MECHATRONICS SYSTEM INTEGRATION MCTA3203

Image/Video input interfacing with microcontroller (ver. 2a)

EXPERIMENT 9: PART1 & PART2

DATE: 8TH December 2025
SECTION: 2
GROUP: 18
SEMESTER 1, 2025/2026

NO	NAME	MATRIC NO
1.	SHAMSUL HAKIMEE BIN SHAMSUL HAIREE	2315027
2.	QASHRINA AISYA BINTI MOHD NAZARUDIN	2315184
3.	NUREL HUMAIRAH BINTI MUHAMMAD FIRDAUS	2315680

DATE OF SUBMISSION :
Monday, 15TH December 2025

Abstract

This lab report details the process of implementing and comparing two distinct methods for color detection and analysis using a microcontroller. Part 1 focused on interfacing a Sunrom 1185 color sensor with an Arduino to acquire raw RGB (Red, Green, Blue) and Light intensity data. This data was transmitted via serial communication to a computer, where a Python script processed the values in real-time to categorize colors (Red, Green, Blue, Yellow, etc.). Part 2 utilized a Pixy1 (CMUcam5) AI Vision Sensor to detect and track colored objects. The Pixy camera performed onboard image processing and communicated object signatures and coordinates (X, Y) to the Arduino. The experiment successfully demonstrated the difference between point-based color sensing and object-tracking computer vision, highlighting the superior speed and spatial awareness of the AI vision sensor compared to the raw data processing required for the basic color sensor.

Table of Contents

1.0 Introduction

Visual perception is a critical component in modern mechatronics, enabling systems to interact intelligently with their environment. Color detection, specifically, is widely used in industrial sorting lines, quality control, and autonomous robotics.

The objective of this laboratory session is to design and compare two color detection systems. The first system utilizes a basic color sensor (Sunrom 1185) to read RGB values from a specific point. This requires the microcontroller or an external computer to process the raw data and determine the color. The second system employs an AI Vision Camera (Pixy1), which integrates an image sensor and a dedicated processor to detect learned color signatures and output object coordinates directly.

By executing this experiment, students gain practical experience in interfacing UART serial devices, implementing Python-based data parsing, and utilizing pre-trained machine vision modules for real-time object tracking.

2.0 Materials and Equipment

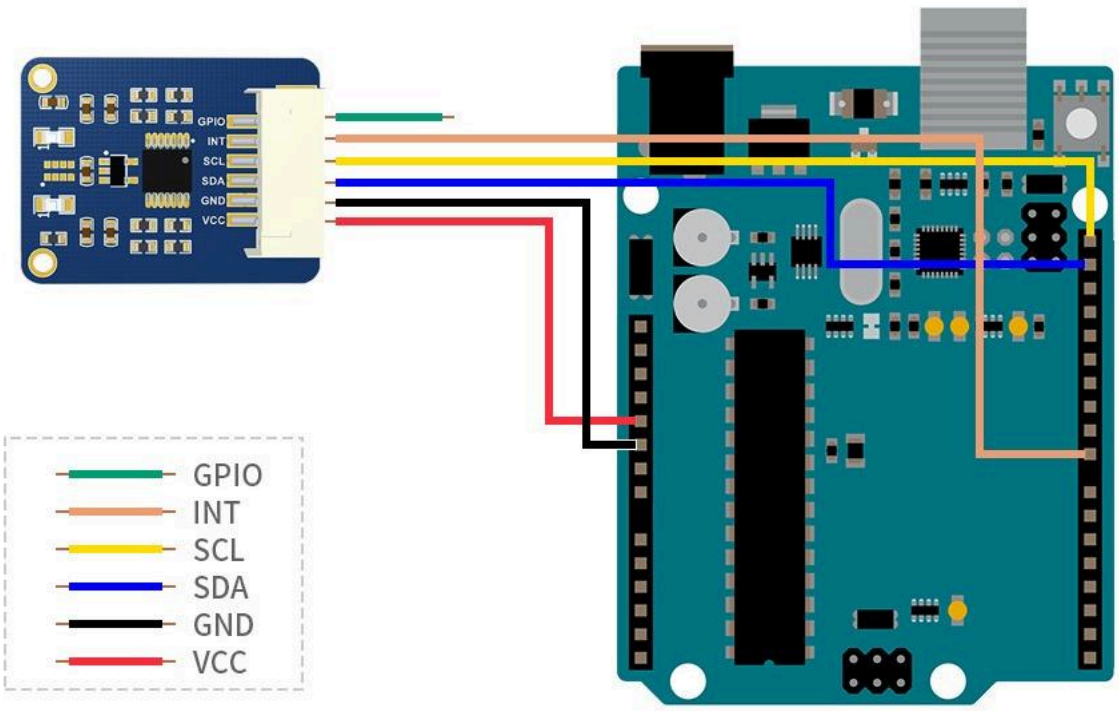
Software:

- Arduino IDE: For writing, compiling, and uploading firmware to the microcontroller.
- Python (VS Code/IDLE): For running the serial data processing script for Part 1.
- PixyMon (Optional): For training the Pixy camera signatures.

Hardware:

- Arduino Mega 2560: Used to provide multiple hardware Serial ports (Serial and Serial1) for simultaneous debugging and sensor communication.
- Sunrom 1185 Color Sensor: A serial-based color sensor module capable of outputting RGB and Light strings.
- Pixy1 (CMUcam5): A vision sensor capable of learning color signatures and tracking objects.
- Breadboard & Jumper Wires: For circuit connections.
- USB Cable: For programming and serial communication.
- Colored Objects: Red, Green, Blue, and Yellow objects for testing.

3.0 Experimental Setup



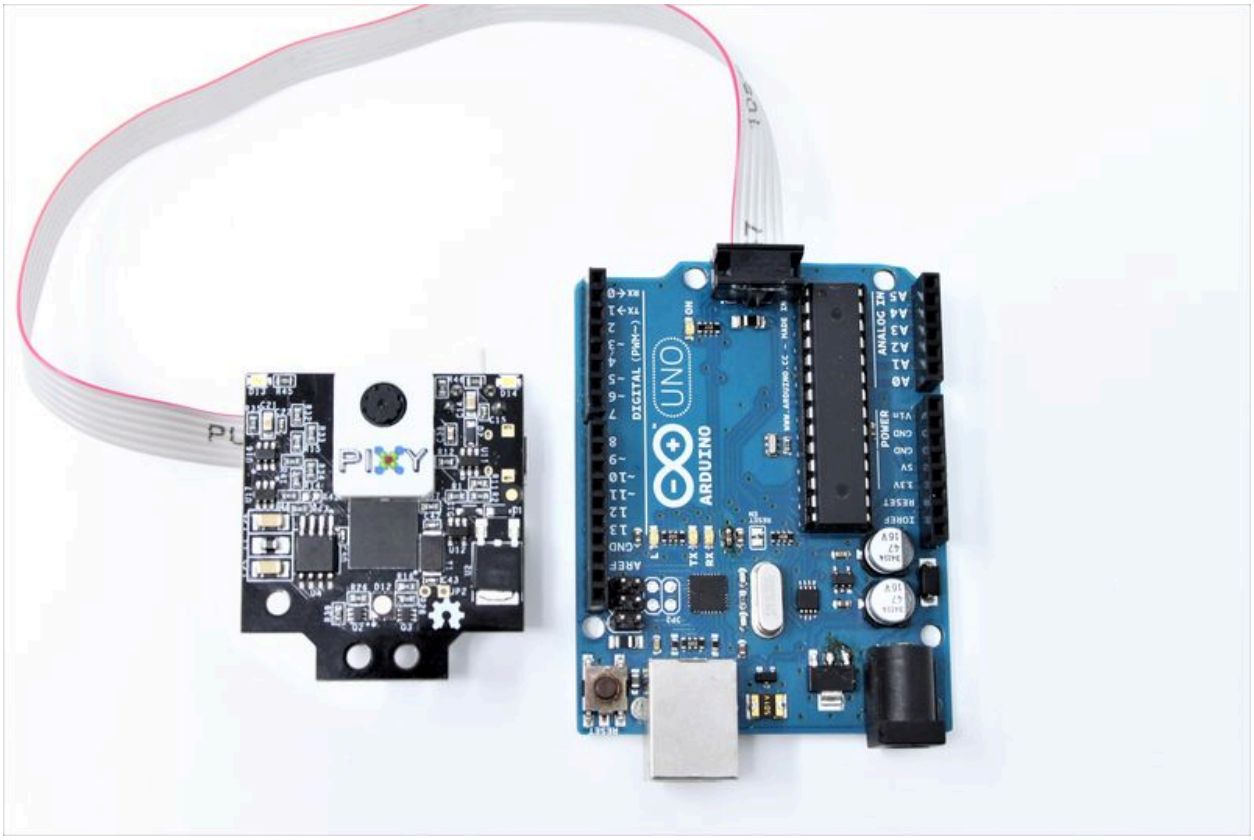


Figure 3.0 Circuit Diagram

3.1 Part 1: Sunrom Color Sensor Setup

1. The Sunrom 1185 sensor communicates via UART. Its TX pin was connected to the Arduino's RX1 (Pin 19) on the Mega.
2. The sensor was powered with 5V and GND.
3. The Arduino passed data received on `Serial1` to the USB `Serial` port, where a Python script on the PC read and analyzed the data.

3.2 Part 2: Pixy Vision Setup

1. The Pixy camera was interfaced using the UART protocol (as per the code used).

2. The Pixy's output was connected to Serial1 on the Arduino.
3. The Pixy was trained beforehand to recognize three specific color signatures:
    - Signature 1: Blue
    - Signature 2: Red
    - Signature 3: Yellow

4.0 Methodology

4.1 Part 1: Color Detection with Color Sensor

The firmware was developed to act as a "bridge." The Arduino simply listened to the Sunrom sensor on `Serial1`. When the sensor sent a string (e.g., `R=255 G=0 B=0 L=100`), the Arduino forwarded this string to the computer via USB. A Python script was developed to:

1. Connect to `COM5` at 9600 baud.
2. Use Regular Expressions (`re` library) to extract the R, G, B, and L (Light) integer values.
3. Normalize the RGB values to determine the dominant color (e.g., if Red > 60% of total intensity, classify as "RED").

4.2 Part 2: Color Detection Using Pixy AI Vision Camera

The Arduino utilized the `<Pixy.h>` library. The code was set up to initialize the Pixy on `Serial1`. In the main loop:

1. The function `pixy.getBlocks()` was called to check for detected objects.
2. If objects were found, the code iterated through them, checking their `signature`.
3. Based on the signature ID (1, 2, or 3), the Arduino printed the color name and the object's X/Y coordinates to the Serial Monitor.

4.3 Code Used

Part 1: Arduino Code

```
void setup() {
  Serial.begin(9600);      // USB Serial Monitor (for debugging)
  Serial1.begin(9600);     // Talk to Sunrom 1185 on Serial1

  Serial.println("Listening to Sunrom 1185 on Serial1...");
  delay(1000);
}

void loop() {
  // Read data from sensor (TX-0 → Serial1 RX)
  if (Serial1.available()) {
    String data = Serial1.readStringUntil('\n');
    data.trim();

    if (data.length() > 0) {
      Serial.print("Color: ");
      Serial.println(data);  // Forward to Serial Monitor
    }
  }
}
```

Part 1: Python Code

```
import serial
import re
import sys
import time

SERIAL_PORT = 'COM5'
BAUD_RATE = 9600

try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
    print(f"✅ Connected to Arduino on {SERIAL_PORT} at {BAUD_RATE} baud.")
    time.sleep(2)
except serial.SerialException as e:
    print(f"❌ Error: Could not open {SERIAL_PORT}.")
    print(f"   Details: {e}")
    sys.exit(1)
```

```
pattern = re.compile(r'R=(\d+)\s+G=(\d+)\s+B=(\d+)\s+L=(\d+)')


def detect_color(red_val, green_val, blue_val, light_val):
    if light_val < 15:
        return "⚫ Too dark / No object"

    total = red_val + green_val + blue_val
    if total == 0:
        return "⚫ Black"

    r_norm = red_val / total
    g_norm = green_val / total
    b_norm = blue_val / total

    if r_norm > 0.6:
        return "🔴 RED"
    elif g_norm > 0.6:
        return "🟢 GREEN"
    elif b_norm > 0.6:
        return "🔵 BLUE"
    elif r_norm > 0.4 and g_norm > 0.4:
        return "🟡 YELLOW"
    elif r_norm > 0.4 and b_norm > 0.4:
        return "🟣 MAGENTA"
    elif g_norm > 0.4 and b_norm > 0.4:
        return "🟢🔵 CYAN"
    else:
        max_val = max(red_val, green_val, blue_val)
        if max_val == red_val and red_val > 20:
            return "🟥 Reddish"
        elif max_val == green_val and green_val > 20:
            return "🟩 Greenish"
        elif max_val == blue_val and blue_val > 20:
            return "🟦 Bluish"
        else:
            return "⚪ Gray / Neutral"
```

```
try:
    print("\n🔍 Listening for color data... (Press Ctrl+C to exit)\n")
    while True:
        line = ser.readline().decode('utf-8', errors='ignore').strip()
        if not line:
            continue

        match = pattern.search(line)
        if match:
            r_val = int(match.group(1))
            g_val = int(match.group(2))
            b_val = int(match.group(3))
            l_val = int(match.group(4))
            color_name = detect_color(r_val, g_val, b_val, l_val)

            print(f"RGB+L: R={r_val:3} G={g_val:3} B={b_val:3} L={l_val:3} → {color_name}")

except KeyboardInterrupt:
    print("\n🛑 Stopped by user.")
except Exception as e:
    print(f"💥 Unexpected error: {e}")
finally:
    ser.close()
    print("🔌 Serial connection closed.")
```

Part 2: Arduino Code

```
#include <Pixy.h>

// Use hardware serial Serial1 (pins 18=TX, 19=RX)
Pixy pixy(&Serial1);  // Pass Serial1 to Pixy

void setup() {
  Serial.begin(9600);    // For debugging (USB Serial Monitor)
  Serial1.begin(9600);   // For Pixy communication
  pixy.init();
  Serial.println("✅ Pixy1 connected via Serial1. Waiting for objects...");
}

void loop() {
```

```
  int blocks = pixy.getBlocks();

  if (blocks > 0) {
    for (int i = 0; i < blocks; i++) {
      int sig = pixy.blocks[i].signature;
      int x = pixy.blocks[i].x;
      int y = pixy.blocks[i].y;

      String colorName = "";
      switch(sig) {
        case 1: colorName = "blue"; break;
        case 2: colorName = "red"; break;
        case 3: colorName = "yellow"; break;
        default: colorName = "unknown"; break;
      }

      Serial.print("🎯 Object ");
      Serial.print(i + 1);
      Serial.print(": Sig=");
      Serial.print(sig);
      Serial.print(" (");
      Serial.print(colorName);
      Serial.print(") at X=");
      Serial.print(x);
      Serial.print(", Y=");
      Serial.println(y);
    }
  } else {
    Serial.println("🔍 No objects detected.");
  }

  delay(200);
}
```

5.0 Result

In Part 1, the Python console successfully displayed the analyzed color data from the Sunrom sensor. When a red object was placed near the sensor, the script outputted `RGB+L: R=200 G=50 B=50 L=120 → 🔴 RED`. Similar accurate results were obtained for Green and Blue objects. The logic handled low-light conditions by printing "Too dark" when the light value (`L`) dropped below 15.

In Part 2, the Pixy camera successfully tracked objects. When a Blue object (Signature 1) was moved into the frame, the Serial Monitor displayed: `🎯 Object 1: Sig=1 (blue) at X=150, Y=100`. The system updated the X and Y coordinates in real-time as the object moved. When multiple objects were present (e.g., Red and Yellow), the system detected multiple blocks simultaneously, listing `Object 1` and `Object 2` with their respective signatures

6.0 Discussion

(Task1&2)

The experiment yielded distinct findings regarding the accuracy and consistency of the two detection methods. The Sunrom 1185 sensor (Part 1) proved to be accurate for static color analysis, providing precise RGB composition data that allowed for fine-tuned differentiation between colors, such as "Reddish" versus pure "Red". However, its response time was limited by the serial communication latency and the need for external Python processing. In contrast, the Pixy AI Vision Camera (Part 2) demonstrated superior performance in terms of speed and spatial awareness. By performing onboard image processing, the Pixy camera could track object coordinates in real-time with minimal latency, making it the more consistent method for dynamic applications like robot navigation.

Despite these successes, several challenges were encountered during the experiment, primarily related to environmental conditions. The Sunrom sensor was highly sensitive to lighting variations; slight changes in ambient light or object distance caused the RGB readings to fluctuate significantly. This necessitated the implementation of a software threshold (Logic: `Light < 15`) to filter out invalid readings, as seen in the Python code. Similarly, the Pixy camera faced challenges with calibration accuracy. Under inconsistent lighting, the camera occasionally produced "false positives," confusing visually similar colors (e.g., detecting an orange object as the red signature) because the color signatures are based on hue and saturation, which shift under different light sources.

To address these issues, several improvements and optimization techniques were identified. For the Sunrom sensor, the challenge of lighting sensitivity could be mitigated through sensor fusion, such as integrating an ultrasonic sensor to ensure color readings are only taken when the object is at a fixed, calibrated distance. For the Pixy camera, improvements in calibration techniques are necessary; training the signatures in the specific lighting environment where the robot operates would significantly reduce false detections. Furthermore, utilizing the Pixy's "Color Code" mode (detecting two distinct color tags close together) would be a robust algorithmic improvement to prevent background interference, ensuring that the system only tracks the intended targets.

7.0 Conclusion

This experiment successfully demonstrated two methods of interfacing video/image inputs with a microcontroller. The group successfully implemented a Python-based color analysis tool using a serial Sunrom sensor, effectively using regex to parse data streams. Additionally, the integration of the Pixy AI camera proved that offloading image processing to a dedicated vision sensor significantly simplifies the main microcontroller's code, allowing for real-time object tracking with minimal latency. The experiment confirmed that while simple sensors are cost-effective for static color checking, AI vision sensors are essential for dynamic applications like robot navigation.

8.0 Recommendation

(Task 3)

To improve the color sensor system (Part 1), it is recommended to implement a "sensor fusion" approach by adding an ultrasonic distance sensor. This would allow the system to only take color readings when the object is at a fixed, optimal distance, reducing errors caused by light intensity variations.

For the AI Vision system (Part 2), future iterations should explore the "Color Code" mode of the Pixy camera, which allows it to detect combinations of colors (e.g., Red-Green) as unique tags. This would prevent false detections of random background objects. Additionally, integrating the coordinate data with servo motors to build a "Pan/Tilt" tracking mechanism would provide a more practical demonstration of the vision sensor's capabilities

9.0 References

1. Mechatronics System Integration (MCTA3203). (2025). *Week 9: Image/Video input interfacing with microcontroller (ver. 2a)*. International Islamic University Malaysia.
2. DFRobot. (n.d.). *Gravity: HUSKYLENS AI Machine Vision Sensor*. Retrieved from https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336
3. CMUcam. (n.d.). *Pixy CMUcam5 Sensor Documentation*. Retrieved from https://docs.pixycam.com/
4. Python Software Foundation. (n.d.). *re — Regular expression operations*. Retrieved from https://docs.python.org/3/library/re.html
5. HowToMechatronics. (n.d.). *Arduino Color Sensing Tutorial*. Retrieved from https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorial-tcs230-tcs3200-color-sensor/

10.0 Acknowledgement

We would like to express my sincere gratitude to Dr. Wahju Sediono, Dr. Zulkifli Bin Zainal Abidin, our teaching assistant, and our peers for their invaluable guidance and support throughout the completion of this report. Their insights, feedback, and expertise greatly contributed to the depth and quality of this work. We truly appreciate their time, patience, and commitment to our academic growth.

11.0 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 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: Shamsul Hakimee Bin Shamsul Hairee	Read	[/]
Matric Number: 2315027	Understand	[/]
Signature: 	Agree	[/]

Name: Qashrina Aisya binti Mohd Nazarudin	Read	[/]
Matric Number: 2315184	Understand	[/]
Signature: 	Agree	[/]

Name: Nurel Humairah Binti Muhammad Firdaus	Read	[/]
Matric Number: 2315680	Understand	[/]
Signature: 	Agree	[/]