LABORATORY REPORT

LAB 9: Image/Video input interfacing with microcontroller and computer-based system.

GROUP B

PROGRAMME: MECHATRONIC ENGINEERING

GROUP MEMBERS:	MATRIC NO:
1. MUHAMMAD AFIQ BIN AHMAD	2115203
2. HANIS BINTI MOHD IZANI	2020590
3. SARAH AISHAH BINTI SA'AID HAZLEY	2117600
4. NUR AYU NAZIERA BINTI ROSLI	2119202
5. NOORUL GHOUSIAH BINTI NOORDEEN SAHIB	2118298

DATE OF SUBMISSION:
Wednesday, 20th December 2023

ABSTRACT

For the purpose of learning to recognise and respond to different colors using microcontrollers, this work focuses on combining color sensors with computer vision technology. It investigates the conception, execution, and operation of an interface that permits smooth communication between computers, microcontrollers, and image/video input devices. The study is divided into two main components. The first uses an Arduino board and a colour sensor to identify the colors red, green, and blue and control an RGB LED appropriately. The RGB LED is then adjusted in accordance with the colors that are recognised by using a Pixy Camera to detect these colors. The project explores how software algorithms and hardware elements work together to quickly identify colors.

TABLE OF CONTENTS

Table of Contents

INTRODUCTION

For this experiment, we know that colour detection and processing are fundamental components of many applications, including industrial automation and robotics. The goal of this study is to find out how colour sensors and computer vision techniques may be integrated into a microcontroller-based system. In the first phase of the experiment, a TCS230 colour sensor detects the basic colours red, green, and blue. The Arduino board processes the sensor data and then controls an RGB LED to graphically represent the observed the colour. A computer vision tool that recognises colour signatures has been introduced in addition to the Pixy Camera. This advanced configuration enhances colour detection skills by dynamically varying the RGB LED with the support of a camera. The experiment shows how varied colour detection techniques may be, as well as how useful they could be in fields like automation and robotics.

PART A: Color Detection with Color Sensor

PROCEDURE

Materials And Equipment

- Arduino board
- Color sensor (TCS3200 or TCS34725)
- Jumper wires
- Breadboard
- RGB LED
- Computer with Arduino IDE and Python installed
- USB cable for Arduino

Experimental Setup

1. Hardware Setup:

- Connect the color sensor to the Arduino using jumper wires. Refer to the sensor's datasheet and Arduino's pinout diagrams for guidance1.
- If using an RGB LED, connect it to the Arduino for color display.

2. Arduino Programming:

- Write an Arduino sketch1 to interface with the color sensor.
- Read RGB color data from the sensor and convert it to a format that can be sent to the computer.
- Calibrate the sensor for accurate color readings.

3. Python Programming:

- Write a Python program to communicate with the Arduino over the serial connection using the pyserial library.
- Receive RGB color data from the Arduino.
- Interpret the received data to determine the detected color. You can modify the code shown below for your own purposes.

4. Testing and Data Collection:

- Test the system with different colored objects.
- Collect data on the detected colors, their accuracy, and how the system performs in various lighting conditions.
- Analyze the response time of the system when detecting colors.

5. Analysis:

- Evaluate the accuracy of color detection by comparing detected colors with actual colors.
- Analyze how the system performs in different lighting conditions.
- Calculate the average response time for color detection.

Methodology

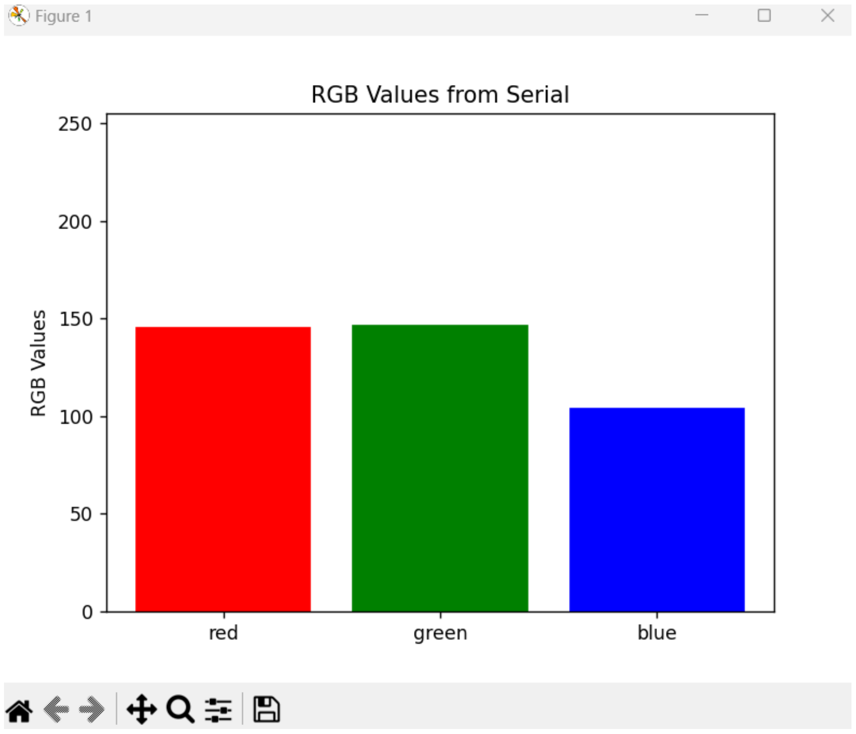
Python Code:

```
import serial
import re
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# Set your serial port name (you may need to change it based on your system)
serial_port = "COM7"  # Change this to the appropriate serial port

# Regular expression pattern to extract RGB values from the received string
pattern = re.compile(r'R:(\d+) G:(\d+) B:(\d+)')

# Initialize serial port
ser = serial.Serial(serial_port, baudrate=9600, timeout=1)

# Initialize plot
fig, ax = plt.subplots()
colors = ['red', 'green', 'blue']
bars = ax.bar(colors, [0, 0, 0], color=colors)

ax.set_ylim(0, 255)
ax.set_ylabel('RGB Values')
ax.set_title('RGB Values from Serial')

# Function to update plot data
def update(frame):
    # Send the AT+COLOR command every 1 seconds
    if frame % 20 == 0:
        ser.write(b'AT+COLOR\r\n')

    # Check if there is data available to read
    if ser.in_waiting > 0:
        # Read and decode the received data
        data = ser.readline().decode('utf-8').strip()
        print(data)  # Print received data for debugging

        # Extract RGB values using regular expression
        match = pattern.search(data)
        if match:
            rgb_values = [int(match.group(i)) for i in range(1, 4)]

            # Update bar chart data and color
            for bar, color, value in zip(bars, colors, rgb_values):
                bar.set_height(value)
                bar.set_color(color)

            # Redraw the bar chart
            fig.canvas.draw()

# Create an animation
ani = animation.FuncAnimation(fig, update, frames=100, blit=False)
```

```
# Show the plot
plt.show()

# Close the serial port when done
ser.close()
```

RESULTS

Output:



Video:

https://youtube.com/shorts/J2JrhF2if_k?si=hxNkvfGwVRW-wNyk

DISCUSSION

The Color Detection with Color Sensor experiment explores the principles of color sensing technology using a color sensor module. The objective is to understand how color sensors can be employed to identify and distinguish different colors. Arduino is used to interface with the color sensor and implement logic to interpret the color data received from the sensor. It is important to calibrate the color sensor so that it recognizes a baseline set of colors. Once it is okay to conduct, the sensor presents various colored objects to the sensor and observes the output. The color sensor's limitations, such as sensitivity to ambient light conditions and potential challenges in accurately distinguishing similar colors. The Color Detection with Color Sensor lab experiment provides hands-on experience in working with color sensing technology.

The accuracy of color detection is evaluated by comparing the colors detected by the color sensor with the actual colors of objects presented to it. By systematically comparing detected colors with actual colors, we can assess the accuracy of the color detection system and gain insights into the performance of the color sensor under various conditions. This evaluation process is essential for fine-tuning the system and understanding its limitations. Next, analyzing the performance of a color detection system in different lighting conditions is crucial, as variations in ambient light can significantly impact the accuracy of color sensing. Performing this experiment with Ideal Lighting Conditions is the best option to get an accurate outcome. Ideal Lighting Conditions means in optimal lighting conditions, where the illumination is consistent and well-distributed, the color sensor is likely to perform accurately. This serves as a baseline for comparison. In addition to the Ideal Lighting, we cannot do this experiment if there is the presence of shadows and glare. Shadows cast by objects and glare from reflective surfaces can introduce challenges. The color sensor may interpret shadows or glare as changes in color, affecting the accuracy of detection.

By systematically evaluating the color detection system under different lighting conditions, we can identify its strengths and limitations. However, the response time may vary based on the specific color detection algorithm, the efficiency of the microcontroller, and the complexity of the color sensor. The goal is to obtain a reliable and representative measure of the system's average response time under the experimental conditions. Furthermore, our experiment on Color Detection with Color Sensor is successfully done. Based on the bar graph, it is shown that the color sensor is able to detect all the color clearly.

PART B: Colour Detection with Pixy Camera

PROCEDURE

Materials And Equipment

- Arduino
- Pixy camera
- Mini B USB Cable
- Power for arduino and pixy camera (laptop)

Experimental Setup

1. IDC Pixy Cable is connected into Arduino's ICSP pins.
2. Mini B USB Cable is connected to Pixy.
3. Both Arduino and Pixy cables are connected to 2.0 USB Ports direct on computer so that they are powered by USB's force.

Methodology

```
#include <Pixy.h>

Pixy pixy;

void setup() {
  Serial.begin(9600);
  pixy.init();
}

void loop() {
  int blocks = pixy.getBlocks();

  if (blocks) {
    for (int i = 0; i < blocks; i++) {
      if (pixy.blocks[i].signature == 1) {
        Serial.println("blue object detected");
      } else if (pixy.blocks[i].signature == 2) {
        Serial.println("Green object detected");
      } else if (pixy.blocks[i].signature == 3) {
        Serial.println("red object detected");
      }
    }
  }
}
```

Procedure For Pixy Camera

1. PixyMon is installed.
2. Pixy is configured in PixyMon :

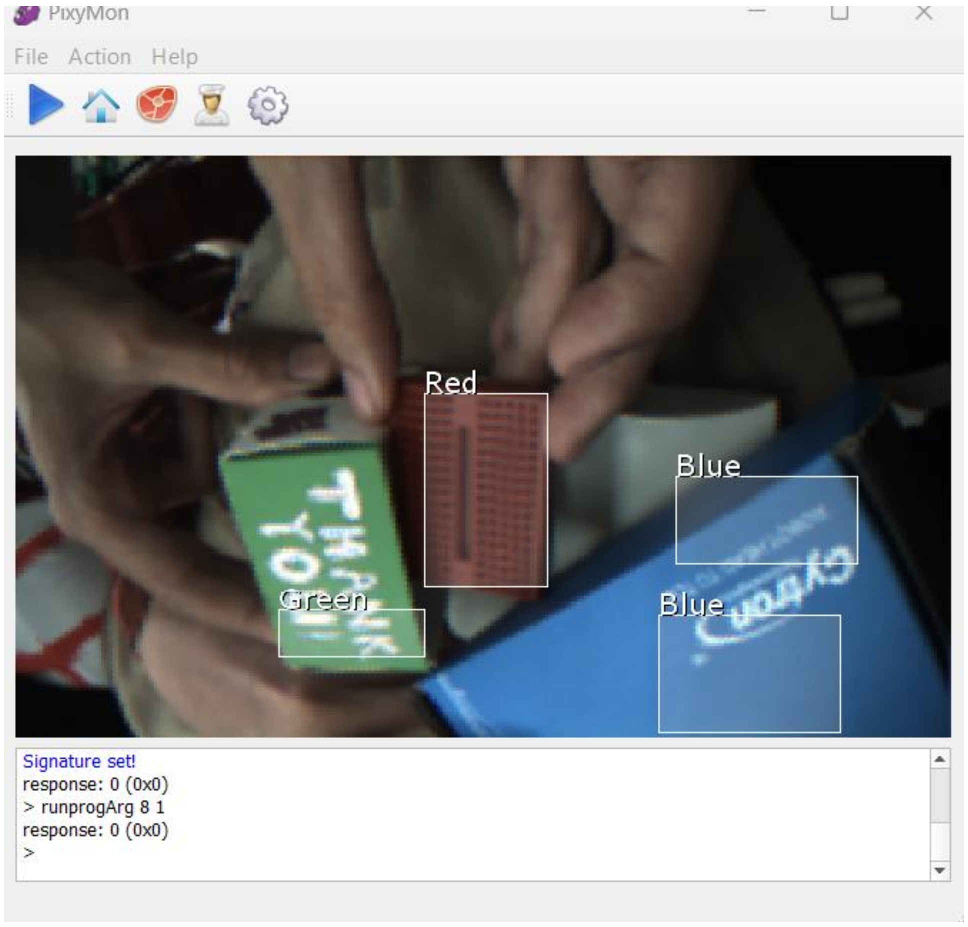
- Icon Config > Pixy Parameters > Signature Labels and write down these three colors: green, blue, red > click ok.
- Config > Pixy Parameters tab > choose SPI Interface (Data out port: Arduino ICSP SPI; I2C address: 0x54; UART baud rate: 19200).

3. Teaching Pixy: Three objects are grabbed: one red, another yellow and another orange and they are positioned in front of the camera; make sure to get the best focus and light.
4. Go to Menu Action > Set Signature #1 > a region is selected from the green figure > the right mouse button is released making a rectangular shape inside it (now green must be shown on the screen)
5. The previous procedures are repeated for the blue and red (choose Set Signature #2 for blue & Set Signature #3 for red)
6. Finally go to Config > PixyMon Parameters (saved on computer) > General tab > Cooked render mode > boxes only; (to get rid of the pixels and emphasize the edges of the captured figure)
7. Go to Config > PixyMon Parameters (saved on Pixy) > Signature Tuning (adjust the all the ranges accordingly); (to make the edges of the captured figure clearer)

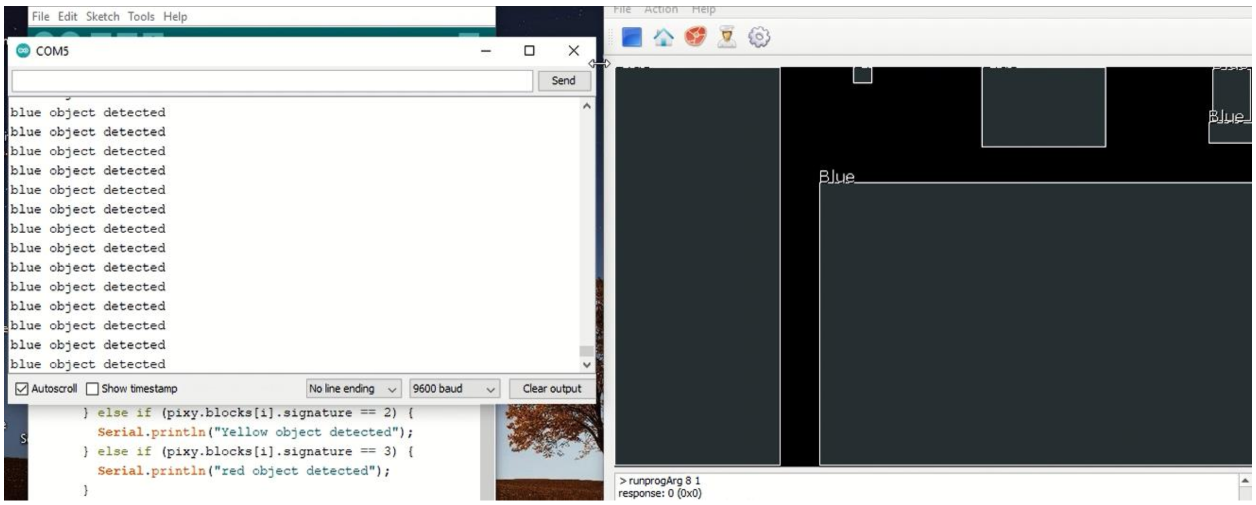
Procedure For Arduino Ide

1. Open Arduino IDE, Go to Sketch > Include Library > Add .ZIP library of pixy.
2. Arduino IDE is loaded, File > Examples > Pixy > hello_world (example) and maintain the baud rate to 9600 as the default value.
3. Arduino hello_world sketch with PixyMon is run together, Arduino's serial monitor is fire side-by-side with PixyMon. (Display Cooked/Processed/Video) button must be unable)

RESULTS



```
Signature set!
response: 0 (0x0)
> runprogArg 8 1
response: 0 (0x0)
>
```



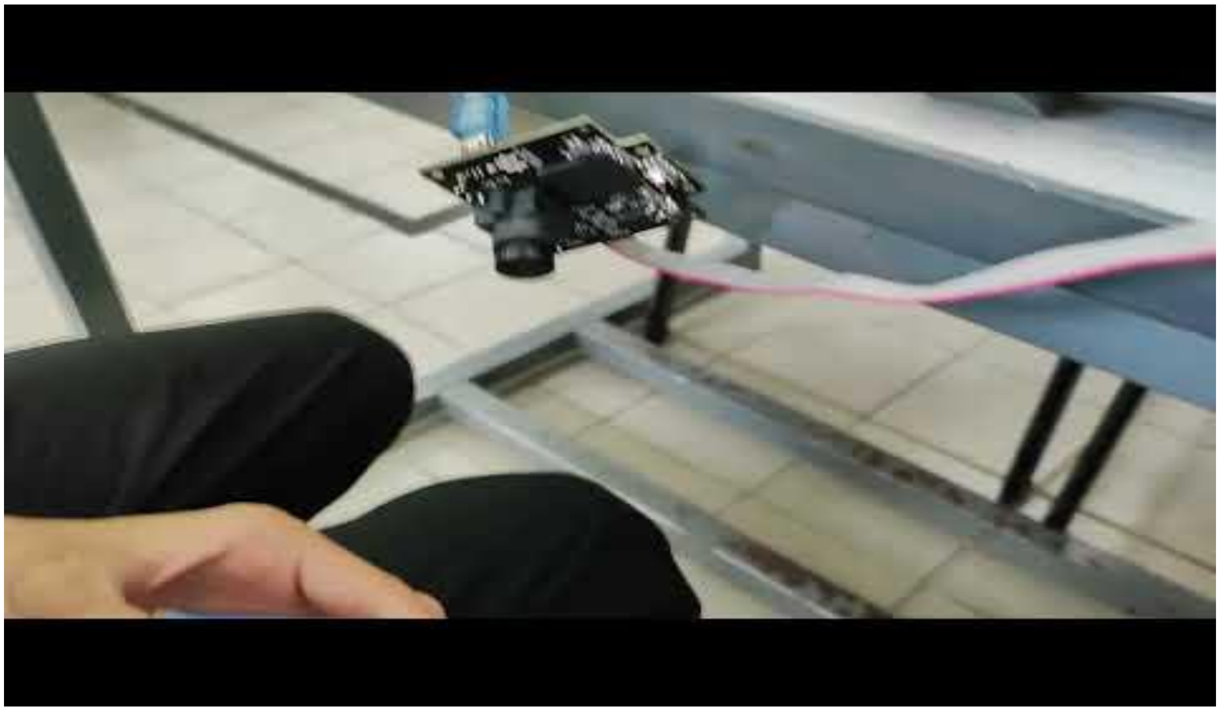
```
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
blue object detected
```

☑ Autoscroll ☐ Show timestamp No line ending 9600 baud Clear output

```
    } else if (pixy.blocks[i].signature == 2) {
      Serial.println("Yellow object detected");
    } else if (pixy.blocks[i].signature == 3) {
      Serial.println("red object detected");
    }
```

```
> runprogArg 8 1
response: 0 (0x0)
```

https://youtu.be/AxjTH0YHwV8?si=4T6zDn9jhNxHqNX6



DISCUSSION

The pixy camera is successfully able to detect and differentiate between colors in our experiment. The three colors detected are green, blue, and red. From our observations the results are accurate. The correct output is displayed on the serial monitor of Arduino IDE depending on the colour detected by the pixy camera. The rate of color detection and its output display to the serial monitor is almost instantaneous.

In the configuration of the PixyMon software, we tuned the range of color signatures so that the detection would be accurate. In this case, the range of blue is decreased and the range of red and green is increased. This is because the ambient environment displayed by the pixy camera has a blue hue.

When tested in different lighting conditions, the result are as follows: The green, red, and blue can be seen in bright lighting, normal lighting and dimmed lighting. However, in dark room condition, the pixy camera could not detect any color. This shows the quality of the Pixy camera and the flexibility of the system, making it suitable for a wide range of applications.

PART C: Color Detection with USB Camera

PROCEDURE

Materials And Equipment

- USB Camera
- Computer with Python installed
- USB cable

Experimental Setup

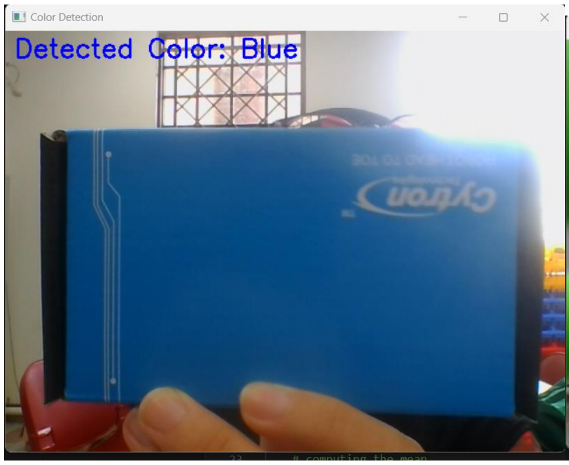
1. Connect the USB camera to the computer using a USB cable.
2. Write a Python program to capture video from the USB camera.
3. Python Code:

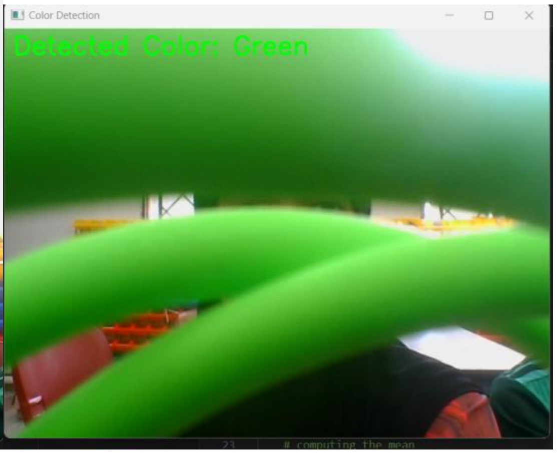
```
# importing required libraries
import cv2
import numpy as np

# taking the input from webcam
vid = cv2.VideoCapture(0)

# running while loop just to make sure that
# our program keeps running until we stop it
while True:

    # capturing the current frame
    _, frame = vid.read()

    # displaying the current frame
    cv2.imshow("frame", frame)

    # setting values for base colors
    b = frame[:, :, :1]
    g = frame[:, :, 1:2]
    r = frame[:, :, 2:]

    # computing the mean
    b_mean = np.mean(b)
    g_mean = np.mean(g)
    r_mean = np.mean(r)

    # displaying the most prominent color
    if b_mean > g_mean and b_mean > r_mean:
        color_name = "Blue"
        color = (255, 0, 0)  # BGR values for blue
    elif g_mean > r_mean and g_mean > b_mean:
        color_name = "Green"
        color = (0, 255, 0)  # BGR values for green
    elif r_mean > g_mean and r_mean > b_mean:
        color_name = "Red"
        color = (0, 0, 255)  # BGR values for red
    elif r_mean > g_mean and b_mean > g_mean:
        color_name = "Purple"
        color = (128, 0, 128)  # BGR values for purple
    elif g_mean > r_mean and b_mean > r_mean:
        color_name = "Yellow"
        color = (0, 255, 255)  # BGR values for yellow
    else:
        color_name = "Unknown"
        color = (0, 0, 0)  # Black for unknown color

    # display the detected color
    cv2.putText(frame, f"Detected Color: {color_name}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
1, color, 2)

    # displaying the most prominent color
    cv2.imshow("Color Detection", frame)

    # break the loop if the 'q' key is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# release the VideoCapture object and close all windows
vid.release()
cv2.destroyAllWindows()
```

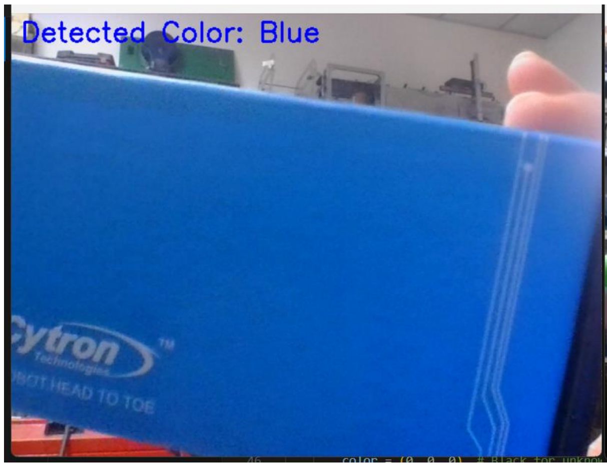
Methodology

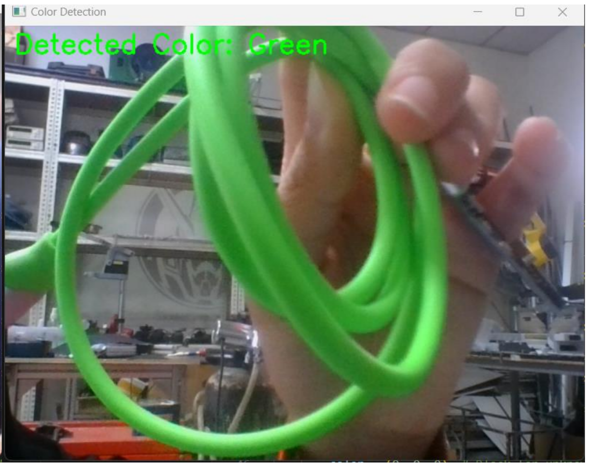
1. The USB camera is connected to the computer by using a USB cable.
2. The Python code is compiled and run to build a colour detection system.
3. The Color detection system is tested with different coloured objects.
4. The data on the detected colors, their accuracy, and how the system performs in various lighting conditions are recorded.

RESULTS









DISCUSSION

Color Detection using a USB camera is successfully done. The USB camera is able to detect the colors clearly with different light intensity. The experiment demonstrates a practical application of color detection using a USB camera and Python programming. It involves the use of a USB camera connected to a computer, where a Python program captures video from the camera and performs color detection on the frames. The Python code utilizes the OpenCV library to capture video from the USB camera. Within a continuous loop, the program reads frames, displays them, and computes the mean color values for the base colors like red, green, and blue. The most prominent color is then determined based on the highest mean value, and its name is displayed on the video frame. The color detection algorithm identifies primary colors such as blue, green, and red. The program also can handle cases where it can detect the color of objects in various lighting conditions. Furthermore, the accuracy of the color detection system is excellent as the detected colors are almost the same as the actual color of the objects.

CONCLUSION

In conclusion, the experiments provided a comprehensive exploration of color detection technologies, covering both hardware (color sensor, Pixy Camera) and software (Arduino, PixyMon, Python). With this knowledge, the colour detection technology can be used in various fields like automation, printing and many more. The combination of hands-on experimentation and programming skills provides learning experience in color sensing technology.

RECOMMENDATIONS

With the knowledge gained in colour detection technology, we could explore advanced colour detection algorithms. By exploring and implementing the algorithms, we could enhance accuracy and versatility in colour sensing. In return, we could detect a wide range of colours.

REFERENCES

Jungletronics (2019, March 27). *Arduino Meets Pixy*. Retrieved December 18, 2023, from https://medium.com/jungletronics/arduino-meets-pixy-36138c474912

APPENDICES

ACKNOWLEDGEMENT

We would like to express our gratitude to Dr. Wahju Sediono, one of the lecturers of this course, mechatronic system integration 1 (MCTA 3203) for his guidance while the experiment is being carried out.

STUDENT'S DECLARATION

This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or person.

We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual's contributor to the report.

We, therefore, agreed unanimously that this report shall be submitted for marking and this final printed report have been verified by us.

Signature: Ghousiah Name: Noorul Ghousiah Binti Noordeen Sahib Matric Number: 2118298 Contribution: Discussion part B, Procedure part C	Read	✓
	Understand	✓
	Agree	✓
Signature: hanis Name: hanis Matric Number: 2020590 Contribution:	Read	✓
	Understand	✓
	Agree	✓
Signature: Afiq Name: Muhammad Afiq bin Ahmad Matric Number: 2115203 Contribution: abstract, introduction, procedure part A	Read	✓
	Understand	✓
	Agree	✓
Signature: ayunaziera Name: Nur Ayu Naziera Binti Rosli Matric Number: 2119202 Contribution: discussion partA, procedure partB, discussion partC	Read	✓
	Understand	✓
	Agree	✓
Signature:Sarah Name: Sarah Aishah binti Sa'aid Hazley Matric Number: 2117600 Contribution: Conclusion, Recommendation	Read	✓
	Understand	✓
	Agree	✓