling a large number of objects simultaneously, as the image processing and motor control may experience delays. The project also lacks advanced adaptability, making it less effective in dynamic environments with continuously changing object types or backgrounds.

## 7.2 Further Enhancement and Future Scope

The "Object Sorter Robot Based on Shape and Colour Using Raspberry Pi" project can be further enhanced in several ways to improve its functionality and broaden its application. First, the addition of machine learning algorithms, such as deep learning models, could enable the system to classify objects more accurately, even in varying lighting conditions or with more complex shapes. Integration of a more advanced camera with higher resolution or 3D imaging would improve

object detection and sorting precision. Moreover, implementing wireless communication, such as Wi-Fi or Bluetooth, could allow remote monitoring and control of the robot, making it more versatile. Future scope also includes expanding the sorting capabilities to handle more diverse object types, integrating sensors for weight or texture detection, and incorporating a more sophisticated robotic arm for better manipulation of objects. Additionally, by integrating the robot with an IoT platform, it could be used for smart manufacturing or warehouse automation, providing real-time data and improving operational efficiency.

## References


- Patil, A., & Jadhav. S (2020)-Object sorting system based on color and shape Recognition using Raspberry pi, IJERT
- Khandelwal, A., & Tiwari, R. (2018)-Automated Object Sorting Using Shape and color Detection with OpenCV. IJRTE
- Website (Visited On): https://www.youtube.com/watch?v=IlUHik79WC4&t=111s
- Raspberry Pi Cookbook: Software and Hardware Problems and solutions ,Simon Monk, O'Reilly Media .

# Appendix 01

## A01.1. Code Listing

| Line/Method | Purpose |
|---|---|
| cv2.circle(...) | Draws a red circle on a white background – simulating a red object |
| cv2.cvtColor(..., COLOR_BGR2HSV) | Converts the image from BGR to HSV format (used for better color detection) |
| cv2.inRange(...) | Creates masks to detect red color ranges in HSV (both light and dark red) |
| cv2.bitwise_and(...) | Applies the mask to extract only the red-colored part of the image |
| cv2.findContours(...) | Detects contours in the masked image (used to find shapes) |
| cv2.contourArea(cnt) | Filters out noise by ignoring very small contours |
| cv2.boundingRect(cnt) | Finds the smallest rectangle bounding the detected object |
| cv2.rectangle(...) | Draws a green rectangle around detected objects (for visualization) |

## A01.2. Main Code

```
import cv2
import numpy as np
import RPi.GPIO as GPIO
import time

SERVO_PIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(SERVO_PIN, GPIO.OUT)
servo = GPIO.PWM(SERVO_PIN, 50)
servo.start(0)

def move_servo(angle):
```

```
    duty = angle / 18 + 2
    GPIO.output(SERVO_PIN, True)
    servo.ChangeDutyCycle(duty)
    time.sleep(1)
    GPIO.output(SERVO_PIN, False)
    servo.ChangeDutyCycle(0)

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.resize(frame, (640, 480))
    blurred = cv2.GaussianBlur(frame, (5, 5), 0)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    lower_red1 = np.array([0, 120, 70])
    upper_red1 = np.array([10, 255, 255])
    lower_red2 = np.array([170, 120, 70])
    upper_red2 = np.array([180, 255, 255])

    mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
    mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
    mask = mask1 + mask2
contours,= cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for cnt in contours:
        area = cv2.contourArea(cnt)
        if area > 500:
            approx = cv2.approxPolyDP(cnt, 0.04 * cv2.arcLength(cnt, True), True
```

```
x, y, w, h = cv2.boundingRect(approx)
shape = "Unknown"
        if len(approx) == 3:
            shape = "Triangle"
            move_servo(30)
        elif len(approx) == 4:
            aspect_ratio = w / float(h)
            if 0.95 <= aspect_ratio <= 1.05:
                shape = "Square"
                move_servo(60)
            else:
                shape = "Rectangle"
                move_servo(90)
        elif len(approx) > 4:
            shape = "Circle"
            move_servo(120)
cv2.drawContours(frame, [approx], -1, (0, 255, 0), 3)
cv2.putText(frame, shape, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)
cv2.imshow("Object Detection", frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
      break
servo.stop()
GPIO.cleanup()
cap.release()
cv2.destroyAllWindows()
```

## A01.3. Libraries

### 1. Raspberry Pi OS

Raspberry Pi OS (formerly Raspbian) is the official operating system for the Raspberry Pi hardware platform. It is a Debian-based Linux distribution optimized specifically for Raspberry Pi devices, providing a lightweight, stable, and user-friendly environment. Raspberry Pi OS comes with a suite of pre-installed software and development tools, making it ideal for learning programming, building electronics projects, and running IoT or robotics applications. Its wide community support and compatibility with hardware make it the go-to OS for Raspberry Pi enthusiasts.

### 2. cv2 (OpenCV-Python)

`cv2` is the Python binding for OpenCV (Open Source Computer Vision Library), a powerful library designed for real-time computer vision and image processing tasks. It includes thousands of algorithms for image filtering, feature detection, object recognition, video analysis, and more. On Raspberry Pi, `cv2` is commonly used to capture and process images or video streams from connected cameras, enabling projects such as motion detection, face recognition, or augmented reality. Its extensive functionality and support for hardware acceleration make it a popular choice for vision-based projects.

### 3. NumPy

NumPy is a foundational library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices along with a broad collection of mathematical functions to operate on them efficiently. In computer vision projects with Raspberry Pi, NumPy is essential for handling image data, as images can be represented as arrays of pixel values. It allows easy manipulation and processing of these arrays, making tasks like filtering, transformation, and feature extraction straightforward. NumPy also integrates seamlessly with OpenCV and other scientific libraries.

### 4. Picamera2

Picamera2 is the modern Python library for interfacing with the Raspberry Pi Camera Module on Raspberry Pi OS. It replaces the original Picamera library, providing enhanced support for the latest camera stack on Raspberry Pi, including libcamera. Picamera2 offers a simple API to capture images and videos, configure camera settings, and stream frames directly into Python.

# Appendix 02

## A02.1. Project Proposal Form

The project proposal form was prepared and duly signed from our Faculty-in-Charge Dr. Biswaranjan Swain. The same is attached at the last of this report.

## A02.2. Project Management

| # | Component | Individual Contributions in % | | | | | | Total |
|---|---|---|---|---|---|---|---|---|
| | | Abhishek | Archit | Jaydev | Debadutta | Bastav | Sunaina | |
| 1. | Planning | 16 | 16 | 16 | 16 | 16 | 16 | 100% |
| 2. | Background Research and Analysis | 16 | 16 | 16 | 16 | 16 | 16 | 100% |
| 3. | Hardware design | 23 | 23 | 23 | 10 | 10 | 10 | 100% |
| 4. | Software design | 16 | 16 | 16 | 16 | 16 | 16 | 100% |
| 5. | Testing | 23 | 23 | 23 | 10 | 10 | 10 | 100% |
| 6. | Final assembling | 23 | 23 | 23 | 10 | 10 | 10 | 100% |
| 7. | Project Report Writing | 23 | 23 | 23 | 10 | 10 | 10 | 100% |
| 8. | Presentation | 16 | 16 | 16 | 16 | 16 | 16 | 100% |
| 9 | Logistics | 16 | 16 | 16 | 16 | 16 | 16 | 100% |

## A02.3. Bill of Material

**Table 1. Component listing.**

| # | Components | Specification | Unit Cost | Quantity | Total |
|---|---|---|---|---|---|
| 1. | Raspberry Pi 4 Model 4 GB | Processor | 6149.00 | 1 | 6149.00 |
| 2. | Raspberry Pi Camera Model 3 | Imaging | 2824.00 | 1 | 2824.00 |
| 3. | Raspberry Pi Case | Encloser | 852.00 | 1 | 852.00 |
| 4. | Robotic Arm | Manipulation | 899.00 | 1 | 899.00 |
| 5. | Servo Motor | Actuator | 324.00 | 4 | 1296.00 |
| 6. | Servo Motor Driver | Controller | 550.00 | 1 | 550.00 |
| 7. | Sorting Bins | Storage | 500.00 | 3 | 1500.00 |
| 8. | Micro SD CARD 32 GB | Storage | 500.00 | 1 | 500.00 |
| 9. | Jumper Wire (M-M) | Connecter | 2.00 | 10 | 20.00 |
| 10. | Jumper Wire(M-F) | Connecter | 2.00 | 10 | 20.00 |
| | | | | Grand Total | 14,610.00 |

# Appendix 03

## A03.1. Data Sheets

### 5. Raspberry Pi Data Sheet

- **Model**: Raspberry Pi 4 Model B (or whichever model is used)
- **Processor**: Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz
- **RAM**: 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM (depending on model)
- **Storage**: microSD card slot for loading operating system and data storage
- **USB Ports**: 2 x USB 3.0, 2 x USB 2.0
- **Networking**: Gigabit Ethernet, 802.11ac Wi-Fi, Bluetooth 5.0
- **GPIO Pins**: 40 GPIO pins for interfacing with sensors, motors, etc.
- **Video Output**: 2 x micro-HDMI ports supporting up to 4K resolution
- **Power Supply**: 5V/3A USB-C power input
- **Software**: Raspbian OS (or other compatible Linux-based operating systems)

### 6. Camera Module Data Sheet:

- **Model**: Raspberry Pi Camera Module V2 (or another compatible camera)
- **Sensor**: Sony IMX219 8MP sensor
- **Resolution**: 8 megapixels (3264 x 2448)
- **Lens**: Fixed focus lens with a 62.2-degree field of view
- **Frame Rate**: Supports up to 30 fps at 1080p (Full HD)
- **Interface**: Camera Serial Interface (CSI) connector
- **Power Supply**: 3.3V through Raspberry Pi's GPIO header

### 7. Motor Driver Data Sheet:

- **Model:** L298N Dual H-Bridge Motor Driver Module
- **Input Voltage:** 5V–35V (motor power), 5V logic level
- **Output Current:** Up to 2A per channel (continuous), 3A peak
- **Control:** IN1–IN4 for direction, ENA/ENB for speed (PWM supported)
- **Compatibility:** Works with Raspberry Pi, Arduino, and other microcontrollers

### 8. Robotic Arm Data Sheet:

- **Model:** 4-DOF/6-DOF Acrylic Robotic Arm (servo-controlled
- **Material:** Lightweight acrylic or aluminum alloy (depends on model)
- **Control:** Controlled via PWM signals using servo motors
- **Degrees of Freedom:** Typically 6 DOF (flexible joint movement)
- **Power Supply:** 5V–6V (depends on connected servo motor ratings)

**9. Servo Motor Data Sheet:**

- **Model:** SG90 or MG995 (commonly used servo motors)
- **Rotation Range:** 0° to 180° (standard servo operation)
- **Operating Voltage:** 4.8V to 6V DC
- **Torque: SG90:** ~2.5 kg.cm; MG995: ~10 kg. cm (at 6V)
- **Control Signal:** PWM (Pulse Width Modulation) from microcontroller

**10. Sorting Bins Data Sheet:**

- **Material:** Plastic or acrylic (lightweight, durable)
- **Compartments:** Multiple sections based on shape/color criteria
- **Integration:** Positioned at the robotic arm's drop-off area
- **Customization:** Can be resized based on object dimensions
- **Function:** Collect sorted objects based on classification logic

**11. MicroSD Card Data Sheet:**

- **Type:** Class 10 micro SDHC or micro SDXC Card
- **Capacity:** 16GB / 32GB / 64GB (based on usage requirements)
- **Speed Class:** UHS-I, Class 10 (Read up to 80–100MB/s)
- **Interface:** SPI/SDIO (via Raspberry Pi's SD slot)
- **Usage:** Stores OS (Raspberry Pi), code, logs, and captured images