LABORATORY REPORT

LAB 9: Image/Video Input Interfacing With Microcontroller and Computer Based System

GROUP H

PROGRAMME: MECHATRONICS ENGINEERING

GROUP MEMBERS:	MATRIC NO:
1. SYEDA SAMIA	2123536
2. ATHIRAH HAZIQAH BINTI BADERULHISHAM	2210240
3. SITI HEIDI AMIRA BINTI AZRUL	2111970

DATE OF SUBMISSION:
Wednesday, 22nd March 2024

ABSTRACT

It is expected in this report for us to be able to design a color detection system that would detect three different colored objects using Arduino, Python, and either a color sensor or a USB camera. It includes setting up the hardware which would be the color sensor and USB camera and then programming using Python and Arduino. Additionally, we will test the color detection accuracy and performance and response time in various situations.

Table of Contents

INTRODUCTION

For today's experiment we will work with a color sensor, specifically the TCS230, and a Pixy camera and the laptop webcam. First and foremost, the TCS3200 color sensor is specially useful for color recognition projects such as color matching, color sorting, test strip reading and much more. The TCS3200 has an array of photodiodes with 4 different filters. A photodiode is simply a semiconductor device that converts light into current. The sensor has a current-to-frequency converter that converts the photodiodes' readings into a square wave with a frequency that is proportional to the light intensity of the chosen color. This frequency is then read by the Arduino. Furthermore, the Pixy camera is a camera that is able to learn to detect objects that you teach it and it connects to Arduino, Raspberry Pi, BeagleBone and similar controllers. It is C/C and Python supported and its configuration utility runs on Windows, MacOS and Linux. We expect for the sensor and cameras to detect objects and state their colors, it is very likely that there will be a difference in the performance of the different hardware.

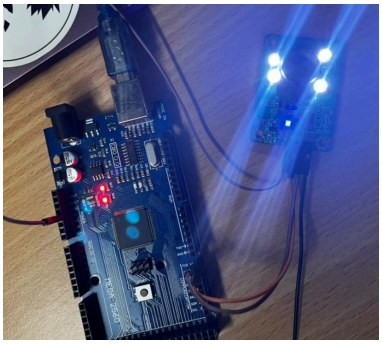
PROCEDURE

Materials And Equipment:

- Arduino board
- Color sensor TCS230
- Jumper wires
- Breadboard
- Computer with Arduino IDE and Python installed
- USB cable for Arduino
- Arduino board
- Laptop with a camera
- Pixy camera
- Camera cable

Experimental Setup for Color Sensor:

1. Connect the color sensor to the Arduino by connecting the pins GND→GND, VCC→+5V, RX→0, and TX→1 by jumper wires.
2. connect the Arduino to the computer by a USB cable.



Methodology for Color Sensor:

1. Write an Arduino sketch to interface with the color sensor.
2. Read RGB color data from the sensor and convert it to a format that can be sent to the computer.
3. Calibrate the sensor for accurate color readings.
4. Write a Python program to communicate with the Arduino over the serial connection using the pyserial library.
5. Receive RGB color data from the Arduino.
6. Interpret the received data to determine the detected color.
7. Test the system with different colored objects.
8. Collect data on the detected colors, their accuracy, and how the system performs in various lighting conditions.
9. Analyze the response time of the system when detecting colors.

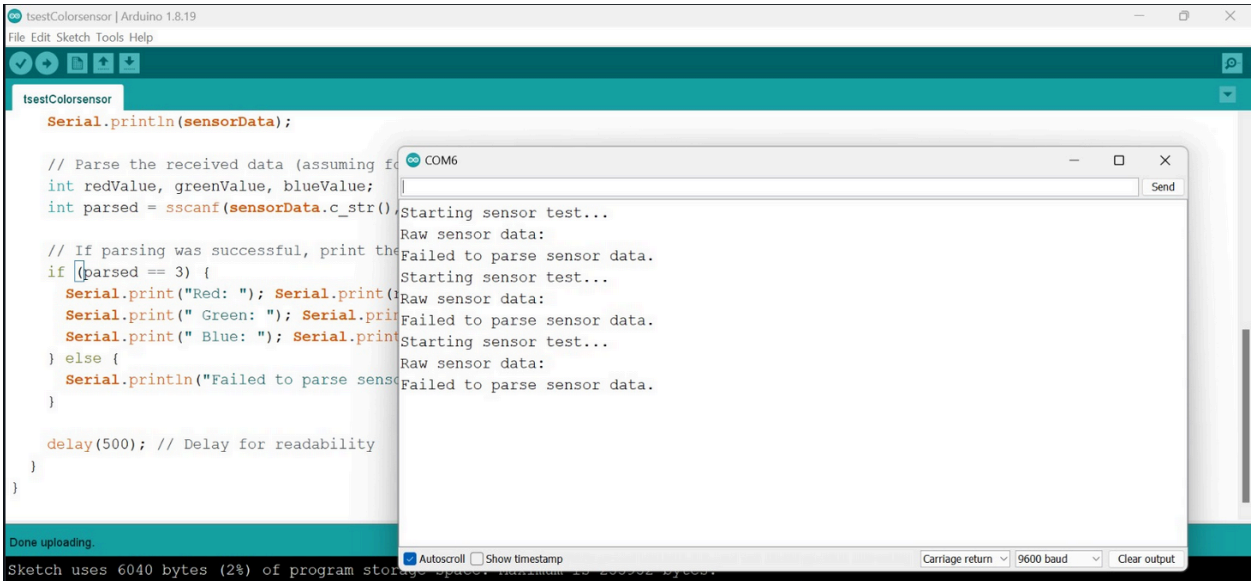
Python code:

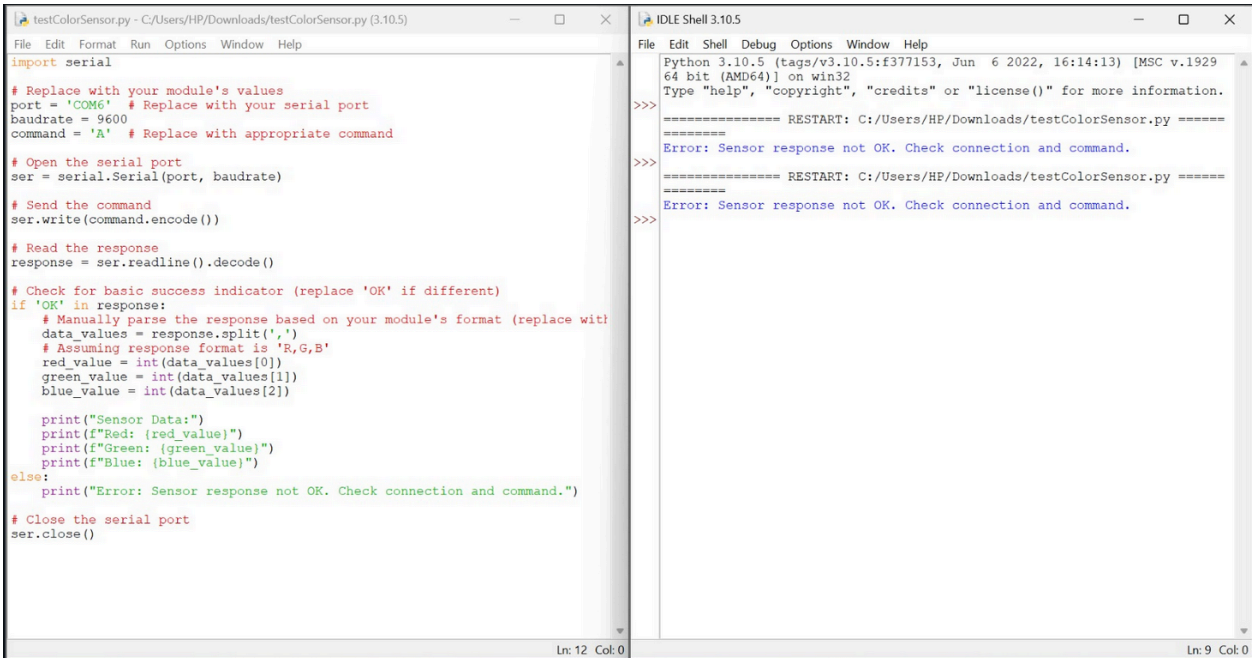
```
import matplotlib.pyplot as plt
import numpy as np
def create_channel(color_val):
	return np.ones((10, 10, 3))*color_val



def display_color(r, g, b):
	color = [r/255, g/255, b/255]
	r1 = create_channel([r/255, 0, 0])
	g1 = create_channel([0, g/255, 0])
	b1 = create_channel([0, 0, b/255])
	rgbchannel = [r1, g1, b1]
```

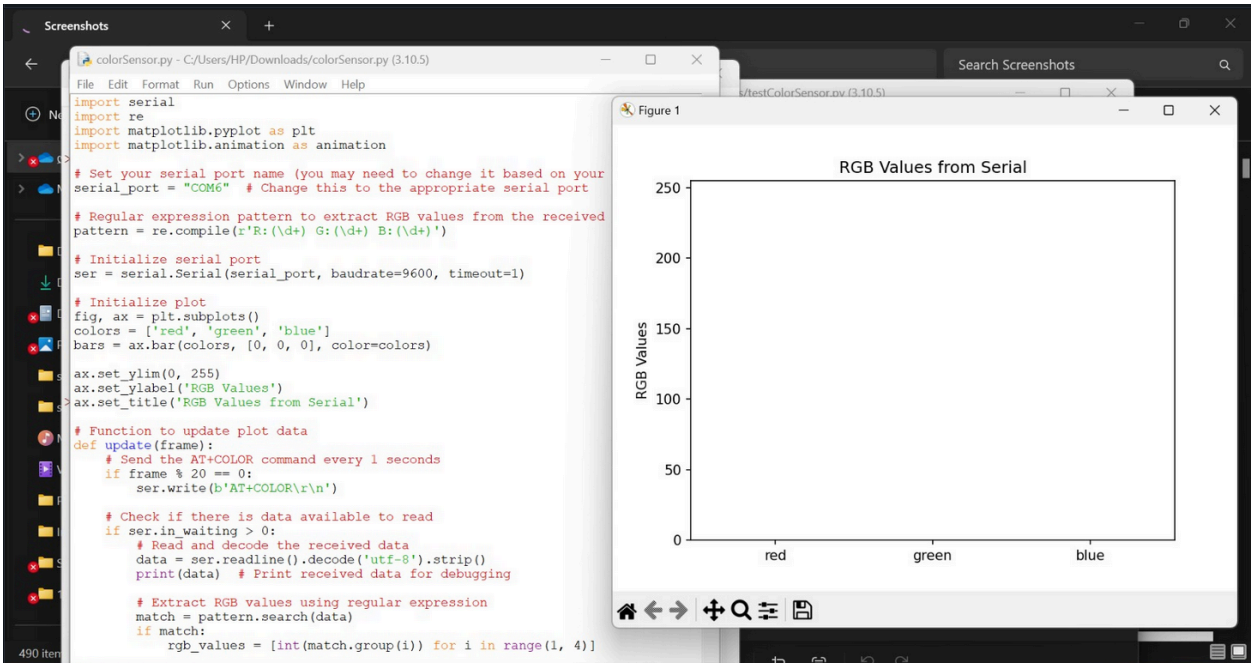
```
        true_color = np.ones((10, 30, 3))*color
        final_display = np.vstack([np.hstack(rgbchannel), true_color])
        plt.imshow(final_display)
        plt.axis('off')
        plt.show()


random_r = np.random.randint(0, 256)
random_g = np.random.randint(0, 256)
random_b = np.random.randint(0, 256)
display_color(random_r, random_g, random_b)
```

Data Collection:

For Color Sensor:







Experimental Setup for Laptop Webcam:

1. Connect the USB camera to the computer using a USB cable.

Methodology for Laptop Webcam:

1. Write a Python program to capture video from the Pixy camera using the OpenCV library.
2. Implement color detection logic using HSV color space.

3. Display the video feed and detect colors on the computer screen.
4. Test the system with different colored objects.
5. Collect data on the detected colors, their accuracy, and how the system performs in various lighting conditions.

Python Code One:

```python
import cv2
import numpy as np

def Color_detection(frame):
    # Convert the frame to HSV color space
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Define the color ranges for red, green, and blue in HSV color space
    red_lower = np.array([0, 120, 70])
    red_upper = np.array([10, 255, 255])
    green_lower = np.array([36, 100, 100])
    green_upper = np.array([86, 255, 255])
    blue_lower = np.array([94, 80, 2])
    blue_upper = np.array([126, 255, 255])

    # Create masks for each color
    red_mask = cv2.inRange(hsv, red_lower, red_upper)
    green_mask = cv2.inRange(hsv, green_lower, green_upper)
    blue_mask = cv2.inRange(hsv, blue_lower, blue_upper)

    # Find contours for each mask
    contours_red, _ = cv2.findContours(red_mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contours_green, _ = cv2.findContours(green_mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contours_blue, _ = cv2.findContours(blue_mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    return contours_red, contours_green, contours_blue

def draw_detected_objects(frame, contours, color_name, color_bgr):
    # Sort contours by area (largest first) and limit to the top 3
    contours = sorted(contours, key=cv2.contourArea, reverse=True)[:1]

    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        size = max(w, h)
        cv2.rectangle(frame, (x, y), (x + size, y + size), color_bgr, 2)
        cv2.putText(frame, color_name, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, color_bgr, 2)

def main():
    vid = cv2.VideoCapture(0)

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

        contours_red, contours_green, contours_blue = Color_detection(frame)

        # Draw the detected red objects
        draw_detected_objects(frame, contours_red, "Red", (0, 0, 255))
        # Draw the detected green objects
        draw_detected_objects(frame, contours_green, "Green", (0, 255, 0))
        # Draw the detected blue objects
        draw_detected_objects(frame, contours_blue, "Blue", (255, 0, 0))

        cv2.imshow("Frame", frame)
```

```
56
57          if cv2.waitKey(1) & 0xFF == ord('q'):
58              break
59
60      vid.release()
61      cv2.destroyAllWindows()
62
63  if __name__ == "__main__":
64      main()
65
```

Python Code Two:

```
1   import cv2
2   import numpy as np
3
4
5   def Color_detection(frame):
6       b, g, r = cv2.split(frame)
7
8       bm = np.mean(b)
9       gm = np.mean(g)
10      rm = np.mean(r)
11
12      if max(bm, gm, rm) == bm:
13          return "Blue"
14      elif max(bm, gm, rm) == gm:
15          return "Green"
16      else:
17          return "Red"
18
19
20  def main():
21      vid = cv2.VideoCapture(0)
22
23      while True:
24          _, frame = vid.read()
25
26          color = Color_detection(frame)
27
28          cv2.putText(frame, "Detected Color: " + color, (10, 30),
29                      cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
30
31          cv2.imshow("Frame", frame)
32
33          if cv2.waitKey(1) & 0xFF == ord('q'):
34              break
35
36      vid.release()
37      cv2.destroyAllWindows()
38
39
40  if __name__ == "__main__":
41      main()
```

Data Collection:

For Python Code One:

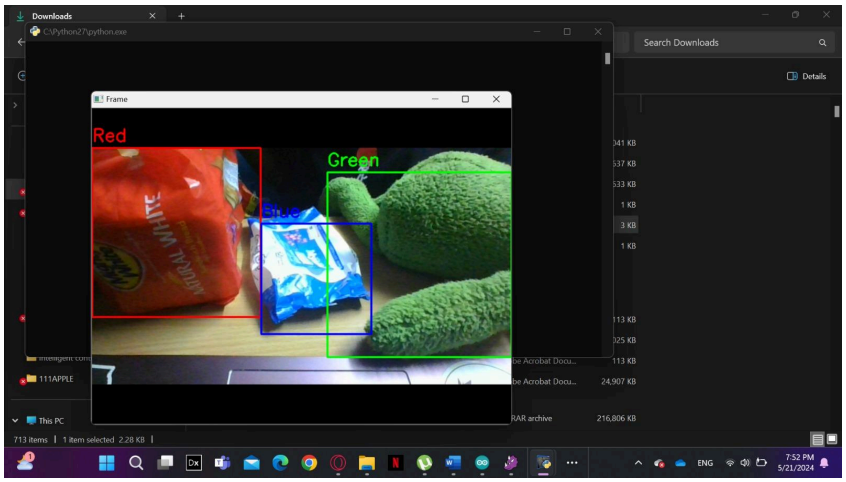


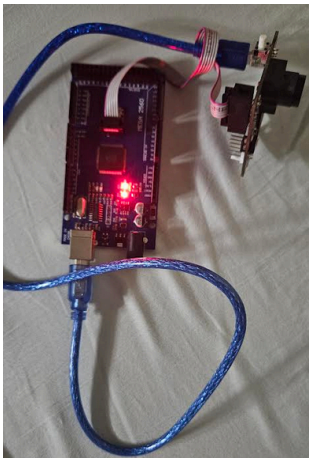
Fig. 1

For Python Code Two:

Fig. 2

Experimental Setup for Pixy Camera:

1. Connect the Pixy camera to the ICSB Arduino Mega pins.
2. Power the Arduino via USB.



Methodology for Pixy Camera:

1. Calibrate the Pixy camera for the three different colored objects.
2. Open the PixyMon software on your computer, connect the Pixy camera, and configure color signatures for each of the three objects.
3. Download and install the Pixy library for Arduino.
4. In the Arduino IDE, go to Sketch -> Include Library -> Pixy.
5. Write code to detect objects and upload it.
6. Open the Serial Monitor in the Arduino IDE (Tools -> Serial Monitor) to view the output and debug any issues.
7. If needed, adjust color signatures and threshold values in the PixyMon software and the Arduino code to improve object detection.

Arduino code:

```
#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("Red object detected");
      } else if (pixy.blocks[i].signature == 2) {
```

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

Data Collection:

For Pixy Camera:

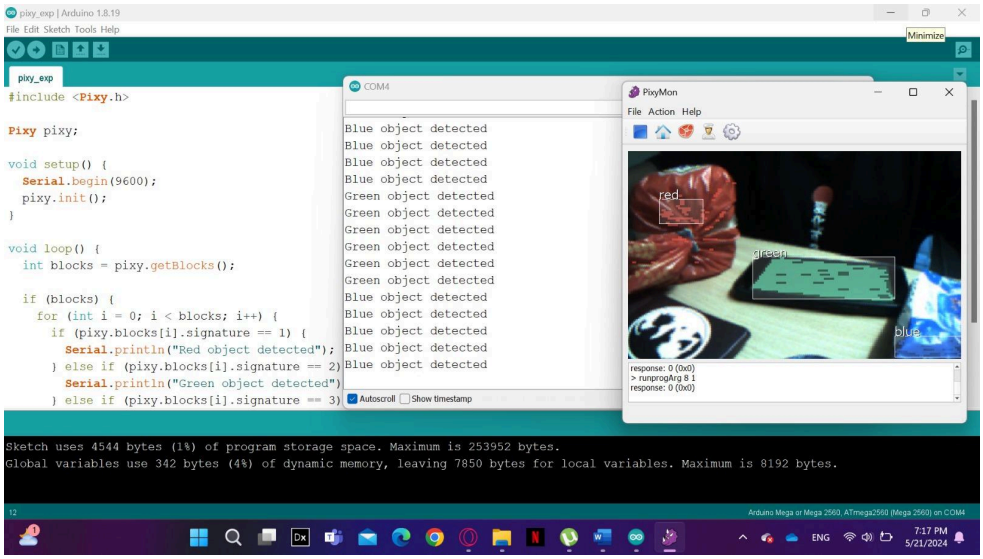


Fig. 3

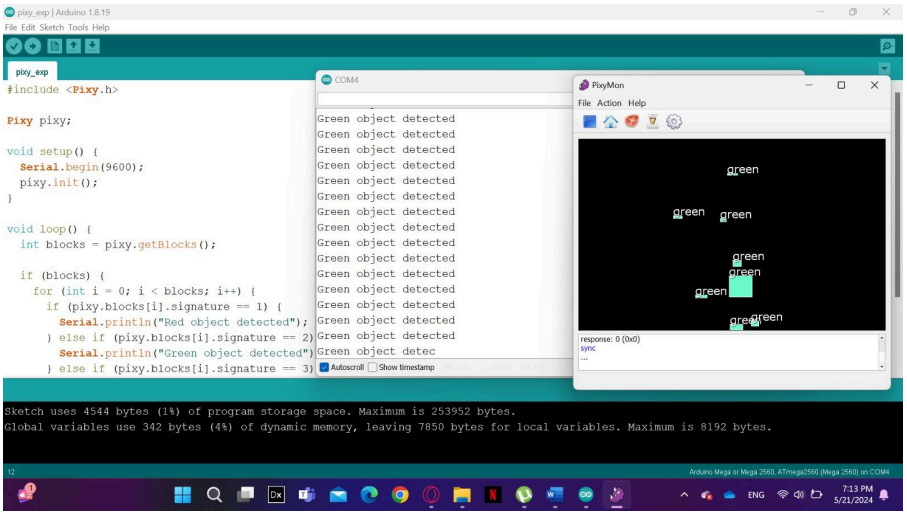


Fig. 4

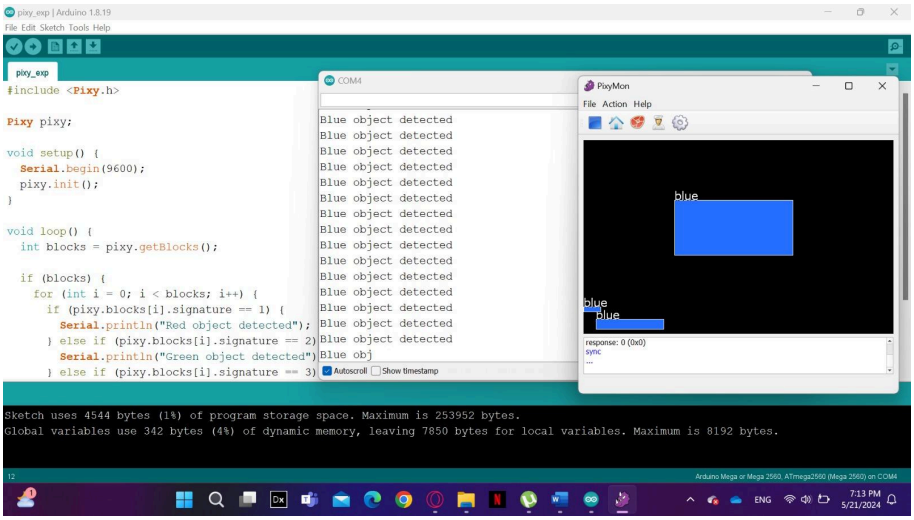


Fig. 5

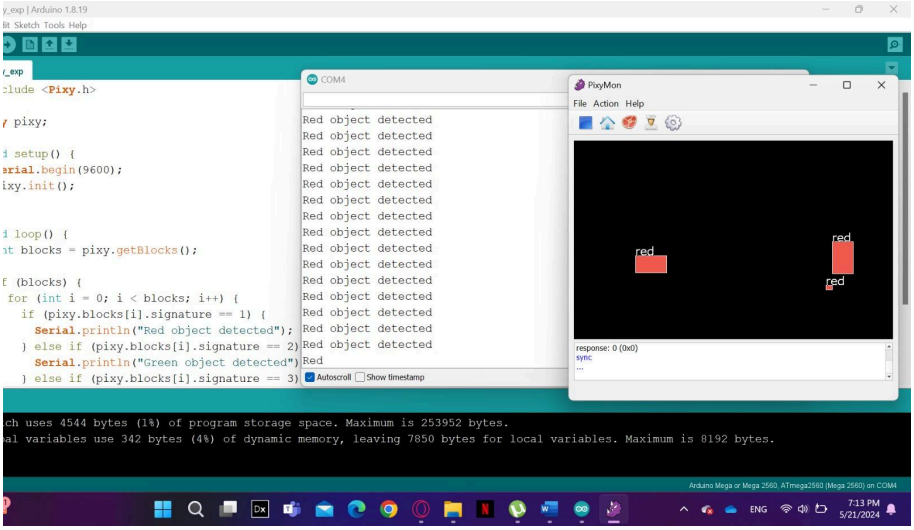


Fig. 6

RESULTS

To begin, the experiment regarding the color sensor was not completed as seen in the figures provided in the data collection for color sensor section.

For the experiment of color detection using the laptop webcam we used the first Python code and what that does is detect objects, then draws rectangles around them and finally specifies the color it detects. The result is shown in Fig. 1. For the second Python code however, it detected the most prominent color on the screen then stated it above the video as shown in Fig. 2.

Moreover, in Fig. 3 the pixy camera and Arduino are used to videograph the objects then detect their colors on the screen and then draw rectangles around them which is done on the cooked video setting and what that means is that the processing is happening on the PC. Additionally, the default

footage is shown in the three images Fig.4, 5, and 6, with the black screens and color rectangles on them. On the default setting the processing happens onboard the Pixy

DISCUSSION

For Color Sensor:

This part of the experiment was not successful and we were not able to see any data or values on the serial plotter or Python from the sensor, we believe it could be because of a hardware failure or environmental interference like electromagnetic interference, radio frequency interference, or environmental conditions software or it could be that the software controlling the sensor might be misconfigured or experiencing bugs that prevent it from sending data correctly.

For Laptop Webcam:

The accuracy of the detection is high, as seen in the video and images the program was able to state the correct colors on the screen 100% of the time. Moreover, even with different lighting conditions it seemed to work fine and retain its accuracy rate. The speed of the response was high and it was detecting colors within seconds of the objects being in frame.

For Pixy Camera:

The Camera is able to detect different colors (red, green, blue) in good lighting without a problem. However, it seemed to have a difficult time recognizing the colors in high or low intensity lit rooms. In our experience, the ideal setting for a Pixy camera is a yellow light that is set to normal brightness. The response rate was also fast but it was slower than the laptop webcam which we believe is because of the lower resolution and quality of the Pixy camera.

CONCLUSION

In this lab experiment, we explored the capabilities of image and video input interfacing with microcontroller and computer-based systems. Through the utilization of color sensors, laptop webcams, and Pixy cameras. Despite encountering challenges with the color sensor, we successfully implemented solutions with the laptop webcam and Pixy camera setups. Overall, the experimentation highlighted the critical role of hardware and software integration in developing efficient image and video input systems. By harnessing the power of microcontrollers and computer-based processing, we can create versatile and responsive systems capable of performing complex tasks such as color detection. Moving forward, further optimization and refinement of hardware configurations and software algorithms will be essential to enhance system robustness and performance in real-world applications.

RECOMMENDATIONS

It takes some time to perfect the configuration in the PixyMon application and Arduino which is done by adjusting color signatures and threshold values, it is important to find a good shade of the colors for configuration and when drawing the rectangular on the PixyMon it should only include the color needed. We also recommend using the Pixy camera in a room with medium light brightness.

REFERENCES

Arduino Color Sensing Tutorial - TCS230 TCS3200 Color Sensor - HowToMechatronics. (2018, August 4). HowToMechatronics. https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorial-tcs230-tcs3200-color-sensor/

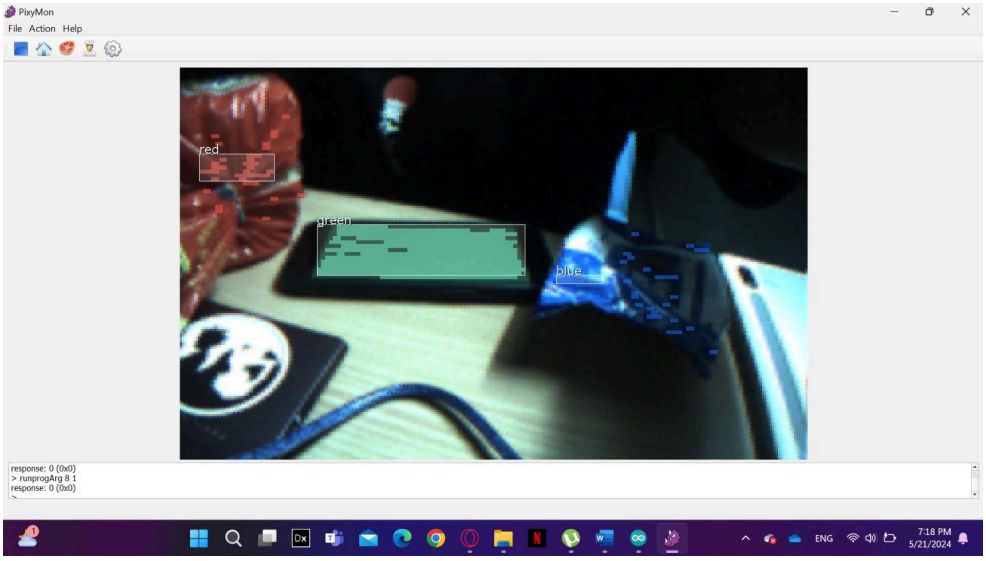
Marc de Vinck. (2019, September 4). Arduino Prototyping Inputs #63: Image Tracking with the PIXY! Camera. YouTube. https://www.youtube.com/watch?v=Y7V7uf4-v70

Workshop, D. (2018, October 19). Pixy2 Camera – Simple Object Recognition camera. DroneBot Workshop. https://dronebotworkshop.com/pixy2-camera/

Santos, R., & Santos, S. (2017, April 25). Arduino Color Sensor TCS230 TCS3200 | Random Nerd Tutorials. Random Nerd Tutorials. https://randomnerdtutorials.com/arduino-color-sensor-tcs230-tcs3200/

wiki:v1:pixymon_overview [Documentation]. (n.d.). Docs.pixycam.com. Retrieved May 21, 2024, from https://docs.pixycam.com/wiki/doku.php?id=wiki:v1:pixymon_overview

APPENDICES



ACKNOWLEDGEMENT

I would like to extend my sincere thanks to DR. Wahju for his guidance and understanding. Additionally, I would like to express my sincere gratitude to our senior for her invaluable guidance and support throughout the execution of this experiment. Her expertise and insightful feedback greatly enhanced my understanding of the principles underlying Arduino-based electronics Furthermore, I am thankful to my peers for their collaborative spirit and exchange of ideas, which enriched the overall learning experience and fostered a collaborative learning environment.

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 persons.

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 contributors to the report.

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

Name: Syeda Samia Matric Number: 2123536 Contribution:	Read	✓
	Understand	✓
	Agree	✓
Name: Athirah Haziqah bt Baderulhisham Matric Number: 2210240 Contribution:	Read	✓
	Understand	✓
	Agree	✓
Name: Siti Heidi Amira Binti Azrul Matric Number: 2111970 Contribution:	Read	✓
	Understand	✓
	Agree	✓