LAB REPORT

MCTA 3203

SECTION 1

GROUP D

EXPERIMENT 9

Image/Video input interfacing with microcontroller and computer based system: Software and hardware

MATIC NO.	NAME
2111633	MUHAMMAD FARID BIN JAFRI
2110765	MUHAMMAD ARIF SYAZWAN BIN MOHD RIZAL
2112969	MUHAMMAD HAZIQ HAIKAL BIN SUAIB
2115207	MUHAMMAD AMIRUL SYAFIQ BIN RUSLANI
2116281	MUHAMMAD FADHLUL WAFI BIN AHMAD NAIM

SUBMISSION DATE: 20/12/2023

INSTRUCTOR: DR. WAHJU SEDIONO

Abstract

The subsequent report consists of comprehensive details on an experiment conducted in the field of mechatronics system integration. The experiment involves creating a color detection system using Arduino IDE, Phyton, and either a color sensor or a USB camera. The experiment comprised two parts. Part 1 involved color detection with a color sensor while Part 2 utilized a USB camera for color detection. The experiment encompassed hardware setup which included , Arduino board, breadboard, jumper wires, RGB LED, and USB cable which are the foundation for this experiment for part 1 while color sensor GY-33 TCS34725 is the key component to complete this part. Same goes to the part 2 of the experiment in which Arduino board, breadboard, jumper wires, and USB cable are the foundation for this experiment and USB Camera is the key component to complete this part.

Table of Contents

Introduction	**4**
Materials and Equipment	**5**
Experimental Setup	**6**
Methodology	**8**
Results (Observation)	**12**
Discussion	**14**
Task	**15**
Recommendations	**16**
Conclusion	**17**
Acknowledge	**17**
References	**18**
Student's Declaration	**19**

Introduction

The GY-33 TCS34725 is a module designed for detecting colors, with the TCS34725 sensor device. It has the ability to distinguish a broad spectrum of colors by analyzing the levels of red, green, and blue light. The module facilitates data transmission with microcontrollers (Arduino UNO R3) over I2C and is commonly employed in industrial applications and projects requiring accurate color detection. Establish a connection between the color data and Arduino UNO R3, then retrieve and analyze the data prior to implementing it in the experiment. In part 1, we use LEDs with various colors such as green, yellow, and red, and detect the color by using GY 33. For the second part of the experiment, we used the same color sensor but we added a camera in the setup so that the camera could picture images while detecting the color that appears on the laptop screen. From this experiment, we can design a color detection system using Arduino, Python, and either a color sensor or a USB camera. We also can analyze the accuracy and performance of color detection in different scenarios.

Materials and Equipment

Part 1

1. Arduino board
2. Color sensor (TCS34725)
3. Jumper wires
4. Breadboard
5. RGB LED
6. Computer with Arduino IDE and Python installed
7. USB cable for Arduino UNO R3
8. Resistors

Part 2

1. Arduino board
2. USB camera
3. Jumper wires
4. Breadboard
5. Computer with Python installed
6. USB cable for Arduino UNO R3

Experimental Setup

Part 1

1)Hardware setup:

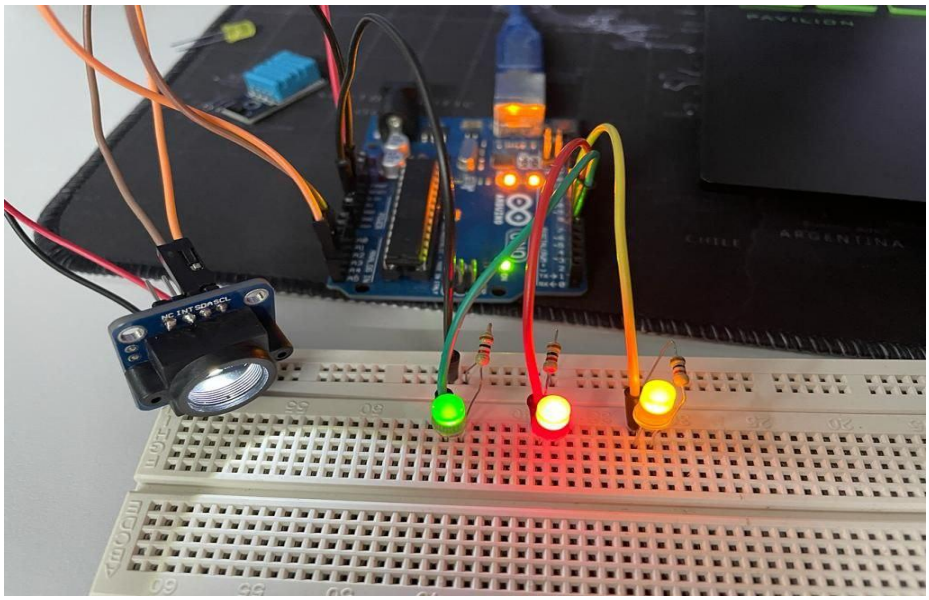
- The color sensor was connected to the Arduino using jumper wires.
- An RGB LED was connected to the Arduino for color display.

2)Arduino programming:

- An Arduino sketch to interface with the color sensor was already written.
- RGB color data from the sensor was read and converted to a format that could be sent to the computer.
- The sensor was calibrated for accurate color readings.

3)Bluetooth programming:

- A Python program to communicate with the Arduino over the serial connection using the pyserial library was written.
- RGB color data from the Arduino UNO R3 was received.
- The received data was interpreted to determine the detected color. The code was modified to make it suitable for the experiment.



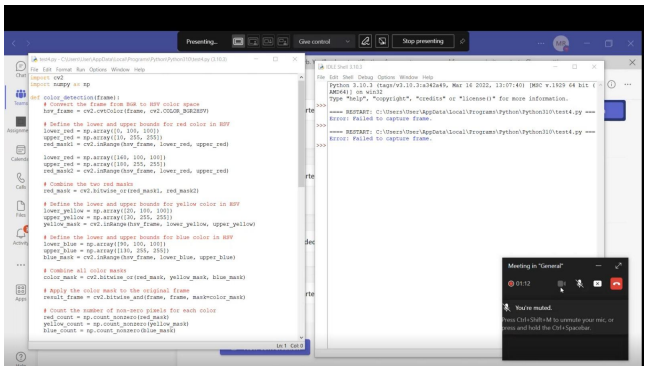
Part 2

1)Hardware setup:

- The USB camera was connected to the computer using a USB cable.

2)Arduino programming:

- The Python program to capture video from the USB camera using the OpenCV library was written.
- A basic code to read video signals from a webcam was written.
- Color detection logic using the HSV color space was implemented.



Methodology

Procedure:

1. The code for the Arduino board was uploaded into the microcontroller using Arduino IDE.
2. The python code was saved and run using Python IDLE.
3. The system has been tested with different colored objects.
4. The data on the detected colors, their accuracy and how the system performs in various lighting conditions has been collected.
5. The response time of the system when detecting colors has been analyzed.

Part 1: TCS34725

```
#include <Wire.h>
#include <Adafruit_TCS34725.h>

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);

#define LED_GREEN 5
#define LED_YELLOW 6
#define LED_RED 7

void setup() {
  Serial.begin(9600);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_YELLOW, OUTPUT);
  pinMode(LED_RED, OUTPUT);
  digitalWrite(LED_GREEN, HIGH);
  digitalWrite(LED_YELLOW, HIGH);
  digitalWrite(LED_RED, HIGH);
  if (!tcs.begin()) {
    Serial.println("Could not find a valid TCS34725 sensor, check wiring!");
    while (1);
  }
}

void loop() {
  uint16_t red, green, blue, clear;

  tcs.getRawData(&red, &green, &blue, &clear);
  Serial.print(red);
  Serial.print(" ");
  Serial.print(green);
  Serial.print(" ");
  Serial.print(blue);
  Serial.println();

  delay(1000);
}
```

```python
import serial
import time

ser = serial.Serial('COM5', 9600)

def interpret_color_data(data):

    red, green, blue = map(int, data.split())

    if red > 100 and green > 100 and red > blue and green > blue:
        return "Detected: Yellow"
    elif red > green and red > blue:
        return "Detected: Red"
    elif green > red and green > blue:
        if red + green + blue > 500:
            return "Detected: White"
        else:
            return "Detected: Green"
    elif red > blue and green > blue:
        return "Detected: Yellow"
    else:
        return "Cannot determine color"

try:
    while True:
        data = ser.readline().decode().strip()

        if data:
            color_result = interpret_color_data(data)
            print(color_result)

        time.sleep(1)

except KeyboardInterrupt:
    ser.close()
    print("Serial connection closed.")
```

Part 2: USB Camera

test4.py - C:\Users\User\AppData\Local\Programs\Python\Python310\test4.py (3.10.3)

File Edit Format Run Options Window Help

```
import cv2
import numpy as np

def color_detection(frame):
    # Convert the frame from BGR to HSV color space
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Define the lower and upper bounds for red color in HSV
    lower_red = np.array([0, 100, 100])
    upper_red = np.array([10, 255, 255])
    red_mask1 = cv2.inRange(hsv_frame, lower_red, upper_red)

    lower_red = np.array([160, 100, 100])
    upper_red = np.array([180, 255, 255])
    red_mask2 = cv2.inRange(hsv_frame, lower_red, upper_red)

    # Combine the two red masks
    red_mask = cv2.bitwise_or(red_mask1, red_mask2)

    # Define the lower and upper bounds for yellow color in HSV
    lower_yellow = np.array([20, 100, 100])
    upper_yellow = np.array([30, 255, 255])
    yellow_mask = cv2.inRange(hsv_frame, lower_yellow, upper_yellow)

    # Define the lower and upper bounds for blue color in HSV
    lower_blue = np.array([90, 100, 100])
    upper_blue = np.array([130, 255, 255])
    blue_mask = cv2.inRange(hsv_frame, lower_blue, upper_blue)

    # Combine all color masks
    color_mask = cv2.bitwise_or(red_mask, yellow_mask, blue_mask)

    # Apply the color mask to the original frame
    result_frame = cv2.bitwise_and(frame, frame, mask=color_mask)

    # Count the number of non-zero pixels for each color
    red_count = np.count_nonzero(red_mask)
    yellow_count = np.count_nonzero(yellow_mask)
    blue_count = np.count_nonzero(blue_mask)

    # Determine the dominant color
    max_count = max(red_count, yellow_count, blue_count)
    if max_count == red_count:
        dominant_color = "Red"
    elif max_count == yellow_count:
        dominant_color = "Yellow"
    else:
```

test4.py - C:\Users\User\AppData\Local\Programs\Python\Python310\test4.py (3.10.3)

File Edit Format Run Options Window Help

```
    return result_frame

def capture_and_detect_color(camera_index=0):
    # Open a connection to the camera
    cap = cv2.VideoCapture(camera_index)

    # Check if the camera opened successfully
    if not cap.isOpened():
        print("Error: Could not open camera.")
        return

    try:
        while True:
            # Capture frame-by-frame
            ret, frame = cap.read()

            # Check if the frame was read successfully
            if not ret:
                print("Error: Failed to capture frame.")
                break

            # Detect color in the frame
            result_frame = color_detection(frame)

            # Display the original frame
            cv2.imshow("Original Video Feed", frame)

            # Display the frame with color detection
            cv2.imshow("Color Detection", result_frame)

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

    finally:
        # Release the camera and close all OpenCV windows
        cap.release()
        cv2.destroyAllWindows()

if __name__ == "__main__":
    # You can specify the camera index (0 is usually the default USB camera)
    camera_index = 0

    # Call the function to capture video and detect color
    capture_and_detect_color(camera_index)
```

Results (Observation)

Part 1: TCS34725

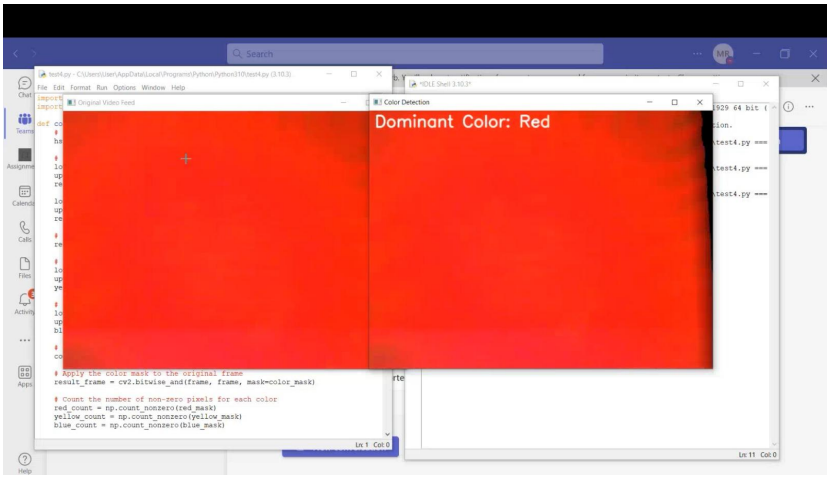
```
Cannot determine color
Cannot determine color
Cannot determine color
Cannot determine color
Cannot determine color
Detected: Green
Detected: Green
Detected: Green
Detected: Green
Detected: Green
Detected: Green
Detected: Red
Detected: Red
Detected: Red
Detected: Red
Detected: Red
Detected: Red
Detected: Red
Detected: Red
Detected: Yellow
Detected: Yellow
Detected: Yellow
Detected: Yellow
Detected: Yellow
Detected: Yellow
Detected: Yellow
Detected: Green
Detected: Green
Detected: Green
Detected: Green
Detected: Green
Cannot determine color
Detected: Green
Detected: White
Detected: White
Detected: White
Serial connection closed.
```

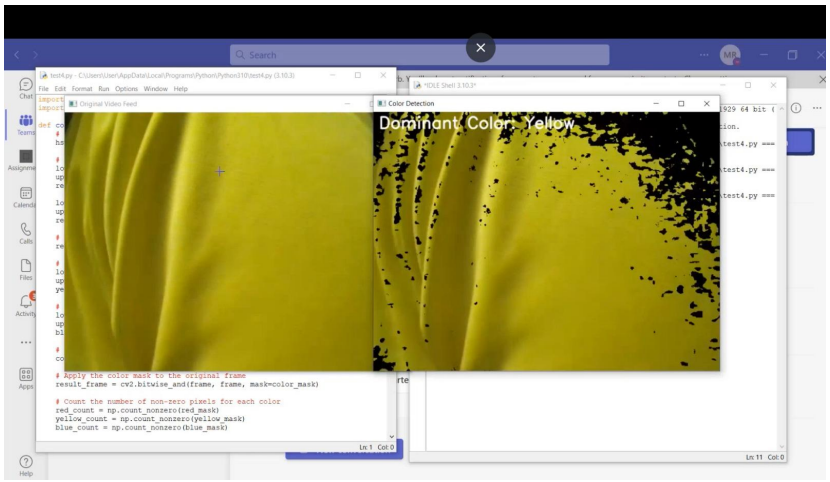
Fig. 1: *Output on the python*

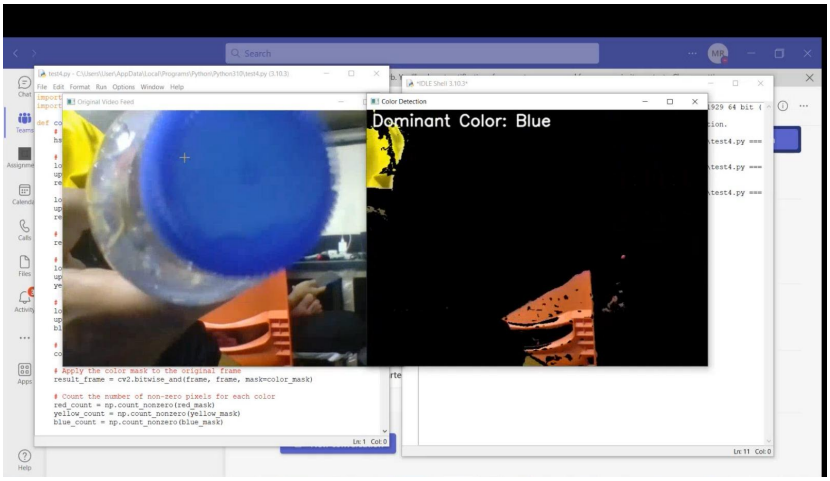
Link for the video:

https://drive.google.com/file/d/1SHQ2w5OMJGSSr2KVaVlPn82MvTIhB5CZ/view?usp=sharing

Part 2: USB Camera







Link for the video:

https://drive.google.com/file/d/1vjTrKJbdBv1Hdr9WFVI1To5VtRPy3q_P/view?usp=drivesdk

Discussion

1. Detection Discrepancies: One problem we had was that the webcam wasn't always able to correctly pick up the blue color when we were testing it. The webcam was able to identify and show red and yellow colors without any problems, but blue was harder. This might be because webcams aren't always good at capturing certain color spectra correctly.

2. Problems with Image Display: Another interesting thing that was noticed was that the webcam could see the blue color, but it couldn't show the image that went with it. This seems to suggest that the webcam might have trouble displaying or processing certain colors. More research needs to be done to find out why this difference exists.

3. Future Improvements: In the future, experiments could look into ways to make the webcam setup better, like changing the camera settings, using image processing methods, or adding more filters to make the colors more accurate. This iterative process might help get around some of the problems that have been found.

In conclusion, the color sensor was more accurate and useful than the webcam, showing that it could be used in situations where precise color recognition is needed. But knowing the exact limits of each tool lets you choose the best one for the job.

Task

Accurate detection of red, green, and yellow colors was demonstrated by the TCS34725 sensor working with Arduino. Reliable data was provided by the sensor, and the Arduino code effectively converted the sensor inputs into color information. Because of its hardware-based color detection solution provided by the integration of TCS34725 with Arduino, it is appropriate for applications where real-time color sensing is essential. When it came to differentiating between the designated hues, our technique showed good accuracy and responsiveness.

A flexible and user-friendly method for color recognition using a laptop camera was provided by using Python code. Real-time photos were successfully taken by the laptop camera, and the hues red, blue, green, and yellow were identified by processing. Applications where a larger range of colors needs to be detected or where hardware sensors may not be feasible benefit from OpenCV. On the other hand, it may be more susceptible to changes in lighting and surroundings.

Issues with lighting that affect color identification accuracy could provide challenges for the experiment, particularly for the laptop camera-based system. There is also room for improvement when it comes to handling color shade fluctuations and calibrating the TCS34725 sensor. The accuracy of software-based color recognition may also be impacted by noise or interference in the camera images.

The laptop camera and TCS34725 sensor should be calibrated under different lighting situations, and the Python code's algorithms should be adjusted for improved noise reduction, in order to improve the system.

Future iterations of the system may investigate a hybrid method that integrates software and hardware solutions. This would take advantage of software-based color recognition employing a camera's flexibility and adaptability while leveraging the precision and real-time capabilities of hardware-based sensors like the TCS34725.

Recommendations

To begin with, it is advisable to further refine the color detection algorithms to bolster the system's resilience across a range of real-world conditions. This could involve fine-tuning color thresholds, delving into advanced color spaces, or incorporating dynamic adaptability through the integration of machine learning techniques capable of adjusting to evolving environments.

Moreover, to fortify the experiment's reliability, broaden the dataset by incorporating a more diverse array of color samples and objects. Enrich the experiment with intricate scenarios mirroring real-world conditions, encompassing variables such as dynamic shifts in lighting, varying distances between the sensor and objects, and a spectrum of background conditions.

Lastly, delve into the exploration of collaborative or multi-sensor systems, wherein multiple color sensors or cameras collaborate to augment color detection precision and dependability. Investigate communication protocols and synchronization techniques to efficiently coordinate data from multiple sources, thereby enhancing the overall effectiveness of the color detection system.

Conclusion

In the end, our experiment looked at how well the color sensor and webcam could identify colors. The color reader was very accurate and flexible, and it was great at recognising colors accurately even from far away. The webcam did a good job of detecting colors in general, but it had trouble with some shades. For example, it had a hard time detecting and showing blue colors correctly. This shows how important it is to think about the needs of the programme when choosing between the color sensor's accuracy and the webcam's general features. What we learned from this experiment will help us with other projects where we work with different kinds of monitors. Because we know how to use the OpenCV library in Python, we will be able to use cameras and video files in Python code for future projects.

Acknowledge

We would like to extend our appreciation to Dr. Wahju Sediono, a lecturer of the course Mechatronic System Integration 1 (MCTA 3203), for his valuable guidance during the execution of the experiment.

References

1. *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/
2. *Arduino Color Sensing Tutorial - TCS230 TCS3200 Color Sensor*. (n.d.). Www.youtube.com. https://www.youtube.com/watch?v=CPUXxuyd9xw
3. *Arduino Color Sorter Project*. (n.d.). Www.youtube.com. https://www.youtube.com/watch?v=g3i51hdfLaw

Student's Declaration

Certificate of Originality and Authenticity

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 acknowledgment, and that the original work contained herein has 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 contributor 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

Signature: *arifsyazwan* — Read ✔️
Name: Muhammad Arif Syazwan Bin Mohd Rizal — Understand ✔️
Matric Number:2110765 — Agree ✔️

Signature: *haziqhaikal* — Read ✔️
Name: Muhammad Haziq Haikal bin Suaib — Understand ✔️
Matric Number: 2112969 — Agree ✔️

Signature: *amirul* — Read ✔️
Name: Muhammad Amirul Syafiq Bin Ruslani — Understand ✔️
Matric Number: 2115207 — Agree ✔️

Signature: *Wafi* — Read ✔️
Name: Muhammad Fadhlul Wafi Bin Ahmad Naim — Understand ✔️
Matric Number: 2116281 — Agree ✔️

Signature: *faridjafri* — Read ✔️
Name: Muhammad Farid Bin Jafri — Understand ✔️
Matric Number: 2111633 — Agree ✔️