COLOR BASED PRODUCT SORTING

MINI PROJECT REPORT

Submitted by

M ASHOK	**(950621106008)**
R BALAGANESAN	**(950621106009)**
M S UMASANKAR	**(950621106049)**
S VASANTHKHAN	**(950621106052)**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

EINSTEIN COLLEGE OF ENGINEERING

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2024

BONAFIDE CERTIFICATE

Certified that this project report **“COLOR BASED PRODUCT SORTING”** is the bonafide work of **“M ASHOK (950621106008), R BALAGANESAN (950621106009), M S UMASANKAR (950621106049), S VASANTHKHAN (950621106052)”** who carried out theproject work under my supervision.

SIGNATURE	**SIGNATURE**
Mrs. G. RENGANAYAGI M.E.,	Mr. S. SANTHOSH A/P M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of ECE,	Department of ECE,
Einstein college of Engineering,	Einstein college of Engineering,
Tirunelveli-12.	Tirunelveli-12.

Submitted for the B.E Degree viva-voce Examination held on ……………….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to thank our Managing Trustee Prof. **A.MATHIVANAN BSc (agri),** Chairman Prof. **A. AMUDHAVANAN B.E M.S(USA), BL.,** and Secretary Prof. **A. EZHILVANAN MBA** for making necessary arrangement to do our project.

We would also like to thank our Principal **Dr. R. VELAYUTHAM M.E., Ph.D.,** for permitting us to do our project in our college and for his consistent encouragement.

We wish to express our sincere thanks and gratitude to **Mrs. G. RENGANAYAGI M.E.,** Head of the Department of Electronics & communication Engineering who has been the guiding force and constant source of inspiration.

We express our sincere gratitude to our project guide

Mr. S. SANTHOSH M.E., Assistant Professor, Department of Electronics and Communication Engineering who guided for the successfulcompletion of the project.

We would also thank to our entire teaching and non-teaching members who had coordinated with us and giving some suggestions for us.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

ABSTRACT

In modern manufacturing and distribution processes, efficient sorting of products plays a pivotal role in optimizing productivity and reducing operational costs. Traditional sorting methods often rely on manual labor or complex machinery, which can be time-consuming, error-prone, and costly to maintain. To address these challenges, color-based product sorting has emerged as a promising solution, leveraging advancements in computer vision, machine learning, and automation technologies.

This abstract explores the concept and implementation of color-based product sorting systems. Firstly, it elucidates the significance of efficient sorting in various industries, emphasizing the need for accurate, rapid, and cost-effective solutions. Next, it delves into the underlying principles of color-based sorting, highlighting its reliance on color sensors, cameras, and intelligent algorithms to identify and categorize products based on their color attributes.

Various methodologies are employed for color-based sorting, including traditional methods such as thresholding and clustering, as well as advanced techniques like deep learning-based approaches. These methodologies leverage color spaces, feature extraction, and classification algorithms to achieve precise sorting outcomes.

Furthermore, the abstract examines the benefits and challenges associated with color-based sorting, including its potential for enhancing throughput, accuracy, flexibility, and scalability, while addressing concerns related to lighting variations, surface textures, and color consistency. It also explores real-world applications of color-based sorting in industries such as food processing, pharmaceuticals, recycling, and logistics, showcasing its versatility and adaptability to different domains and use cases.

In conclusion, color-based product sorting represents a transformative approach to streamlining manufacturing and distribution processes, offering unparalleled speed, accuracy, and efficiency compared to traditional methods. By harnessing the power of color vision and computational intelligence, businesses can optimize their operations, reduce waste, and deliver superior quality products to consumers, thereby gaining a competitive edge in today’s dynamic marketplace.

CHAPTER 2

INTRODUCTION

In the landscape of modern manufacturing and logistics, the efficiency of product sorting processes stands as a critical factor influencing overall productivity and cost-effectiveness. Traditional methods often rely on manual labor or expensive, complex machinery, presenting challenges in terms of speed, accuracy, and scalability. However, the emergence of affordable, accessible technologies, such as the Raspberry pi, coupled with advances in image processing, has paved the way for innovative solutions to these longstanding challenges.

This introduction sets the stage for exploring the integration of Raspberry Pi and image processing techniques in color-based product sorting systems. The Raspberry Pi, a low-cost, credit-card-sized computer, has garnered widespread attention for its versatility, affordability, and ease of use. When combined with image processing algorithms, it becomes a powerful platform capable of driving intelligent, autonomous sorting systems with remarkable precision and efficiency.

Color-based sorting, as a subset of machine vision, relies on the interpretation of color information to categorize and segregate products based on predefined criteria. By leveraging the computational capabilities of the Raspberry Pi and sophisticated image processing techniques, such as color segmentation, feature extraction, and classification, it becomes possible to automate the sorting process with unprecedented accuracy and speed.

Moreover, the utilization of the Raspberry Pi offers several advantages Hover traditional sorting systems. Its compact size and low power consumption make it suitable for deployment in diverse environments, from small-scale manufacturing facilities to large warehouses. Additionally, its affordability and accessibility democratize the implementation of advanced sorting technology, enabling businesses of all sizes to benefit from enhanced efficiency and competitiveness.

Furthermore, the integration of image processing techniques amplifies the capabilities of color-based sorting systems. By analyzing and interpreting visual data in real-time, these systems can adapt to variations in product appearance, lighting conditions, and environmental factors, ensuring consistent and reliable sorting outcomes across different scenarios.

In this paper, we delve into the principles, design considerations, and implementation details of a color-based product sorting system powered by Raspberry Pi and image processing. We explore the technical components, software architecture, and workflow involved in building such a system, highlighting the synergy between hardware and software elements to achieve optimal performance.

Additionally, we discuss potential applications, benefits, and challenges associated with this approach, as well as avenues for future research and development. Through this exploration, we aim to demonstrate the transformative potential of Raspberry Pi-enabled color-based sorting systems in revolutionizing manufacturing, logistics, and supply chain operations, ushering in a new era of efficiency, flexibility, and competitiveness.

2.1 PROBLEM STATEMENT

Manual sorting of different colored objects has been tedious task in the field of agriculture, shopping malls, industries and so on. This consumes a lot of human effort which is not reliable. So this automatic sorting machine sorts the objects based on the color and greatly reduces the human effort. Also it makes it fast and error free that can be occur during the time of manual sorting. This provides huge automation in the field of agriculture, shopping malls, and industries and greatly reduces the human effort.

2.2 OBJECTIVE

1. **Efficiency:** To streamline processes by categorizing items efficiently based on color, reducing sorting time and increasing productivity.

2. **Quality Control:** To ensure consistency and quality in products by sorting based on color attributes, especially in industries like manufacturing and agriculture.

3. **Inventory Management:** To manage inventory effectively by categorizing items based on color, enabling easier tracking and restocking.

4. **Data Visualization:** To represent and analyze data visually by sorting data points based on color attributes, facilitating easier interpretation and decision-making.

5. **Automation:** To automate sorting processes using color recognition technology, reducing manual labor and potential errors

CHAPTER 3

SOFTWARE IMPLEMENTATION

This document outlines the software implementation details for a color based product sorting system powered by Raspberry Pi and image processing techniques. The system aims to automate sorting tasks by analyzing the color characteristics of products and making sorting decisions based on predefined criteria.

3.1 SOFTWARE COMPONENTS:

Raspberry Pi Operating System:

Fig : 1-symbol of Raspberry Pi

- Install and configure the Raspberry Pi operating system (e.g., Raspbian) on the Raspberry Pi board.
- Ensure proper setup of peripherals such as the camera module, display, and input/output devices.

Python Scripts:

- Develop Python scripts to implement various stages of the sorting process, leveraging the capabilities of Raspberry Pi and relevant libraries.
- Utilize the OpenCV library for image processing tasks such as image capture, preprocessing, color segmentation, and feature extraction.

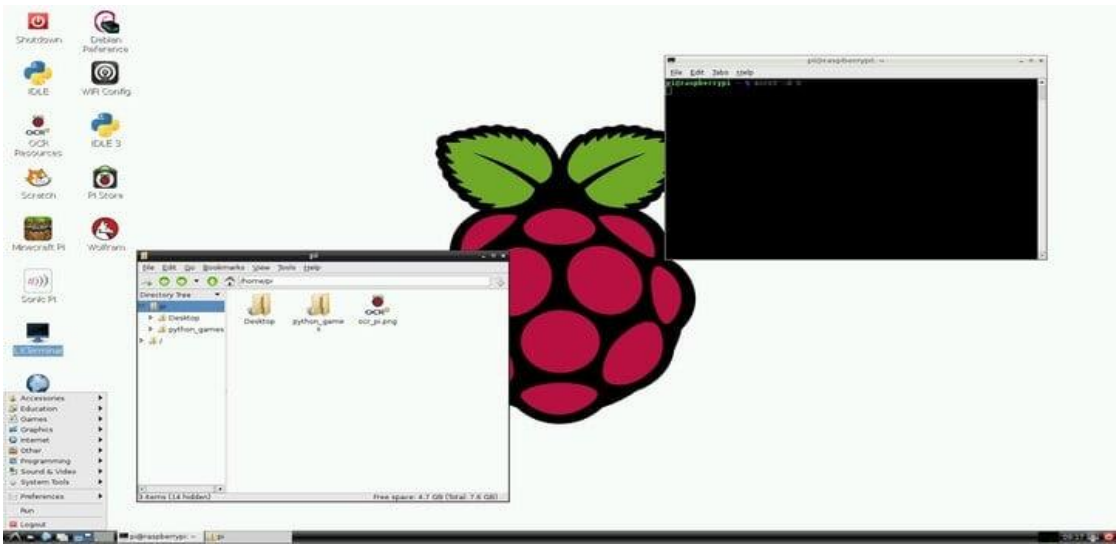


Fig : 2-Raspberry Pi OS

Image Acquisition:

- Write a Python script to interface with the camera module and capture images of products on the conveyor belt.
- Use the pi camera library to control camera settings, capture images at regular intervals, and save them for further processing.

Image Preprocessing**:**

- Implement preprocessing functions to enhance image quality and facilitate accurate color analysis.
- Apply techniques such as resizing, noise reduction (e.g., Gaussian blur), and histogram equalization to standardize image inputs.

Color Segmentation:

- Develop algorithms for color segmentation to extract regions of interest corresponding to different product categories.
- Implement thresholding methods (e.g., Otsu’s method) or clustering algorithms (e.g., K-means clustering) to separate foreground objects based on their color attributes.

Feature Extraction:

- Write functions to extract relevant color features from segmented regions, such as color histograms or color moments.
- Calculate feature descriptors to quantify the distribution of colors within each region and characterize their color properties.

Classification:
   - Train a machine learning model for product classification based on color features extracted from segmented regions.
   - Utilize libraries such as scikit-learn or TensorFlow to implement classification algorithms (e.g., Support Vector Machine, K-Nearest Neighbors) and train the model using labeled training data.

Sorting Decision:
   - Develop decision-making logic to determine the sorting destination for each product based on the classification results.
   - Map product categories to corresponding output bins or conveyor lanes and generate sorting commands accordingly.

Actuation Control:
   - Interface with actuation mechanisms (e.g., GPIO pins) to control the movement of sorting gates or conveyor diverters.
   - Write scripts to trigger actuation commands based on the sorting decisions generated by the classification module.

By implementing these software components on the Raspberry Pi platform, the color-based product sorting system can achieve automated and efficient sorting operations. The modular design allows for flexibility, scalability, and customization to suit various sorting requirements and industrial applications.

3.2 ALGORITHM

Step 1: Capture video through Pi camera.
   - This step is correct. You'll use the Pi camera to capture a video stream.

Step 2: Read the video stream into image format.
   - After capturing the video stream, each frame needs to be read and processed as an image. This step involves converting each frame of the video stream into an image format for further processing.

Step 3: Convert image format from RGB (RGB color space represented as three matrices of red, green, and blue with integer values from 0-255) to HSV (Hue-Saturation-Value) color space.
   - Correct, this step involves converting the color space from RGB to HSV.
   - Note: In the HSV color space, the components are:
      - Hue: Describes the actual color.
      - Saturation: Represents the purity or intensity of the color.

Step 4: Define the range of each color and create a corresponding mask.
- After converting to HSV, you define a range of HSV values for the color you want to detect. This range represents the acceptable variation in hue, saturation, and value for the color you're interested in.
- Then, you create a mask by thresholding the HSV image based on this defined range. The mask will highlight the areas of the image that match the specified color range.

Step 5: Apply morphological transformations to remove noise in the image.
- Morphological operations like erosion and dilation can be applied to the mask to remove small noise or unwanted artifacts and to fill in gaps in detected regions.

Step 6: Perform bitwise AND operation between the original image frame and the mask to isolate the specific color.
- After obtaining the mask, you use bitwise operations to apply the mask to the original image frame. This isolates the areas of the image that correspond to the detected color.

Step 7: Create contours for each color to display the detected areas.
- Contours are created around the isolated areas of color in the image. These contours can then be used to draw boundaries around the detected objects of the specified color.

3.3 PROGRAM

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

# Set up GPIO pins for servo motors
servo_red_pin = 18
servo_green_pin = 23
servo_blue_pin = 24

GPIO.setmode(GPIO.BCM)
GPIO.setup(servo_red_pin, GPIO.OUT)
GPIO.setup(servo_green_pin, GPIO.OUT)
GPIO.setup(servo_blue_pin, GPIO.OUT)
```

```
# Function to control servo motor
def move_servo(pin, angle):
    pwm = GPIO.PWM(pin, 50)
    pwm.start(0)
    duty_cycle = angle / 18.0 + 2.5
    pwm.ChangeDutyCycle(duty_cycle)
    time.sleep(1)
    pwm.stop()

# Main function for color detection and sorting
def color_based_sorting():
    cap = cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read()

        # Convert frame to HSV color space
        hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        # Define color ranges
        lower_red = np.array([0, 100, 100])
        upper_red = np.array([10, 255, 255])

        lower_green = np.array([40, 100, 100])
        upper_green = np.array([80, 255, 255])

        lower_blue = np.array([100, 100, 100])
        upper_blue = np.array([140, 255, 255])

        # Create masks for each color
        mask_red = cv2.inRange(hsv_frame, lower_red, upper_red)
        mask_green = cv2.inRange(hsv_frame, lower_green, upper_green)
        mask_blue = cv2.inRange(hsv_frame, lower_blue, upper_blue)

        # Find contours for each color
        contours_red,    _    =    cv2.findContours(mask_red,    cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        contours_green,   _   =   cv2.findContours(mask_green,   cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        contours_blue, _ = cv2.findContours(mask_blue, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        # If contours are found, move the corresponding servo motors
        if len(contours_red) > 0:
            move_servo(servo_red_pin, 90)
        else:
            move_servo(servo_red_pin, 0)

        if len(contours_green) > 0:
            move_servo(servo_green_pin, 90)
        else:
            move_servo(servo_green_pin, 0)

        if len(contours_blue) > 0:
            move_servo(servo_blue_pin, 90)
        else:
            move_servo(servo_blue_pin, 0)

        cv2.imshow('Frame', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
    GPIO.cleanup()
# Call the main function
color_based_sorting()
```

CHAPTER 4

HARDWARE IMPLEMENTATION

This document outlines the hardware components and setup required for implementing a color-based product sorting system using Raspberry Pi and image processing techniques. The system aims to automate sorting tasks by analyzing the color attributes of products and making sorting decisions based on predefined criteria.

4.1 HARDWARE COMPONENTS:

Raspberry Pi Board:

- Select a Raspberry Pi board (e.g., Raspberry Pi 4 Model B) as the central computing platform for the sorting system.
- Ensure sufficient processing power, memory, and connectivity options to support image processing tasks and interfacing with peripherals.

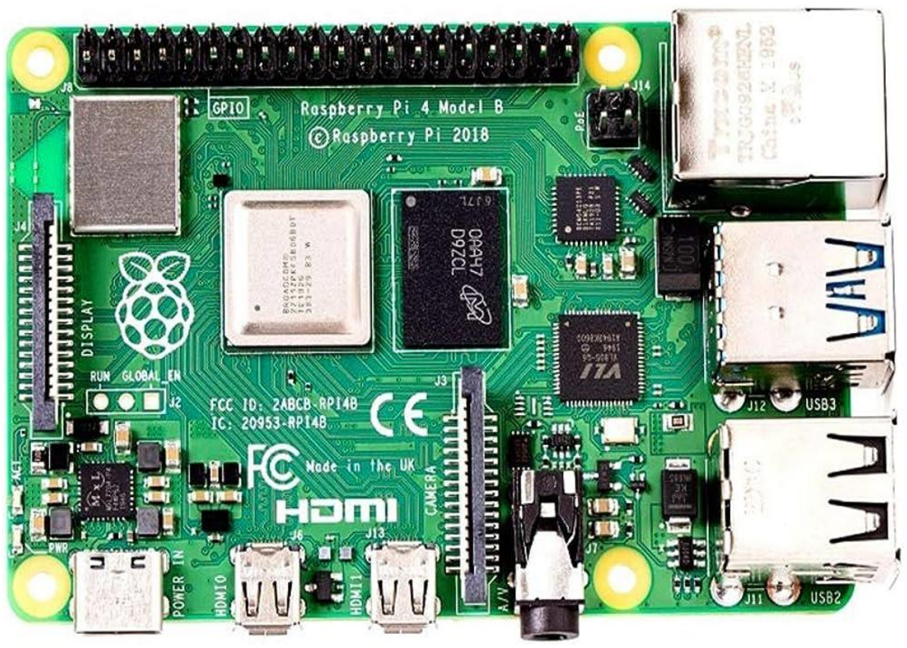


Fig : 3-Raspberry Pi

Camera Module:

- Choose a compatible camera module for Raspberry Pi capable of capturing high-resolution images with sufficient color fidelity.
- Consider factors such as resolution, frame rate, field of view, and mounting options based on the application requirements.

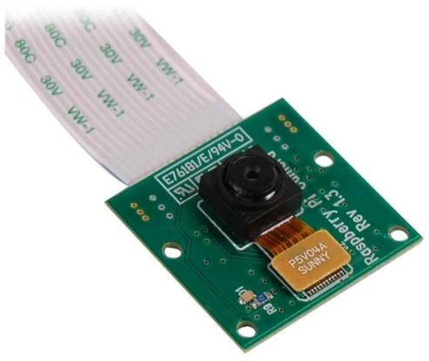


Fig : 4-Camera Module

Servo Motor:

- Use a color sensor or a camera along with image processing techniques to detect the colors of objects. You can use various libraries and frameworks depending on your hardware and programming language preference. OpenCV is a popular choice for image processing tasks.

- Once you've made the decision, actuate the servo motor to move the mechanism that sorts the objects. The servo motor's angle can be controlled to position the sorting mechanism accurately.

Fig : 5- Servo Motor

Conveyor System:

- Set up a conveyor belt system to transport products through the sorting area. Ensure smooth and consistent movement of products for accurate image capture and sorting operations.

Lighting System:

- Install adequate lighting sources to ensure uniform illumination of the sorting area.
- Choose lighting options (e.g., LED lights) that provide consistent color temperature and intensity for accurate color analysis.

Power Supply:

- Provide stable power supply to Raspberry Pi, camera module, and other peripherals to ensure uninterrupted operation.
- Use a reliable power source with sufficient capacity to meet the power requirements of all components.

Actuation Mechanisms:

- Implement actuation mechanisms (e.g., motors, solenoids) to control the movement of sorting gates or conveyor diverters.
- Choose actuators capable of precise and responsive operation to facilitate accurate sorting decisions.

Housing and Enclosure:

- Design and fabricate a housing or enclosure to protect Raspberry Pi and other sensitive components from dust, debris, and environmental factors.
- Ensure proper ventilation and heat dissipation to prevent over

heating during prolonged operation.

4.2 HARDWARE SETUP:

1. Mount the Raspberry Pi board securely in the housing or enclosure, ensuring access to GPIO pins, USB ports, and other interfaces.
2. Attach the camera module to the Raspberry Pi board and adjust its position to capture clear images of products on the conveyor belt.
3. Position the lighting sources around the sorting area to achieve uniform illumination without casting shadows or glare.
4. Integrate the conveyor system with Raspberry Pi and ensure proper alignment for smooth product movement.
5. Connect the actuation mechanisms to Raspberry Pi GPIO pins or other suitable interfaces for controlling sorting gates or diverters.
6. Establish power connections to Raspberry Pi, camera module, lighting system, and actuation mechanisms, ensuring proper voltage and current levels.

By assembling the hardware components and setting up the infrastructure described above, the color-based product sorting system can be effectively
This deployed for automating sorting tasks in various industrial applications. The robust hardware configuration, coupled with advanced image processing algorithms running on Raspberry Pi, enables accurate and efficient sorting operations, contributing to enhanced productivity and cost-effectiveness.

CHAPTER 5

PROPOSED METHOD

Outlines a method for implementing a color-based product sorting system using Raspberry Pi and image processing techniques. By analyzing product colors, the system automates sorting tasks, enhancing efficiency in industrial processes.

5.1 METHOD OVERVIEW:

Hardware Setup:

- Select Raspberry Pi (e.g., Raspberry Pi 4 Model B) for computing tasks.
- Choose a compatible camera module for Raspberry Pi.
- Set up conveyor system, lighting, and actuators for sorting.

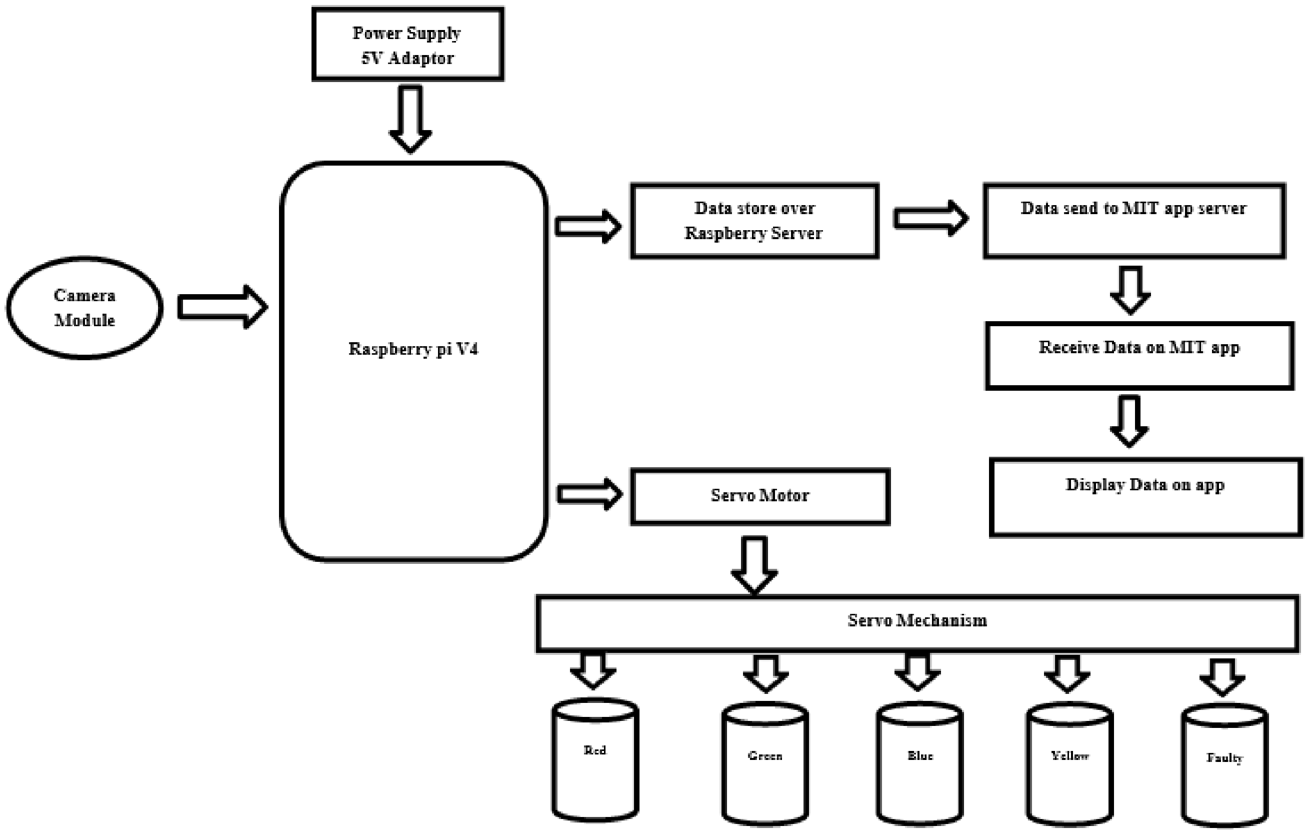


Fig : 5-Block Diagram

Software Configuration:

- Install Raspberry Pi OS and required libraries (e.g., OpenCV) for image processing.
- Develop Python scripts to capture images, process them, and control sorting mechanisms.

Image Acquisition:

- Capture images of products on the conveyor using the camera module.

Image Preprocessing:

- Apply techniques like resizing and noise reduction to enhance image quality.

Color Segmentation:

- Segment products based on color attributes using thresholding or clustering algorithms.

Feature Extraction:

- Extract color features (e.g., histograms) from segmented regions.

Classification:

- Train a machine learning model to classify products into predefined categories based on color features.

Sorting Decision:

- Determine sorting destinations for products based on classification results.

Actuation Control:

- Control sorting mechanisms (e.g., motors, solenoids) based on sorting decisions.

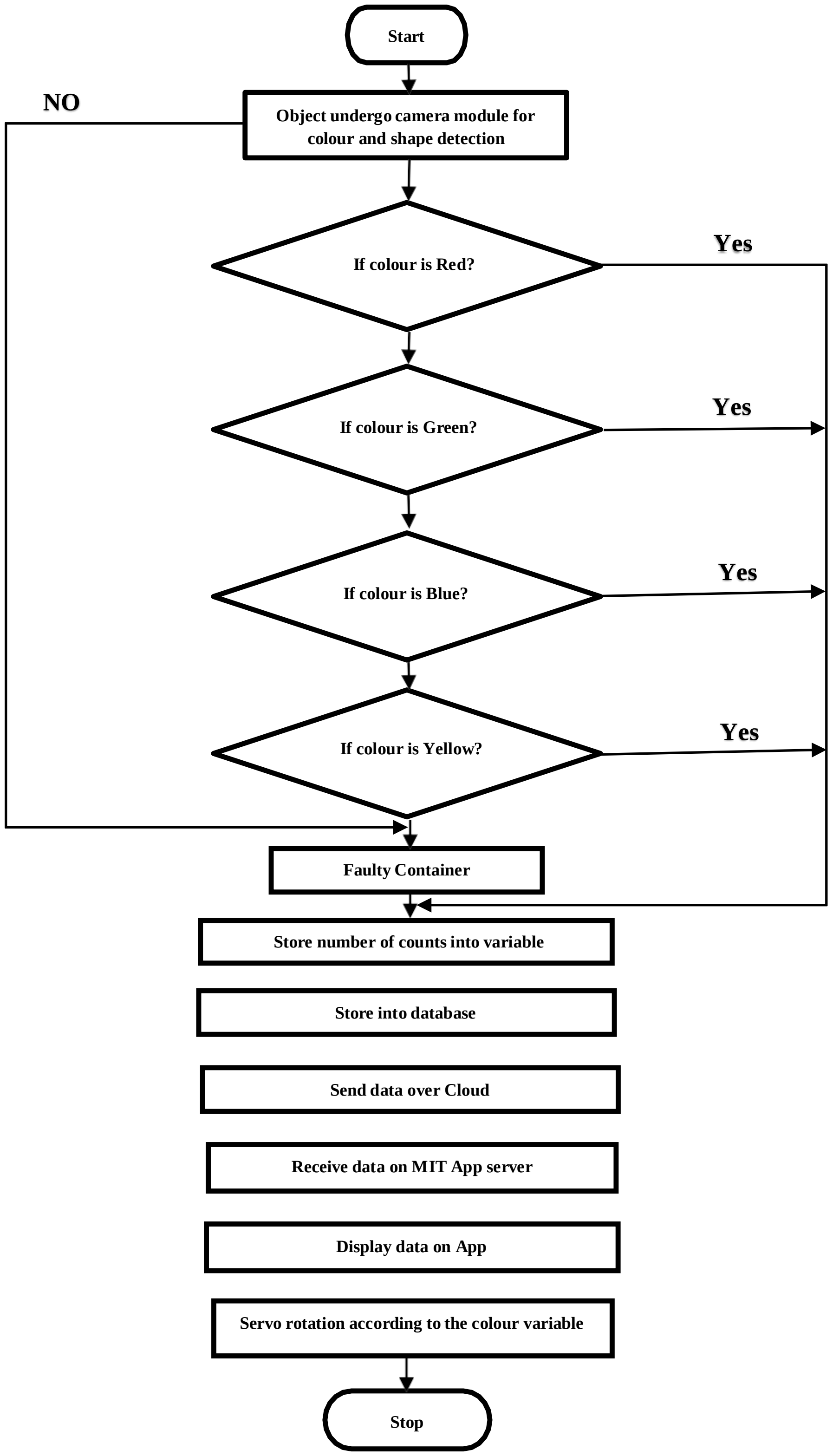
Real-time Processing:

- Ensure all processing steps occur in real-time for continuous operation.

5.2 WORKING

The mechanism is designed with the help of the components like Power supply, Camera Module, Raspberry pi V4 and Servo motor. Power supply provide electric power to the components. The color sorting mechanism makes use of a camera module to detect the color and shape of the object. Raspberry pi is used to store the data received from the camera module and to give the commands to the camera module and servo motor. Data store over Raspberry pi server is send to the MIT app server. And through the MIT app data is received from the MIT server and then data is displayed on the app. The mechanism makes use of servo motor to carry the object to the container making overall mechanism technically advanced and futuristic.

5.3 FLOW CHART



Start
NO
Object undergo camera module for colour and shape detection
If colour is Red?
Yes
If colour is Green?
Yes
If colour is Blue?
Yes
If colour is Yellow?
Yes
Faulty Container
Store number of counts into variable
Store into database
Send data over Cloud
Receive data on MIT App server
Display data on App
Servo rotation according to the colour variable
Stop

Fig : 6- Flow chart

5.4 DESIGN IMPLEMENTATION

The starting process of IOT based color sorting machine is described by the flowchart in the Fig. where the firstly object undergoes the camera module for color and shape detection. If the shape does not match to the given data to the machine, then it will go directly to the faulty container through servo motor and if the shape matches to the given data, then the color sorting begins. If the color of the object matches to the given data, the object is counted. And it stores the count into variable then it is stored in database. Data is sent over cloud then data is received on MIT app server. And then data is displayed on app. Then servo motor makes rotation according to the color variable. And if the color of the object does not match to the given data, then it will go directly into faulty container through servo motor.

Image processing steps:

1. RGB channel of image will be separated in the algorithm will the check Intensity of different channel.
2. Color label will be attached according to highest intensity in the spectrum.

The container for different color object is placed on either side of the tube and servo motor is placed in front of the tube. Depending upon the color detected flap attach on stepper motor will kick out, the last object from the row and throws into respective container. If object found to be defected it will be thrown out in the other container through the flap of stepper motor. As candies are sorted into respective of their color, they can be packed.

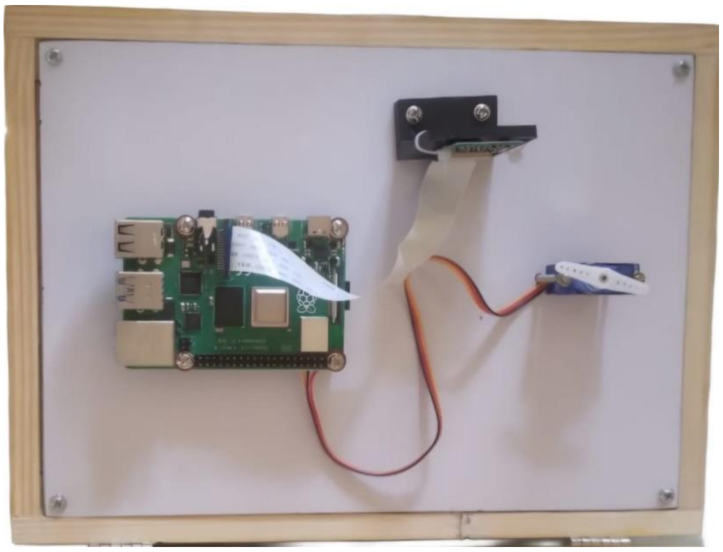
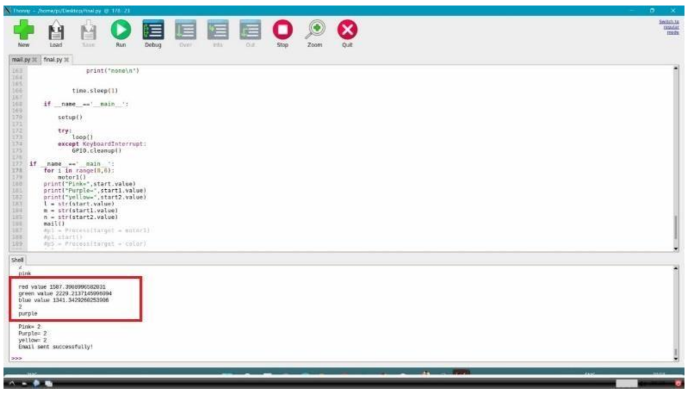


Fig : 7- Project Model

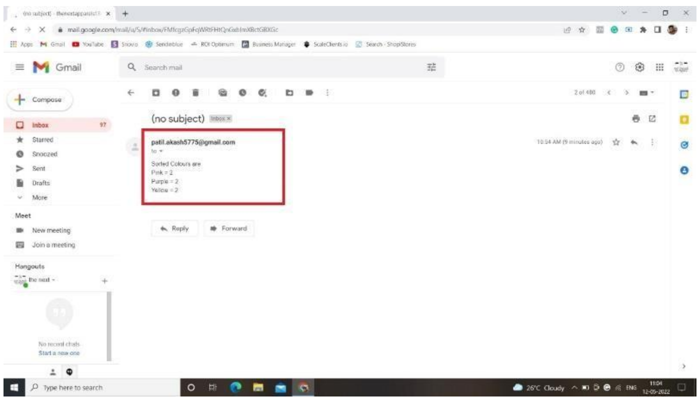
CHAPTER 6

RESULTS



```
                print("none\n")


            time.sleep(1)

    if __name__=='__main__':

        setup()

        try:
            loop()
        except KeyboardInterrupt:
            GPIO.cleanup()

if __name__=='__main__':
    for i in range(0,6):
        motor1()
    print("Pink=",start.value)
    print("Purple=",start1.value)
    print("yellow=",start2.value)
    l = str(start.value)
    m = str(start1.value)
    n = str(start2.value)
    mail()
    #p1 = Process(target = motor1)
    #p1.start()
    #p5 = Process(target = color)
```

```
2
pink

red value 1587.3968996582031
green value 2229.2137145996094
blue value 1341.3429260253906
2
purple

Pink= 2
Purple= 2
yellow= 2
Email sent successfully!
>>>
```



(no subject)

patil.akash5775@gmail.com

to

Sorted Colours are
Pink = 2
Purple = 2
Yellow = 2

Fig : 8-Output for color sorting

CHAPTER 7

CONCLUSION

In conclusion, the integration of Raspberry Pi and image processing techniques for color-based product sorting offers a robust and efficient solution for various industrial applications. By leveraging the computational power of Raspberry Pi and sophisticated image processing algorithms, the sorting system can accurately analyze the color attributes of products and make sorting decisions in real-time.

Throughout this process, we have outlined the hardware components, software architecture, and proposed method for implementing such a system. The hardware setup involves selecting suitable components such as Raspberry Pi boards, camera modules, conveyor systems, lighting, and actuation mechanisms, while ensuring proper integration and setup for seamless operation.

On the software side, we have discussed the image acquisition, preprocessing, color segmentation, feature extraction, classification, sorting decision-making, and actuation control steps. Through Python scripting and the utilization of libraries such as OpenCV and scikit-learn, the system can effectively process captured images, extract relevant color features, classify products, and control sorting mechanisms.

By following the proposed method and integrating the hardware and software components as outlined, the color-based product sorting system can achieve high levels of accuracy, efficiency, and reliability. It can adapt to various product types and sorting criteria, offering flexibility and scalability for different industrial environments.

Overall, this approach to color-based product sorting using Raspberry Pi and image processing represents a cost-effective and accessible solution for automating sorting tasks in manufacturing, logistics, and supply chain operations. It enables businesses to streamline their processes, reduce manual labor, minimize errors, and ultimately enhance productivity and competitiveness in today’s dynamic market landscape.

REFERENCES

1.Zanella, A., Bui, N., Castellani, A., Vangelista, L. and Zorzi, M., 2014. Internet of things for smart cities. IEEE Internet of Things journal, 1(1), pp.22- 32.
2.Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M., 2015. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4), pp.2347-2376
3.Shen, L.J. and Hassan, I., 2015. Design and Development of Color Sorting Robot. Journal of Engineering Science and Technology, 10, pp.71-81
4.Wongphati, M., Matsuda, Y., Osawa, H. and Imai, M., 2012, September. Where do you want to use a robotic arm? And what do you want from the robot?. In RO-MAN, 2012