CPSC-440 Spring 2022 **Project Report** **Due: 05/05/2022**

Fulya Kocaman CWID: 803023878

Project Title: Color Recognition Using Raspberry Pi and TCS3200 Color Sensor

Materials used: Raspberry Pi, TCS3200 Color Sensor, Breadboard, Connecting Wires and connectors, Different colored materials

Software used: Color_Raw_RGB.py and Color_Recognition.py

Reference: https://www.electronicshub.org/raspberry-pi-color-sensor-tutorial/

This project consists of three parts; hardware component, software component, calibrating the colors, and testing for color recognition.

1. **Hardware Component:** The project uses the schematic from Figure 1 below that describes how to connect the TCS3200 color sensor to Raspberry Pi. TCS3200 Color Sensor has 8-pins and each one of them needs to get connected to the Raspberry Pi correctly. For the color sensor to work, the S0, S1 and V_{DD} pins of the sensor will be connected to the power source (+5V) of the Raspberry Pi. Then OE and GND pins of the sensor get connected to the GND of the Raspberry Pi. Finally, the S3, S4, and OUT pins of the sensor will be connected to GPIO23, GPIO24, and GPIO25 on the Raspberry Pi, respectively.

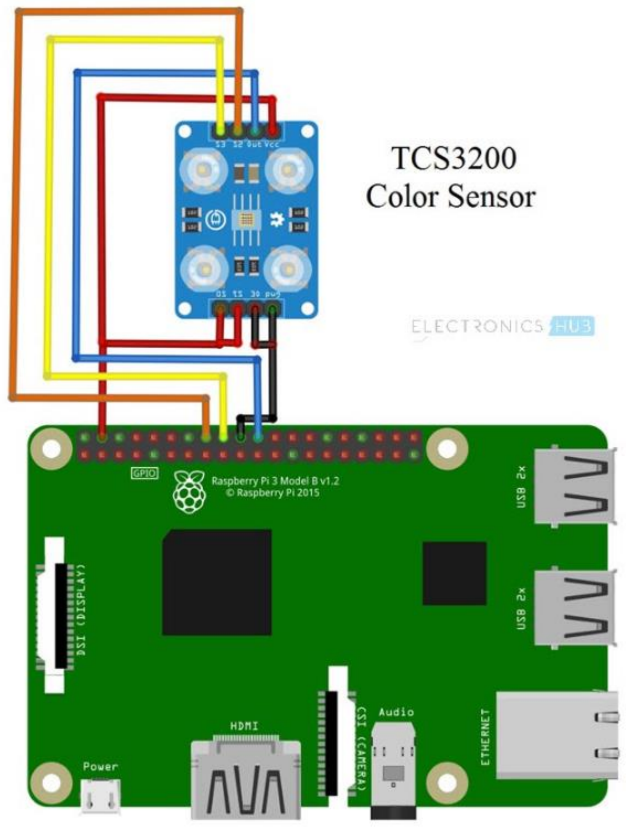

Figure 1: Circuit Diagram of Raspberry Pi Color Sensor TCS3200

Credit: https://www.electronicshub.org/raspberry-pi-color-sensor-tutorial/

The first challenge I had was connecting the TCS3200 correctly to the Raspberry Pi. The color sensor that was purchased seemed to be a little bit larger than the picture they had online. The sensor completely covered the breadboard, so I had no room to connect the pins of the sensor to the Raspberry Pi.

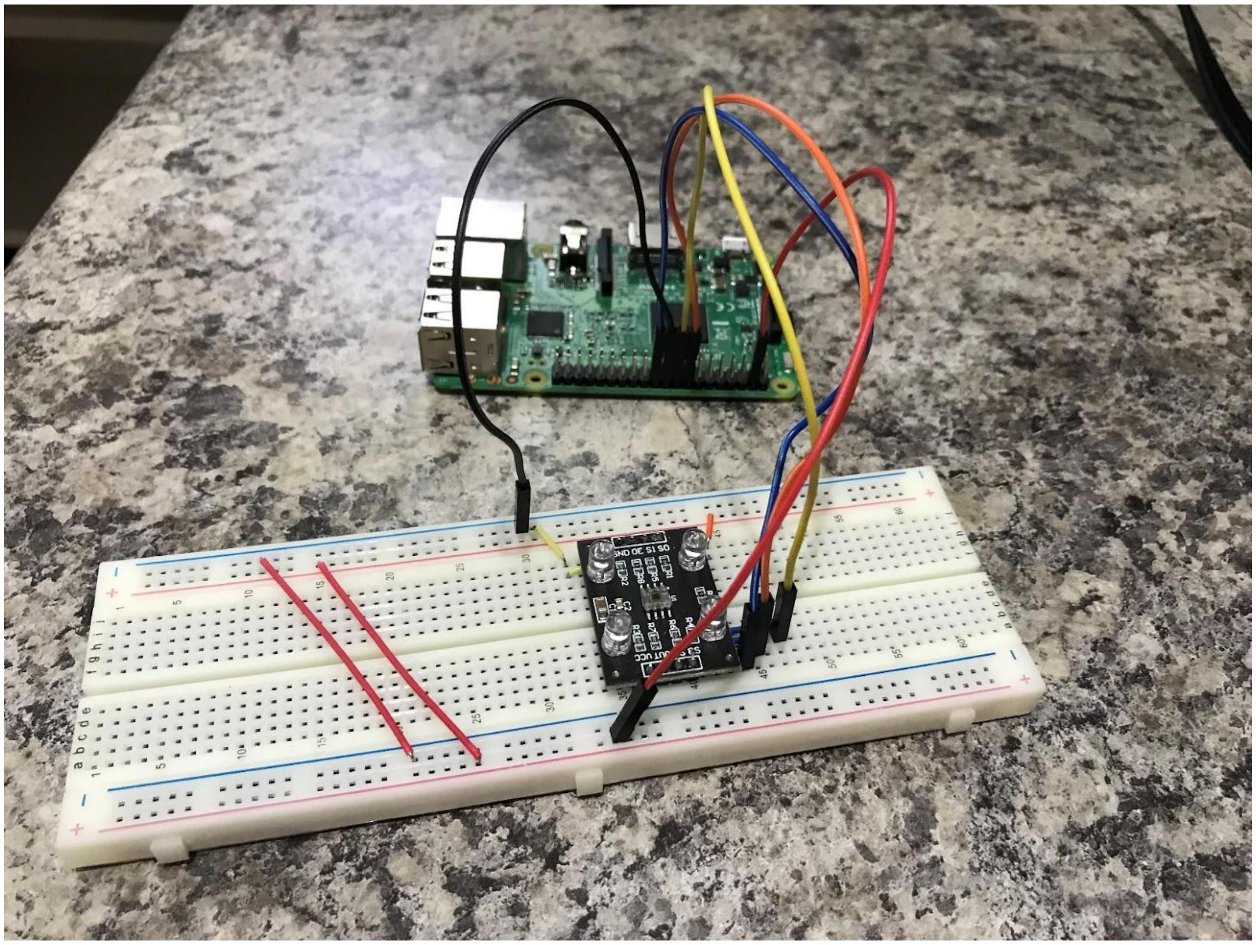

Figure 2: My Own Circuit Diagram of Raspberry Pi Color Sensor TCS3200

Figure 2 shows how I connected my color sensor to the Raspberry Pi. The way that I resolved this issue was to extend connections from the pins of the sensor through connected wires on the breadboard so that they can get properly connected to the Raspberry Pi. Since I had no knowledge of connecting wires, I took a few days to try it out, but finally, I was able to make the color sensor work. Once the sensor worked, all four lights on the sensor lit up as seen in Figure 3.

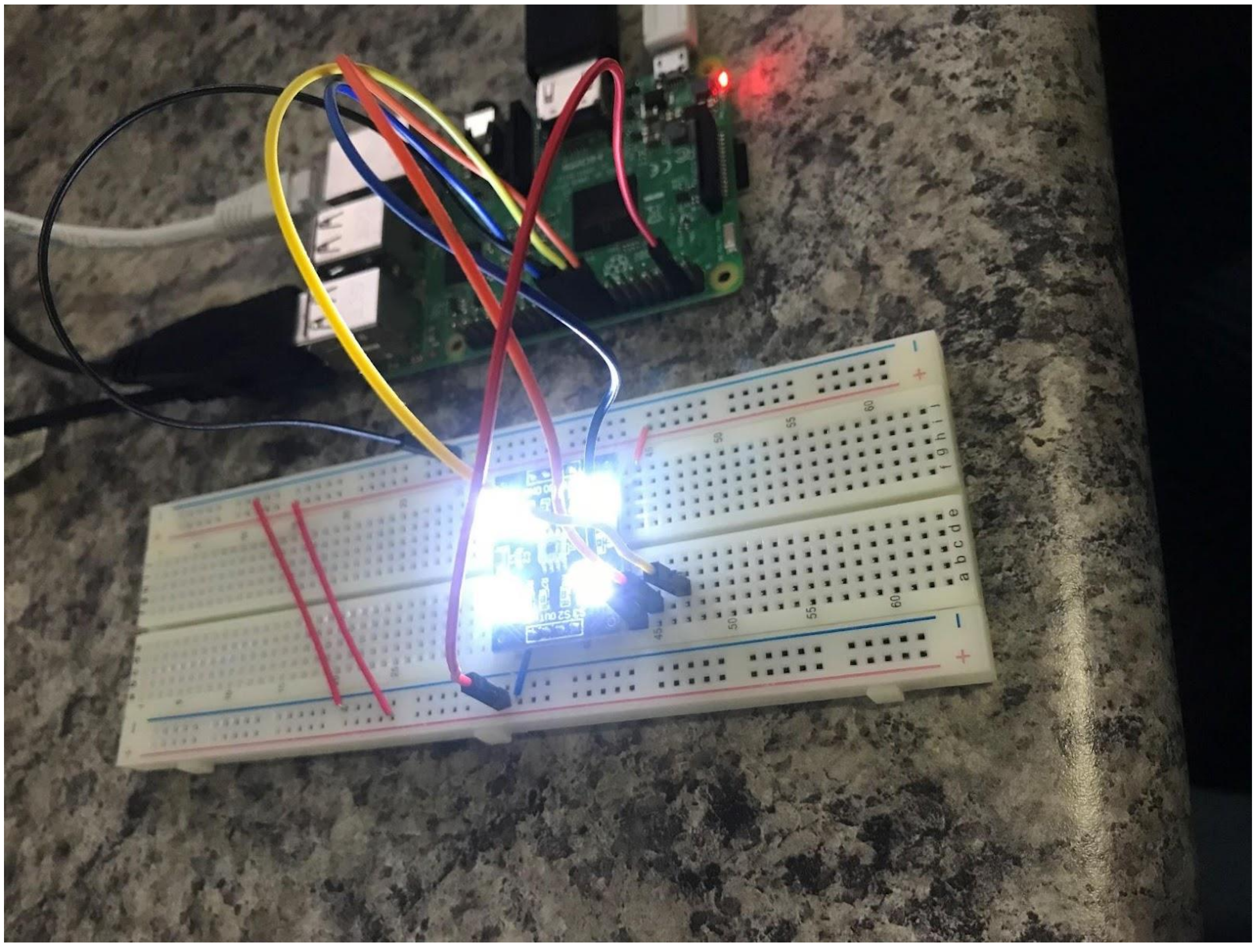

Figure 3: My Own Working TCS3200 Color Sensor

2. **Software Component:** This project uses two python files that run on Raspberry Pi.
    - **color_Raw_RGB.py** contains the code to see the Raw values of the RGB. When this file runs, GPIO.output(s2,GPIO.LOW) and GPIO.output(s3,GPIO.LOW) gives Raw value for color Red, GPIO.output(s2,GPIO.LOW) and GPIO.output(s3,GPIO.HIGH) gives Raw value for color Blue, and GPIO.output(s2,GPIO.HIGH) and GPIO.output(s3,GPIO.HIGH) gives Raw value for color Green. I used this code to calibrate the colors.
    - **color_Recognition.py** is so similar to color_Raw_RGB.py, but it contains calibrated color ranges for color detection. Below are the ranges I used for the colors:
      red when green<17000 and blue<20000 and red>22000
      green when red<16000 and blue<16000 and green>18000
      blue when green<19000 and red<19000 and blue>19000
      black when green<12000 and red<12000 and blue<10000
      white when green>20000 and red>20000 and blue>22000
      orange when green<20000 and red>22000 and blue<22000
      pink when green<19000 and red>22000 and blue<24000
      template when red>10000 and green<14000 and blue<16000

Even though the examples from online sources only detected colors red, blue, and green, I added more colors such as black, white, orange, and pink in my project.

3. **Calibrating the Colors and Testing for Color Recognition:** The second challenge, the most time-consuming part of the project, was calibrating the colors correctly. While the color_Raw_RGB.py program is running, I recorded the raw RGB values of each color along with the template separately. Once I get a sense of the lowest and highest RGB values of a color, I move on to the next one. Once I am done calibrating them all, I would go ahead and adjust the code on color_Recognition.py by choosing the best unique range of RGB values for each color to detect each color correctly.

   For a few days, I went back and forth with calibrating and adjusting RGB ranges, then I did not get promising test results because, by the time I calibrated each color and adjusted the code, I would see that the calibration has already changed. It was frustrating for a few days.

   Then, I realized that I was working on this project in daylight and the light coming from my windows was naturally changing during the day. Instead, I covered all my windows in the room to make the room dark and then turn on the lights, hoping that I would have a steady light stream while working on this project. And I was right!! Once I calibrated the colors and adjusted the code, I was able to get the correct test results.

   Here is the demonstration video of my project:
   https://drive.google.com/file/d/1lM85SF5pR3vrAwjbpnbv_3IV-JA77_Aj/view?usp=sharing

   This demo shows that my system can detect colors red, blue, green, black, white, and orange correctly, but mix the color pink with orange due to the color range similarities between colors pink and orange.