Mechatronics System Integration (MCTA3203)

Week **9**: Image/Video input interfacing with microcontroller and computer based system: Software and hardware.

Color Detection and Analysis

Objective:

To design a color detection system using Arduino, Python, and either a color sensor or a USB camera. The experiment will involve hardware setup, programming Arduino and Python, and analyzing the accuracy and performance of color detection in different scenarios.

Part 1: Color Detection with Color Sensor

Materials Needed:

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

Experiment Steps:

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 guidance[1].
- If using an RGB LED, connect it to the Arduino for color display.

2. Arduino Programming:

- Write an Arduino sketch[1] 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[2].

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.

Part 2: Color Detection with USB Camera

Materials Needed:

- Arduino board
- USB camera
- Jumper wires
- Breadboard
- Computer with Python installed
- USB cable for Arduino

Experiment Steps:

1. Hardware Setup:

- Connect the USB camera to the computer using a USB cable.
- If necessary, connect an Arduino for additional functionalities (optional).

2. Python Programming:

- Write a Python program to capture video from the USB camera using the OpenCV library. A basic code to read video signal from a webcam is shown below[3].
- Implement color detection logic using HSV color space.
- Display the video feed and detected colors on the computer screen.

3. 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*.

4. Analysis:

- Evaluate the accuracy of color detection by comparing detected colors with actual colors.
- Analyze how the system performs in *different lighting conditions*.
- *Compare* the performance of the USB camera-based system with the color sensor-based system.

Task

Summarize the *key findings* from both parts of the experiment, discuss any *challenges* or *improvements*, and provide insights for *future* iterations of the system.

[1]Useful links:

[1] https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorial-tcs230-tcs3200-color-sensor/
[2] https://www.youtube.com/watch?v=CPUXxuyd9xw
[3] https://www.youtube.com/watch?v=g3i51hdfLaw

[2]Python code:

```
# --
# Simple code for RGB color detection
# --
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]       # Normalize RGB values

    # Create a block for each channel with linear gradients
    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]

    # Combine the color channel horizontally
    # channels_combined = np.hstack(rgbchannel)

    # Create the final RGB values for the specified color
    true_color = np.ones((10, 30, 3))*color

    # Combine all vertically
    # final_display = np.vstack([channels_combined, true_color])
    final_display = np.vstack([np.hstack(rgbchannel), true_color])

    # Display the final block
    plt.imshow(final_display)
    plt.axis('off')
    plt.show()


# -- Example: Display color for random RGB values
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)
```

[3]Python code:

```
# --
# Basic code for checking connection to webcam (or USB camera)
# --
import cv2

print("Press ESC to exit.")
capture = cv2.VideoCapture(0)    # adjust as needed
if capture.isOpened():
    print('Camera is opened')
```

```
while True:
    ret, frame = capture.read()

    if ret:
        cv2.imshow('Vid1', frame)

    t = cv2.waitKey(1)           # wait for 1 second

    # Check if the window is closed
    if (cv2.getWindowProperty('Vid1', cv2.WND_PROP_VISIBLE) < 1):
        break

    # Check if ESC or 'x' is pressed
    if (t == 27) or (t & 0XFF == ord('x')):
        break

capture.release()
cv2.destroyAllWindows()
print('Camera is closed')
```