Politecnico di Torino

Politecnico di Torio

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING (DAUIN)

MASTER'S DEGREE IN MECHATRONIC ENGINEERING

A.a. 2021/2022

Development of a Color measurement tool

Academic Supervisor:

Prof. Marco Vacca

Company Supervisor:

Eng. Maurizio Vincenti

Candidate:

Mohammed Ali Agamy Mostafa

Abstract

One of the challenges in any business is to produce a product with a good quality that satisfies customer needs, and checking the color of the product is essential to verify the quality of the product; that is why color matching and identification are essential aspects in the industrial field.

The main emphasis of this thesis project is to develop a portable and user-friendly device that can measure the color of any flat surface and provide the data in the form of digital color spaces that can be useful in the industrial field or even for personal uses and present the measuring data to the user in a proper way.

The method used to measure the colors is a spectrometer sensor; the idea of the spectrometer is to measure the wavelength of the reflected light from the material surface; the wavelengths can be mapped and translated into many color spaces. Then, the measured data will be sent to a mobile application to display it using a Bluetooth connection between the device and a smartphone.

With a six-channel spectral sensor, The device can read the red, green, and blue wavelengths; the RGB wavelengths are then translated into the RGB color space and converted to the other color spaces like XYZ, CIE-Lab, CIE-LCH, CMYK, HSL, and HSV.

A Microcontroller mounted with BLE (Bluetooth Low Energy) module will read the measures of the sensor and send them to the smartphone.

The smartphone has a mobile application developed To present and display the data. The user can interact with the device, take new measurements, or compare two samples and display the Delta. The device's size is small so that a man can hold it with one hand, and it contains a rechargeable battery to be used outside and in the industrial environment.

To conclude, the output of this thesis project is a Color measurement tool that can read and identify any color and provide the data in terms of the most famous seven color spaces used in the industrial field.

Acknowledgments

This thesis project is a collaboration with **Segula technologies company;** to create a color measurement tool that is used in the quality control for the production of any product to assure that the product color is the same as expected and compares it and an original sample showing the difference between the two colors.

Table of Contents

1. Introduction 9

1.1. Motivations 9

1.2. Thesis goals 9

1.3. Method 9

2. Background Theory 11

2.1. Spectrometer 11

How it works 11

2.2. Color spaces 12

2.2.1. RGB 12

2.2.2. XYZ 13

2.2.3. LAB 13

2.2.4. LCh 13

2.2.5. CMYK 14

2.2.6. HSL 14

2.2.7. HSV 14

2.3. Color conversions 15

2.3.1. From RGB to XYZ 15

2.3.2. From XYZ to Lab 15

2.3.3. From Lab to LCh 16

2.3.4. From RGB to CMYK 16

2.3.5. From RGB to HSL and HSV 16

3. Methodology 17

3.1. Design Requirements 17

3.2. HardWare design of the system 18
3.2.1. Color sensor selection 18
3.2.2. The Microcontroller selection 21
3.2.3. The battery selection 23
3.2.4. battery charging board 25
3.2.5. Power ON/OFF control switch 26
3.3. Electronic circuit Diagram 27
3.3.1. Power circuit 27
3.3.2. Control circuit 30
3.3.3. Programming connections 30
3.4. Mobile Application Features 31
3.5. Software algorithm design 35
3.5.1. AS2762 sensor Configuration 35
3.5.2. AS2762 sensor Calibration 37
3.5.3. Color spaces conversion 38
3.5.4. Bluetooth Data frame 39
3.5.5. Mobile app Algorithm 39
3.6. System mechanical design 41

4. Results 43
4.1. Measurement results 43
4.2. Analysis and discussion 44
4.2.1. PANTONE 19-4305 44
4.2.2. PANTONE 19-3935 44
4.2.3. PANTONE 19-3438 45
4.2.4. PANTONE 19-0825 45
4.2.5. PANTONE 19-1255 45
4.2.6. PANTONE 19-1564 46

4.2.7. PANTONE 19-1248 46
4.2.8. PANTONE 19-5722 46
4.2.9. PANTONE 19-4330 47
4.2.10. PANTONE 19-6437 47
4.3. Troubleshooting 48

5. Conclusion and future work 49
5.1. Conclusion 49
5.2. Future work 49

6. References 50

7. Appendices 51

Appendix A ESP 32 Code 51

Appendix B Mobile APP Code 60

Appendix B API JSON response code 64

List of Figures

Figure 1: diffraction of light lines 11

Figure 2:AS7262 output spectral responsitivity [15] 19

Figure 3: AS7262 sensor [15] 20

Figure 4: photodiodes array of AS7262 sensor [15] 20

Figure 5: ESP 32 devkit v1 22

Figure 6: Li-Ion Battery 23

Figure 7: Lithium-Ion Protection Circuit 23

Figure 8: discharging diagram of Li-Ion battery 24

Figure 9: TP4056 charging board 25

Figure 10: Power ON/OFF switch 26

Figure 11: Electronic circuit diagram of the system 27

Figure 12: Connection between the USB port and charging board 27

Figure 13: external LEDs connection With charging board 28

Figure 14: the power connection circuit of the system 28

Figure 15: ESP32 devkit v1 schematic [5] 29

Figure 16: programming and charging port 30

Figure 17: Bluetooth connection indication LED 31

Figure 18: Color read mode selection 32

Figure 19: measure two colors and show the difference 33

Figure 20: sharing the report using Email 34

Figure 21: AS7262 sensor reading modes 36

Figure 22: reading a color sample and showing the result 38

Figure 23: color spaces conversion method 38

Figure 24: the Data frame from the ESP32 to the mobile APP 39

Figure 25: mechanical design of the device 41
Figure 26: design of device reading hole 42
Figure 27: design and positioning of user interface 42
Figure 28: the mobile App measurements of ten samples 43
Figure 29: the first sample measurement 44
Figure 30: the second sample measurement 44
Figure 31: the third sample measurement 45
Figure 32: the fourth sample measurement 45
Figure 33: the fifth sample measurement 46
Figure 34: the sixth sample measurement 46
Figure 35: the seventh sample measurement 46
Figure 36: the eighth sample measurement 46
Figure 37: the ninth sample measurement 47
Figure 38: the tenth sample measurement 47

List of Tables

Table 1: color sensors comparison 18
Table 2: the current control mode of the color sensor LED 35
Table 3: RGB mapping lookup tables..... 37
Table 4: measurements results of 10 color samples 43
Table 5: color Distance calculation for the ten samples 47

1. Introduction

1.1. Motivations

It is challenging to identify the colors using the naked eye, as Human vision is subjective, leading to communication errors and confusion. A little brighter, a smidge darker, or a touch bluer are impossible to achieve without hours of trial and error and, in the end, will not have a quantitive measurement. Furthermore, due to the massive numbers of colors, if we consider the RGB color space, which can be represented in 24-bits, there are 16,777,216 different color possibilities, so there was a need to develop a digital tool can differentiate between the colors.
Measuring colors with an instrument such as a spectrophotometer instead of evaluating them by the naked eye can dramatically decrease the time wasted for noticing the difference between two colors and gives digital quantitive data.

1.2. Thesis goals

This project aims to develop a low-cost portable color sensing device that can measure the color of different surfaces and provide the data described digitally in terms of color spaces that are used in many industrial fields, and present the measuring data to the user in a proper way so that it can be shared or processed.

1.3. Method

The method used to measure the colors is a spectrometer sensor; the idea of the spectrometer is to emit the light against the measured surface, then measure the wavelength of the reflected light; the sensor receives the wavelengths. Then, filter it into different colors wavelengths using the optical filters; the data will be processed using a microcontroller and sent to a smartphone using a Bluetooth connection.

The device contains a six-channel spectral sensor that reads the red, green, and blue wavelengths; the RGB wavelengths are translated into the RGB color space, and from RGB will be converted to CMYK, HSL, HSV, and XYZ, which then will be converted to LAB and LCH.

The ESP32 Microcontroller is used to communicate with the sensor, acquiring the data and processing it by Calibration to RGB and conversion to the required color spaces; it is mounted with BLE (Bluetooth Low Energy) module that will send the processed data to the smartphone.

A mobile application was developed to present and display the data coming from the device. The user can interact with the device, measure one color, or compare two samples and display the Delta. Also, it allows the user to share the report of the data with any sharing tool installed on the mobile. The device's design is tiny so that it will be easy for the user to carry it and use it; the device contains a battery to use in the industrial environment without connecting the device to the PC.

2. Background Theory

In this section, the color spectrometer definition, the working principle, and an introduction about the color spaces and conversion between the color spaces will be hand-coded and used in our Application.

2.1. Spectrometer

How it works

A traditional spectrometer functions by utilizing the diffraction that occurs when a light wave hits an object; in this case, a diffraction grating occurs according to equation 2

$$dsin(\theta m) = m\lambda \quad (1)$$

(d is the distance between light lines, λ is the wavelength of the light, and θm is the diffraction angle of the light. After diffraction occurs, several beams of light will appear for each wavelength; m corresponds to the number of each beam starting from 0 along the centerline)

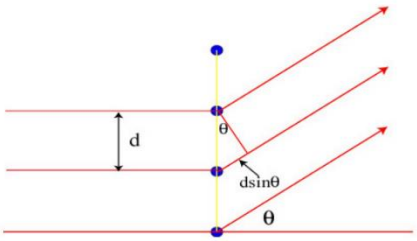


Figure 1: diffraction of light lines

Solving equation (1) for θm gives

$$\theta m = \sin^{-1}\left(\frac{m\lambda}{d}\right) \quad (2)$$

From equation 2, it becomes evident that the diffraction angle is different for each wavelength of light. It means that visible light can be spread into its wavelength/color components, acquiring a light spectrum. One can then let this diffracted light hit an array of detectors (Photodiodes) then the intensity of each wavelength can be measured.

The photodiode is considered a light sensor that converts light energy into electrical energy as current or voltage. The photodiode structure is a semiconducting device with a PN junction Between the p (positive) and n (negative) layers; an intrinsic layer is present. It receives light energy as input to generate electric current.

The photodiode contains optical filters; the objective of the optical filter is to block specific light wavelengths while permitting others to pass through it. The blocking can be done either by absorbing or reflecting the unwanted light. Placing one such filter in front of a photodiode refines the sensitivity curve of the diode, The simplest type of optical filter is a colored sheet of glass or other transparent material.

By filtering the reflected light from the measured object and forcing a specific light color to hit the Phtodidode based on the optical filters, the wavelength can be calculated, which is the color spectrometer's working principle.

2.2. Color spaces

A color space is a specific colors organization; it is a useful conceptual tool for understanding the color capabilities of any digital file or physical surface. Color space consists of a color model and a specific mapping of that model onto an absolute color space. The color model can be represented as tuples of numbers, typically three or four values like RGB and CMYK. A Gamut can describe the color space; it depicts a range of colors within the spectrum of colors or a color space identifiable by the human eye (visible color spectrum).

2.2.1. RGB

sRGB color space (standard RGB) is a color space derived from the RGB color model, is an abbreviation for (Red, Green, Blue); it describes the amount of light that needs to be emitted to produce a given color; when these three colors are combined, they create variations of other colors. RGB is described in 24 bit 8 for each color, which means a value (0 -255)

2.2.2. XYZ

The CIE-XYZ color space is one of the first color spaces to be defined mathematically, established by the International Commission on Illumination (CIE). Inside the CIE XYZ color space, the X, Y, and Z approximately correspond to the colors of red, green, and blue. The X value in the XYZ model represents the red/green part of the color, the Y value represents approximately the lightness, and the Z value corresponds roughly to the blue/yellow part. The X value range (from 0 to 95.047), the Y value from 0 to 100, and the Z accepts values between 0 and 108.883.

2.2.3. LAB

CIE-L*a*b* color space (also called CIE-LAB) is one of the most popular color spaces for measuring object colors. CIE defined it for color communication and is widely adopted today in many color control and management industries. In the L*a*b* color space, L* indicates lightness represented on a vertical axis with values from 0 (black) to 100 (white), and a* and b* are chromaticity coordinates. a* and b* are color directions: +a* (positive) and -a* (negative) indicate red and green values, respectively. +b* is the yellow axis, and -b* is the blue axis. The vertical distance from the origin represents the saturation of the color. Moreover, the angle on the chromaticity axes represents the hue.

2.2.4. LCh

The L*C*h CIE color space is similar to the LAB space, and most industry professionals prefer it because its system correlates well with how the human eye senses color.
It has the same L*a*b* color space diagram but uses cylindrical instead of rectangular coordinates. L* indicates lightness in this color space, C* represents chroma, and h is the hue angle. The value of chroma C* is the distance from the lightness axis (L*) and starts at 0 in the center. Hue angle starts at the +a* axis and is expressed in degrees (0° is +a*, red, and 90° is +b or yellow).

2.2.5. CMYK

CMYK space stands for Cyan, Magenta, Yellow, and Black (Key) is a subtractive color space based on the primary colors of printing. The CMYK color model derives from the inefficient use of the RGB color model in the printing process as in RGB, inks of (red, green, blue) will be mixed to get white, which is usually the most dominant color of the painting paper containing text. So CMYK color space works opposite to the additive RGB color space. In CMYK space, inks subtract brightness from white (cyan, magenta, yellow and black are subtractive primaries) by layering cyan, magenta, yellow and black to achieve a specific hue.

2.2.6. HSL

Unlike RGB and CMYK, which base on primary colors, HSL is more relative to how humans perceive color; HSL is an acronym for hue, saturation, and lightness; they describe the color space as a double-cone or sphere, the vertical axis represents all shades of gray from 0 (black) to 1 (white). All fully saturated colors then lie on the outer circle of a horizontal cross-section with middle gray at its center, allowing the hue to be defined as an angle. The third parameter, saturation, corresponds to the circle radius around the vertical axis at the position of the current lightness.

2.2.7. HSV

HSV follows the same concept as HSL; it visually represents colors for the human eye. H(Hue) is the color portion of the model, expressed as a number from 0 to 360 degrees, S (Saturation) describes the amount of gray in a particular color, from 0 to 100 percent, V(Value or Brightness) describes the brightness or intensity of the color, from 0 to 100 percent. The HSV color space diagram appears as a wheel, cone, or cylinder, but always with these three components.

2.3. Color conversions

2.3.1. From RGB to XYZ

$$R_{linear} = \begin{cases} \frac{R_{sRGB}}{12.92}, & R_{sRGB} \leq 0.04045 \\ \left(\frac{R_{sRGB}+0.055}{1.055}\right)^{2.4}, & R_{sRGB} > 0.04045 \end{cases}$$

$$G_{linear} = \begin{cases} \frac{G_{sRGB}}{12.92}, & G_{sRGB} \leq 0.04045 \\ \left(\frac{G_{sRGB}+0.055}{1.055}\right)^{2.4}, & G_{sRGB} > 0.04045 \end{cases}$$

$$B_{linear} = \begin{cases} \frac{B_{sRGB}}{12.92}, & B_{sRGB} \leq 0.04045 \\ \left(\frac{B_{sRGB}+0.055}{1.055}\right)^{2.4}, & B_{sRGB} > 0.04045 \end{cases}$$

$$\begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} + \begin{bmatrix} R_{linear} \\ G_{linear} \\ B_{linear} \end{bmatrix}$$

$$X_{D65} = ((R_{linear} * 0.4124) + (G_{linear} * 0.3576) + (B_{linear} * 0.1805))$$
$$Y_{D65} = ((R_{linear} * 0.2126) + (G_{linear} * 0.7152) + (B_{linear} * 0.0722))$$
$$Z_{D65} = ((R_{linear} * 0.0193) + (G_{linear} * 0.1192) + (B_{linear} * 0.9505))$$

2.3.2. From XYZ to Lab

$$L = 116 f_y - 16$$
$$a = 500(f_x - f_y)$$
$$b = 200(f_y - f_z)$$

where

$$f_x = \begin{cases} \sqrt[3]{x_r} & if\ x_r > \varepsilon \\ \frac{K x_r + 16}{116} & Otherwise \end{cases}$$

$$f_y = \begin{cases} \sqrt[3]{y_r} & if\ y_r > \varepsilon \\ \frac{K y_r + 16}{116} & Otherwise \end{cases}$$

$$f_z = \begin{cases} \sqrt[3]{z_r} & if\ z_r > \varepsilon \\ \frac{K z_r + 16}{116} & Otherwise \end{cases}$$

$$x_r = \frac{X}{X_r}, \quad y_r = \frac{Y}{Y_r}, \quad z_r = \frac{Z}{Z_r}$$

$X_r = 95$, $Y_r = 100$, $Z_r = 108$, $\varepsilon = 0.008856$, $K = 909.3$ (CIE standard)

2.3.3. From Lab to LCh

$$L = L$$

$$C = \sqrt{a^2 + b^2}$$
$$h = \arctan\left(\frac{b}{a}\right)$$

$$H = \begin{cases} \frac{h}{\pi}180 \\ 360 - \frac{|h|}{\pi}180 \end{cases}$$

2.3.4. From RGB to CMYK

$$R' = \frac{R}{255} \quad , G' = \frac{G}{255} \quad , B' = \frac{B}{255}$$
$$K = 1 - Max(R', G', B')$$
$$C = \frac{(1-R'-K)}{(1-K)}$$
$$M = \frac{(1-G'-K)}{(1-K)}$$
$$Y = \frac{(1-B'-K)}{(1-K)}$$

2.3.5. From RGB to HSL and HSV

$$Cmax = Max(R', G', B') \ , \qquad Cmin = min(R', G', B'), \ \ \Delta = Cmax - Cmin$$

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60\left(\frac{G'-B'}{\Delta} mod 6\right) & Cmax = R' \\ 60\left(\frac{B'-R'}{\Delta} + 2\right) & Cmax = G' \\ 60\left(\frac{R'-G'}{\Delta} + 4\right) & Cmax = B' \end{cases}$$

$$For\ HSV:\ S = \begin{cases} 0 & \Delta = 0 \\ \frac{\Delta}{1-|2L-1|} & otherwise \end{cases} \ , \ For\ HSV:\ S = \begin{cases} 0 & Cmax = 0 \\ \frac{\Delta}{Cmax} & otherwise \end{cases}$$

$$L = \frac{Cmax+Cmin}{2}$$
$$V = Cmax$$

3. Methodology

This chapter describes the design and implementation of the proposed device and overall methodologies. First, section 3.1 shows the system's design requirements; section 3.2 describes the selection of hardware components. Next, section 3.3 discusses the electronic circuit and connections of the system; Section 3.4 shows the feature of the mobile Application; section 3.5 includes the software algorithms design. Finally, section 3.6 describes the mechanical design of the system.

3.1. Design Requirements

For developing a low-cost portable color measurement tool, the following design requirements have been set:

- The system should easily match and identify any color with a simple scan
- The system should be low-cost
- Low Power consumption
- The developed system should provide the user with the different color spaces used in the industrial field
- Perform quality analysis (Read the color) with wireless measurement solution
- Quickly review and export the measured data
- Estimate the closest known color to the measured one calculating the distance if there is a difference
- Compare between two samples and calculate the Delta
- The device should be portable and lightweight to be carried around the field
- The device should be rechargeable to use outside in the working environment.

3.2. HardWare design of the system

3.2.1. Color sensor selection

The objective of the thesis is to create a low power consumption and portable device that can provide the user with the color code of the measured surface; based on that, it was significant to select a suitable color sensor.

A study was done, comparing the available sensors in the market, showing if it is suitable for the device.

Sensor	company	Output data (color space)	Comm. Protocol	ESP 32 comp (Drivers)	Size	Price
TCS230/TCS3200	(TAOS)	RGB	Direct Analog read	yes	accepted	8 €
ISL29125	Sparkfun	RGB	I²C	yes	accepted	9 €
AS7262	Sparkfun	6 visible channels (V,B,G,Y,O,R)	UART or I²C	yes	accepted	27 €
AS7261	MIKROE	XYZ,NIR	UART or I²C	No	Not	27€
AS7261 DEMO KIT	AMS	XYZ,NIR	UART or I²C	Partially supported	accepted	109€
AS7264 Demo Kit	AMS	XYZ, NIR,B440,B490	UART or I²C	Partially supported	accepted	133€
AS7265x	Sparkfun	18 channels	UART or I²C	yes	accepted	60€

Table 1: color sensors comparison

The size of the sensor matters as, in the end, it will impact the final size of the device, so it was essential to select a sensor with a geometric shape and has good fixing points to be easy to assemble; that is why for instance, the AS7261 MIKROE sensor was excluded as it has a non-geometric and big size.

The sensor price should be economical as the price is crucial for this project, so high price sensors were also excluded.

The sensor's output should be one of the famous color domains to be easily converted to the other color domains required by the market, and the most used two are RGB and XYZ.

At the end of this study, the AS7262 Sparkfun sensor was selected based on the design requirements; an additional advantage, Sparkfun also provides a library that supports the Microcontroller, which will acquire the data from the sensor using the I2c protocol.

The AS7262 sensor has A 6-channel multispectral sensor in the visible range of around (430 nm to 670 nm) with a full-width half-max (FWHM) of 40 nm. The six visible channels are 450 nm (channel V), 500 nm (channel B), 550 nm (channel G), 570 nm (channel Y), 600 nm (channel O), and 650 nm (channel R).

This study will use the R, G, and B channels to provide the RGB color domain, and it will be easy to convert to all the other required domains.

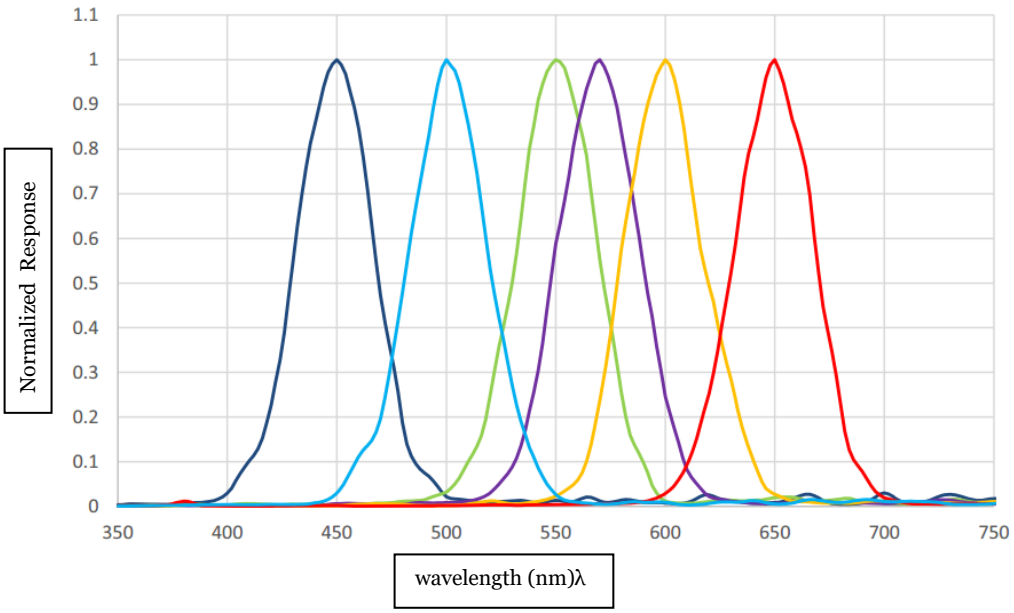


Figure 2:AS7262 output spectral responsitivity [15]

The sensor is mounted on a spectral breakout board (SparkFun number SEN-14347) to integrate the sensor. It has a 16-bit ADC (Analog to Digital Converter). Moreover, its operating voltage ranges from 2.7V to 3.6V. Also, it has an I2C register set by which spectral data can be accessed.
The temperature range of the sensor is -40°C to 85°C temperature, which is very suitable for our Application to use the device outside in the industrial environment or even for personal uses.

Figure 3: AS7262 sensor [15]

The sensor has a built-in white LED connected to a LED driver output that can be used to control the LED. This method will allow different wavelength light sources to be used in the same system. In addition, the LED output sink currents are programmable and can drive external LED sources. The current can be set to 12.5mA, 25mA, 50mA, and 100mA. Also, it can be turned off and on Using the I²C registers control.
The reflected light from the measured surface will hit an array of photodiodes in the AS7262. There are six photodiodes G, Y, B, O, V, and R

The AS7262 can communicate by both the I2C and UART serial interfaces; The I2C is the default setting (The default I2C address is **0x49**). UART can be enabled easily by removing solder from the jumper on the board adding solder to another t jumper.

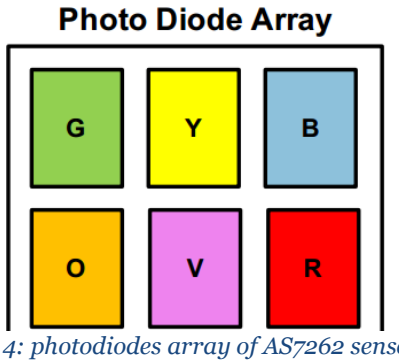


Figure 4: photodiodes array of AS7262 sensor [15]

3.2.2. The Microcontroller selection

The selection criteria of the Microcontroller are the shape, size, and memory capabilities; moreover, it should contain an I2c Communication Protocol to communicate with the AS7262 sensor. Again the size is essential; the selection was between Arduino nano, Nodemcu ESP 8266, and ESP 32.
In the end, ESP32 was selected because it has all the specifications required for the project.

ESP32

ESP32 is a SOC (System on Chip) microcontroller; it has Dual-Core 32-bit LX6 microprocessors, which run up to 600 DMIPS. The ESP32 Working frequency is settable between 80 MHz and 240 MHz.
The first core is named Protocol CPU, and the second one is Application CPU. The Protocol CPU processor handles the WiFi, Bluetooth, and other internal peripherals like SPI, I2C, ADC. on the other side, The Application CPU is left out for the application code.

Memory

It has a 448 Kb ROM for core functions and booting and 520 Kb on-chip SRAM for instruction and data. In addition, a flash memory of 4Mb is used for storing the application code.

Physical characteristics

The Operating temperature (-40 to 85) degrees Celsius and Operating Voltage of (3.3V) Moreover, the current of (80 mA) and finally, the module dimensions is (51 x 28 x 14 mm).

ESP32 Microcontroller is a perfect selection for our Application based on the previous specifications. The most valuable advantage is that the ESP contains a built-in BLE (Bluetooth low energy) module, So instead of using, for instance, an Arduino nano or ESP8266 with a separate Bluetooth module like hc-05 or hc 06 instead, ESP32 is used, and it contains BLE, not just the standard Bluetooth module and ESP32 supports both Bluetooth 4.0 (BLE / Bluetooth Smart) and Bluetooth Classic (BT).

BLE (Bluetooth low energy)

Bluetooth Low Energy, BLE for short, is a power-conserving variant of Bluetooth. BLE's primary Application is the short-distance transmission of small amounts of data (low bandwidth). Unlike Bluetooth, which is always ON, BLE remains in sleep mode constantly except when a connection is initiated.

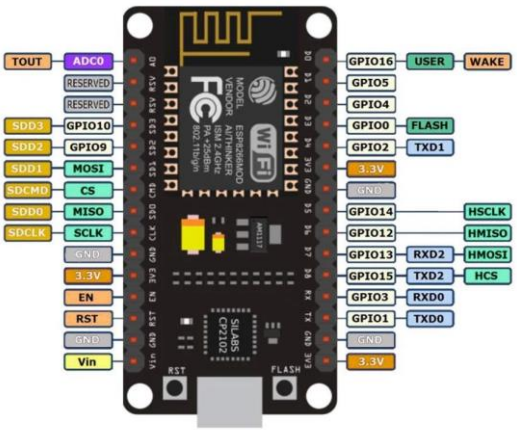


Figure 5: ESP 32 devkit v1

3.2.3. The battery selection

The operating Voltage of the ESP32 and the AS7262 color sensor is 3.3V, so it was convenient to select a battery with a voltage higher than 3.3V to avoid any voltage drop or unwanted operating behaviors. In addition, the battery current should be above 800mA to be suitable for our loads; the battery should be rechargeable to be used efficiently, small size, and lightweight to be suitable for our case design.

One of the essential requirements is that the battery should be recharged using USB power 5v, so it was not suitable to select a battery with a voltage above 5v.

The lithium-Ion cells are a perfect choice based on the specifications mentioned above due to their lightweight, small size, and operating power.

Battery specifications

- Size 48.5mm x 30mm x 10mm
- Nominal Voltage 3.7v
- Maximum current 1.6A
- Weight 31g
- Charging ending voltage 4.2 v
- Discharging ending voltage 2.75 v
- Maximum charging current 0.5 C (5 ° ~ 50 °)
- Maximum discharging current 0.5 C (-20 ° ~ 50 °)
- Battery protection circuit (PCM)

Figure 6: Li-Ion Battery

Lithium-Ion Protection Circuits (PCM) provides voltage cut-off security for Lithium rechargeable batteries. The circuit will stop the battery from overcharging or discharging too far.

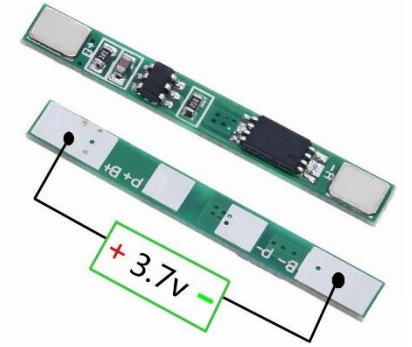


Figure 7: Lithium-Ion Protection Circuit

The AS7262 color sensor works as mentioned in the previous section by emitting light on the object then measuring the reflected light; the light source is the built-in LED on the sensor.

It is crucial to have stable light intensity during the measuring to avoid changing the sensor reading, so it was essential to understand the discharging behavior of lithium-ion batteries.

The C-rate represents the level at which the battery provides the energy; the higher power, the higher the discharge rate; 1 C means that the battery is full charged and discharged within one hour; our battery is 0.5 c, the discharge rate is 0.5c*1.6Ah=0.8Ah which means the battery is discharged 800mAh per hour and could operate around 2 hours.

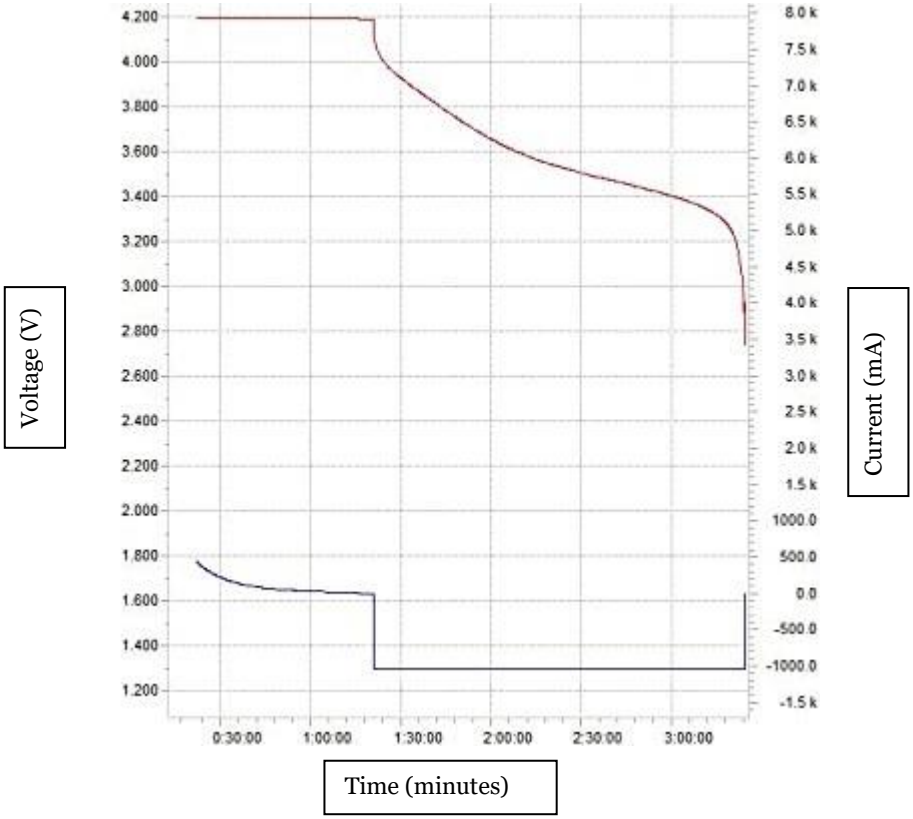


Figure 8: discharging diagram of Li-Ion battery

It is evident from the curve that the device must operate at the flat area of the curve, which means the first 2 hours if the rate is 800mA to avoid the voltage drop or unwanted behavior.

3.2.4. battery charging board

The TP4056 module is used to charge the lithium rechargeable battery; it uses the constant-current/constant-voltage (CC/CV) charging method. In addition to safely charging a lithium battery, the module also provides the necessary protection required by lithium batteries.

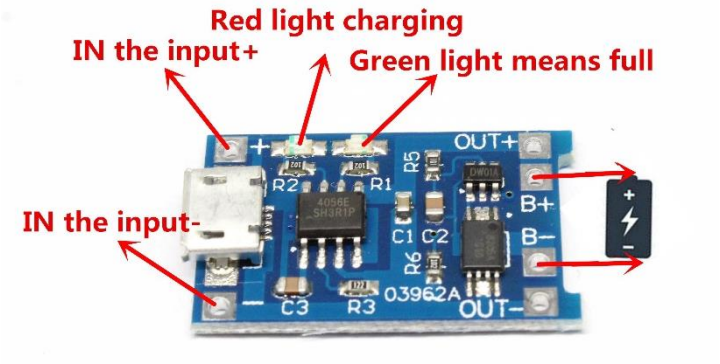


Figure 9: TP4056 charging board

The charging module uses both the TP4056 and the DW01A Li-Ion battery protection IC, which together in combination, can give high protection for the battery:

- Control the constant current to constant voltage charging of the connected lithium battery
- Over-discharge protection: keeps the battery from being discharged below 2.4V, which is the healthy minimum voltage level for the battery.
- If the battery is discharged below 2.4V, the module will Turn off output power until the battery voltage recharge above 3.0V (the over-discharge release voltage); After that, the module will again allow power discharge from the battery to the connected loads
- Overcharge protection: the module will safely charge the battery to 4.2V without exceeding that limit
- Overcurrent and short-circuit protection: the module will cut off the output from the battery if the discharge rate exceeds 3A or if there is a short circuit.

- Trickle charge: (battery reconditioning) If the connected battery's voltage level is below 2.9V, the module will use a 130mA trickle charge current when the battery's voltage reaches 2.9V. The charge current will be increased linearly till the average charging current.

The module Can be powered from a micro USB cable or the board terminals. However, the power source needs to provide at least 1A for the charger to charge the connected battery correctly.

The module contains two indicator LEDs; a red LED indicates the charging state, and a blue LED indicates the charging completed state.

The module contains six terminals, two for the input power charging (which can be neglected if the USB port is used), two to connect the battery, and finally, two to connect the loads for discharging.

3.2.5. Power ON/OFF control switch

A push-button was used to turn ON and OFF the device; it is connected to the positive wire of the battery in series; the button operates in ON-OFF-ON mode, which means that one push to turn on the device and another one to switch OFF, The push-button switch contains four terminals; two for the switch and two terminals for the Indicating LED.

Figure 10: Power ON/OFF switch

The led will illuminate green when power is ON and will be off when switching the device OFF.

3.3. Electronic circuit Diagram

The diagram is created using fritzing it is, an open-source software tool that makes electronics circuits and diagrams with the possibility to convert them into PCB (printed circuit board).

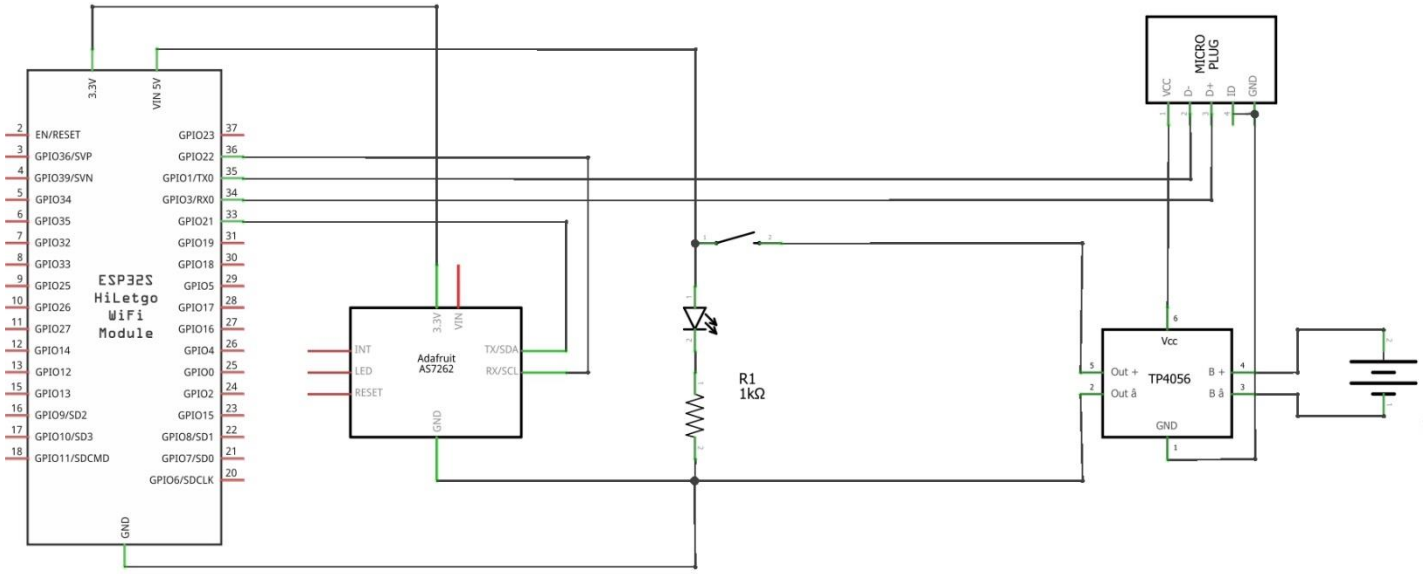


Figure 11: Electronic circuit diagram of the system

3.3.1. Power circuit

As shown in the diagram, a micro USB port was used; it is connected to the input power terminals TP4056, so the USB power port of the module is not used; the output power terminals of the recharging module is connected directly to the Lithium battery.

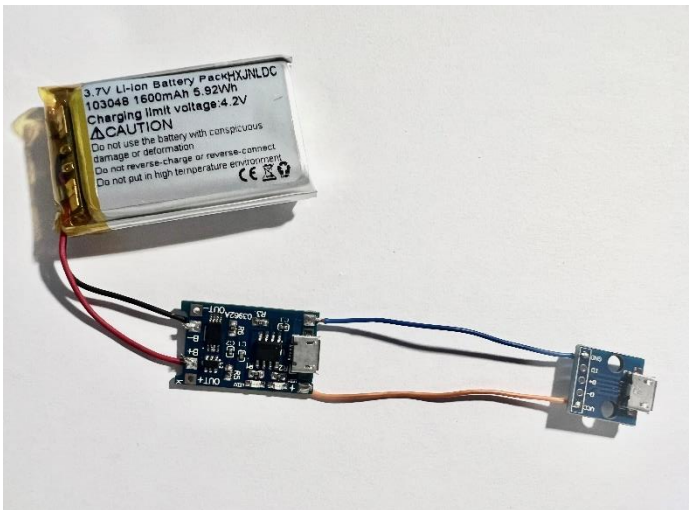


Figure 12: Connection between the USB port and charging board

The two indication LEDs of the TP4056 module were removed because it is hidden inside the dives; instead, two external LEDs were fixed to be visible to the user what is the charging state is.

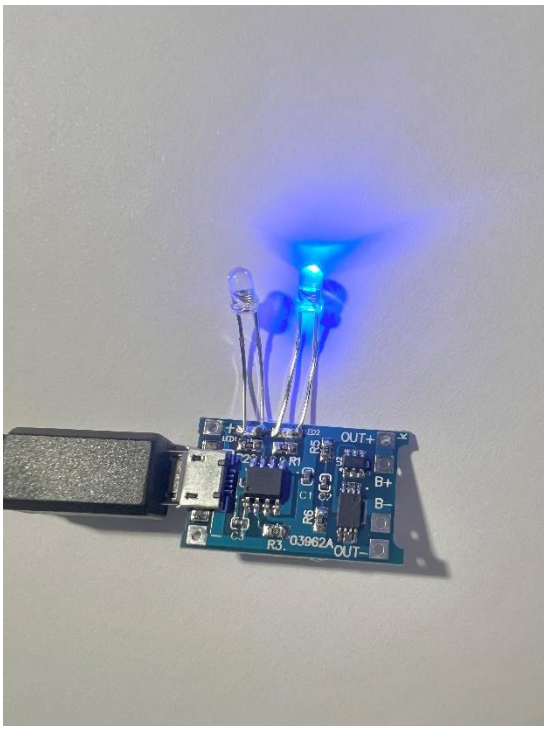


Figure 13: external LEDs connection With charging board

The battery is connected directly to B+ and B- of the charging board. The push-button terminal is connected to the OUT+ of the charging board in series, and the other pin is connected to both the indication LED positive pin and the Vin pin of the ESP32 that will power up the board. The negative pin of the LED is connected to the OUT- terminal of the board.

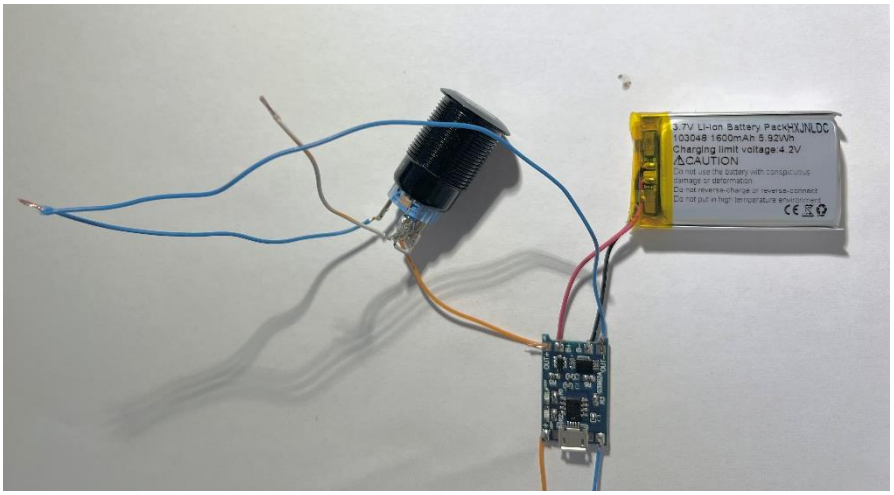


Figure 14: the power connection circuit of the system

The ESP32 board contains an ams1117 3.3 voltage regulator used to power up the ESP module itself, it converts the input voltage into 3.3v, and it is connected to a GPIO 3v3 pin; the 3.3 voltage regulator is also used to power up the as7262 color sensor.

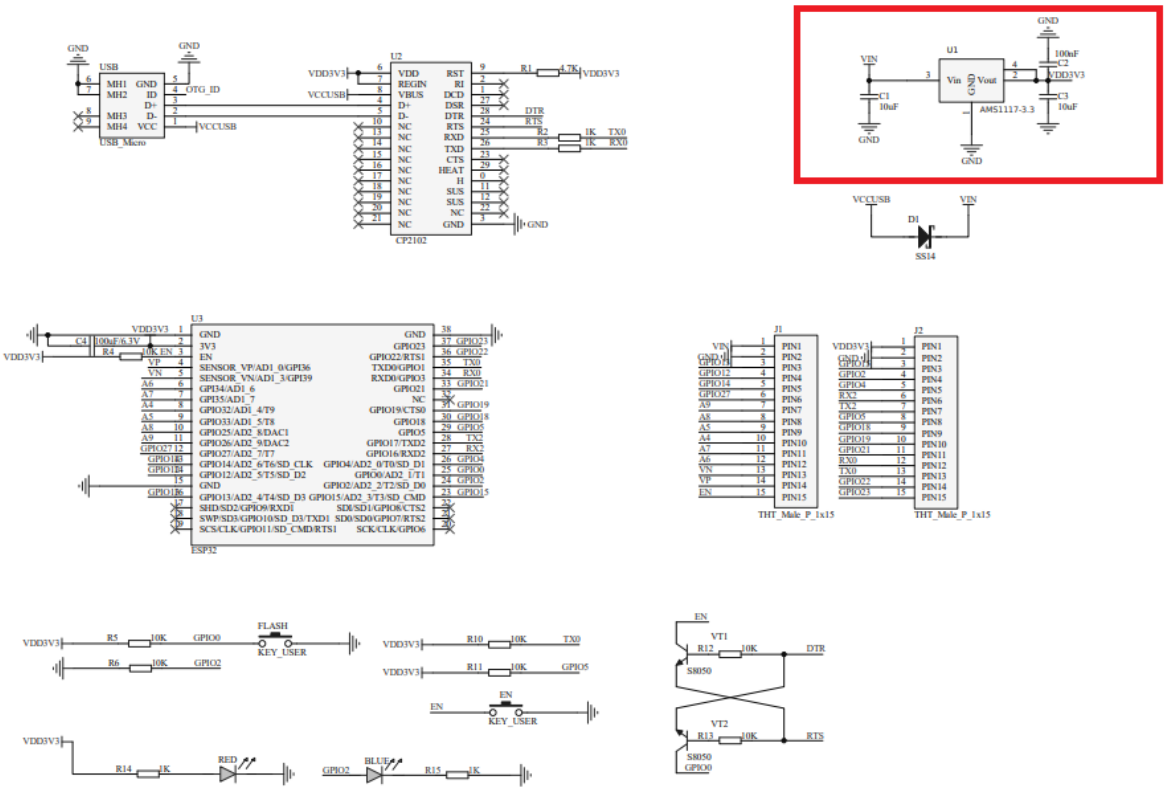


Figure 15: ESP32 devkit v1 schematic [5]

3.3.2. Control circuit

The communication between the ESP32 and the color sensor module is set up using the I2c protocol, so the SDA and SCL pins of the color module are connected to GPIO 21 and GPIO 22, related to the I2c protocol.

3.3.3. Programming connections

A male micro USB port is attached to the ESP32, and the output wires of the port are connected to wires of the main input USB female port that is used to charge the device; with this connection, it will be available to reprogram and update the code of the ESP32 without disassembling the device.

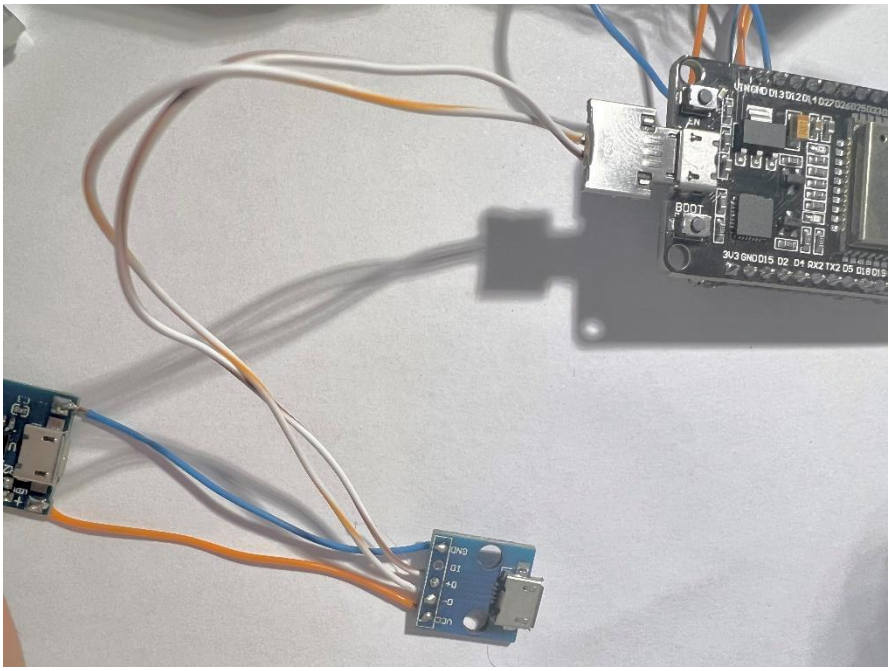


Figure 16: programming and charging port

3.4. Mobile Application Features

In order to monitor the data coming from the Microcontroller, there was a need to develop a mobile application; the APP will connect the Mobile to the device using Bluetooth communication.

It will help the user communicate with the device and read the colors or even compare between two colors and show the Delta.

The app was developed using MIT App Inventor Platform; it is an open-source platform. It was created by MIT university to develop mobile apps using the scratch programming language.

The app's first step is to verify the Bluetooth connection using a virtual led; when the Bluetooth connection is off, it is red, and when the connection is set up correctly, it turns green.

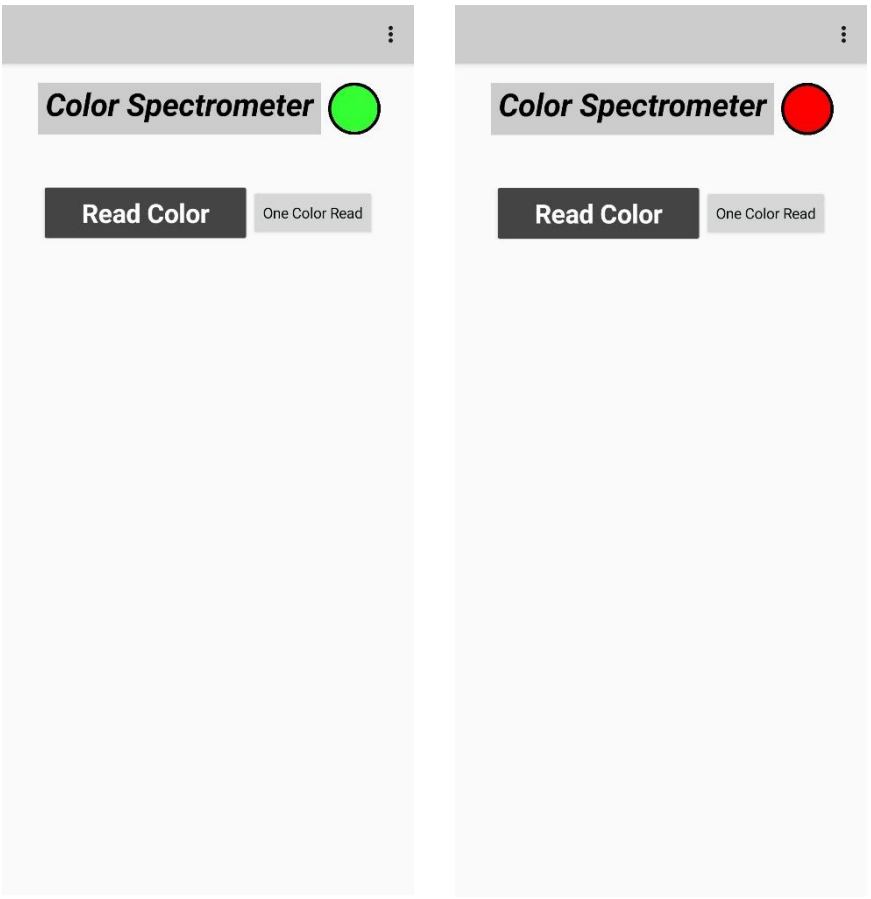


Figure 17: Bluetooth connection indication LED

The app provides two color reading modes; One read color mode and Second color mode, and the mode selection is from a list besides the Read color button.

One read color mode means taking the measurements of a single color.

the measurement contains eight color spaces presenting the RGB, RGB Hex, XYZ, LAB, LCH, CMYK, HSL, HSV

it also provides the user with a visual presentation of the measured color and the name of the closest named color to the measured one; it also provides the HEX RGB code of the new color and the distance between the two colors.

If there is a match state, a named color is precisely like the measured one; otherwise, the app will give a "No match color" message with the closest color.

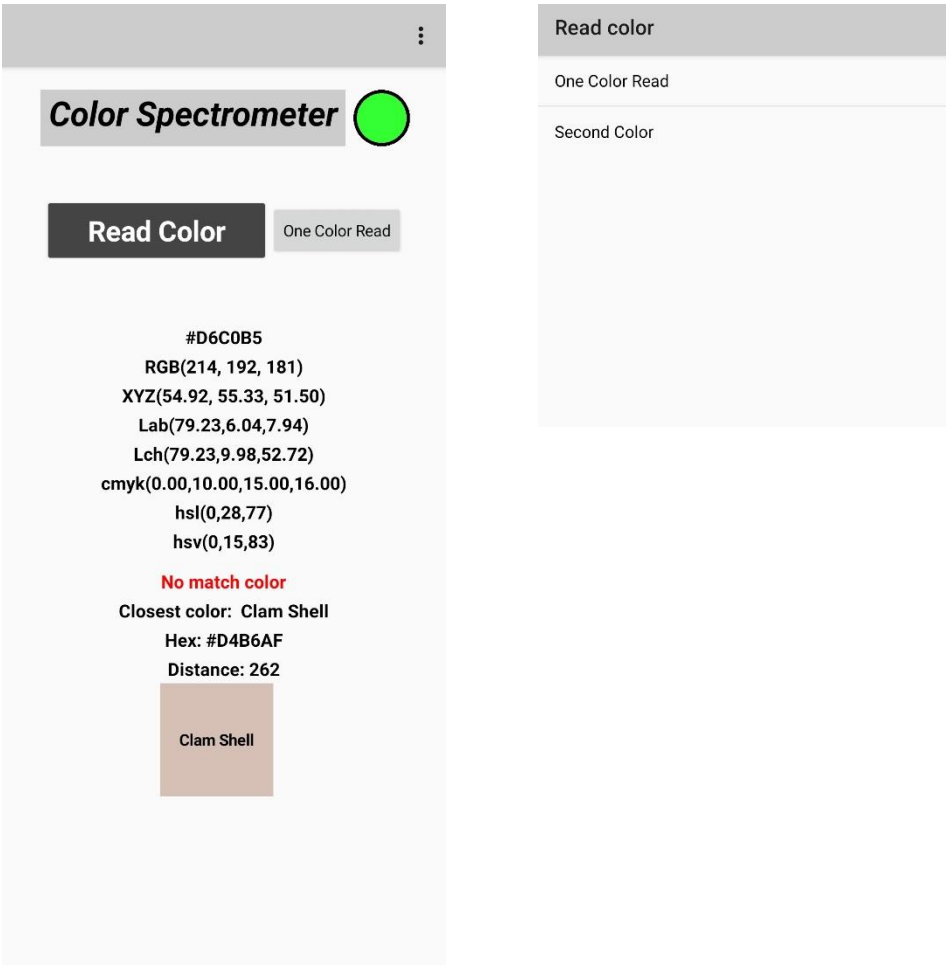


Figure 18: Color read mode selection

The second read mode takes the measurements of a second color and enables the comparison with the first color showing the difference.

After selecting the mode from the list and pressing the Read color, a Compare button will appear automatically.

If the compare button is pressed, the app will calculate the difference between the two colors The app died the data into three sections; the first section contains the data of the first color, the second section for the econ, and the last section for the difference.

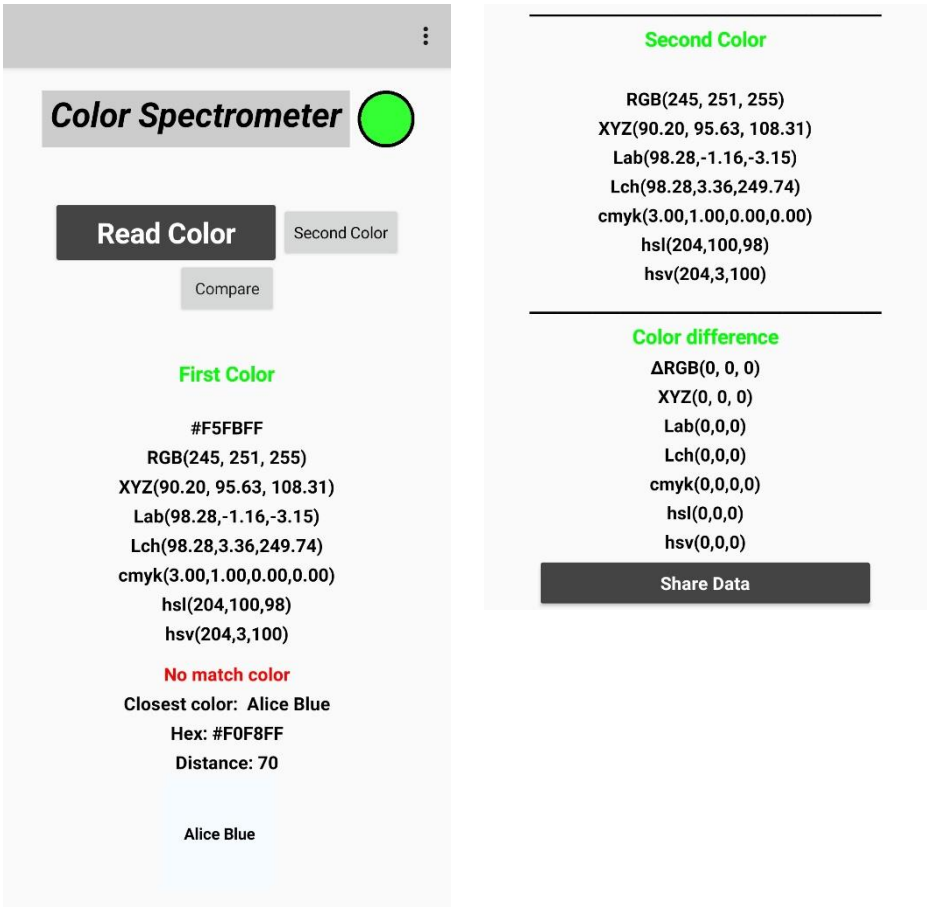


Figure 19: measure two colors and show the difference

There is a possibility to share the data by clicking the share button at the end of the app; it allows sharing all the measured data by Email or any sharing tool.

The shared data is in the form of a report showing the color and deference measurements.

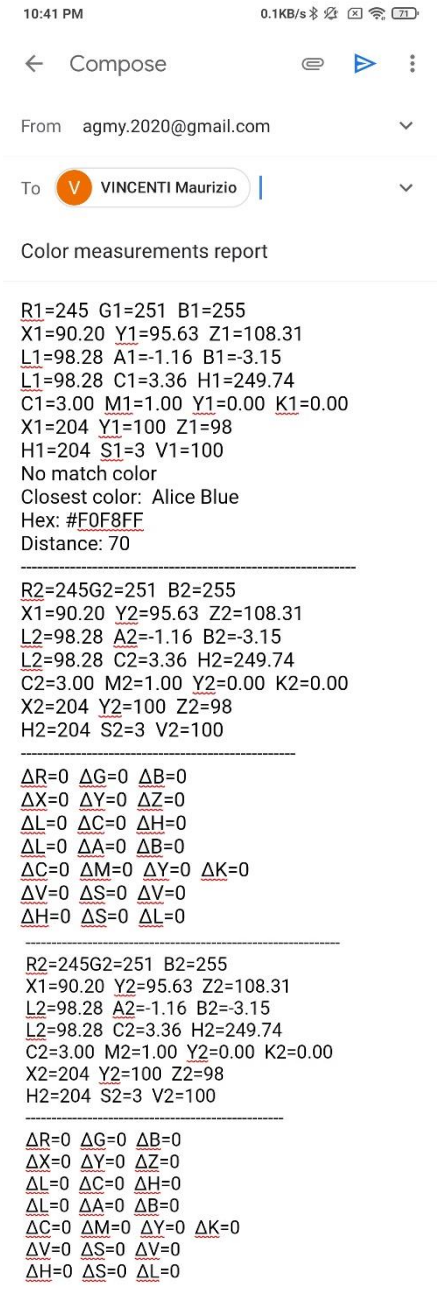


Figure 20: sharing the report using Email

3.5. Software algorithm design

The tool used to program the ESP 32 is Arduino IDE; it is a robust open-source developing environment.

3.5.1. AS2762 sensor Configuration

Sparkfun provides a library to configure and take the measurements of the sensor
As discussed in previous sections, the sensor is mounted with a 5700K LED to reduce power consumption; the current can be controlled using a dedicated function from the SparkFun library.

```
sensor.setBulbCurrent(0);
```

The current mode is set to zero, which means the current will be a minimum value of 12.5mA.
the value can be set according to the table:

0	1	2	3
12.5 mA	25 mA	50 mA	100 mA

Table 2: the current control mode of the color sensor LED

Configuring the time the sensor takes a sample is done with a function that can be set by changing the input value between(0-255) multiplied by 2.8ms.

```
sensor.setIntegrationTime(200);
```

$$IT = 200 * 2.5\ ms = 500\ ms$$

It means that the sensor will take a sample every 500 mA
This calculation is for one spectral conversion and only if the required data exists in one data Bank.

The AS7262 provides V, B, G, Y, O, and R (Violet, Blue, Red, Green, Yellow, and Orange) data. The data are available after the spectral conversion; data is stored in two

photodiode Banks; Bank 1 contains data of the V, G, B, Y photodiodes. Bank 2 consists of G, Y, O, R photodiodes data. Spectral conversion requires an entire (IT) integration time. If both photodiode banks are required to complete the conversion, the second Bank requires an additional IT. Minimum IT for a single bank conversion is 2.8 ms. If data is required from all six photodiodes, the device must perform two total conversions (2 x Integration Time).In the project, the required data are R, G, and B so that both banks will be used, and the integration will be the double

$$Total_{IT} = 2 * IT = 1000\ ms$$

So it needs one second to acquire the data from the sensor and use it for our Application Moreover, to have the date of the SIX channels (after two IT), the measurement Mode should be set to mode2 using this function: `setMeasurementMode(byte mode);`

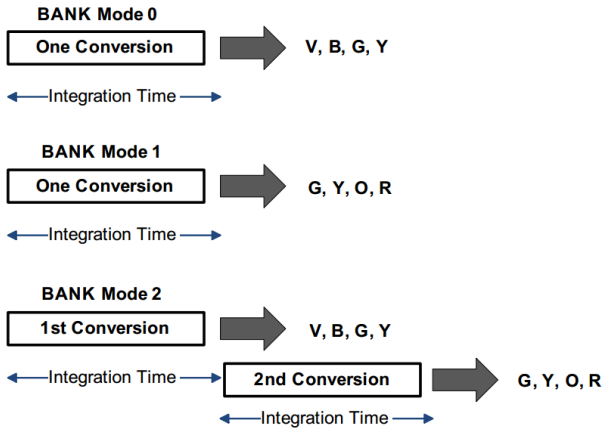


Figure 21: AS7262 sensor reading modes

to take the measurements, the `takeMeasurementsWithBulb();` function is used; it takes measurements by Illuminates the onboard bulb; then the required three needed channels are fetched using;

```
float getCalibratedRed();
float getCalibratedBlue();
float getCalibratedGreen();
```

calibrated values mean that it is Lifetime-calibrated sensing with minimal drift over time or temperature, and it is afloat more precise data than uncalibrated values.

3.5.2. AS2762 sensor Calibration

The AS7262 provides the wavelength response of six channels in this project; we will use the R, G, and B channels.

The values are calibrated using lookup tables to map the output values of the sensor into the RGB color space.

R_SnsrOut	R_mapped
2.69	0
144.52	0
159.78	38
306.99	120
489.21	204
754.01	246
781.62	255

G_SnsrOut	G_mapped
6.87	0
104.22	49
176.36	107
295.47	130
308.07	132
576.05	169
973.44	195
1037.57	224
1370.83	251
1392.77	255

B_SnsrOut	B_mapped
5.22	0
139.67	69
271.51	139
454.25	162
489.49	180
616.11	173
736.20	180
882.40	220
1156.51	255

Table 3: RGB mapping lookup tables

On the left is the output wavelength response of the sensor; on the right is the True Value of the RGB color space. The actual values are measured using Pantone 100 postcards; it is 100 colors with the given RGB code for each color.

The lookup tables will take the raw input value of the sensor and convert it into RGB color space; if the value does not exist, it will linearly interpolate the two values that the input exists between and give the output.

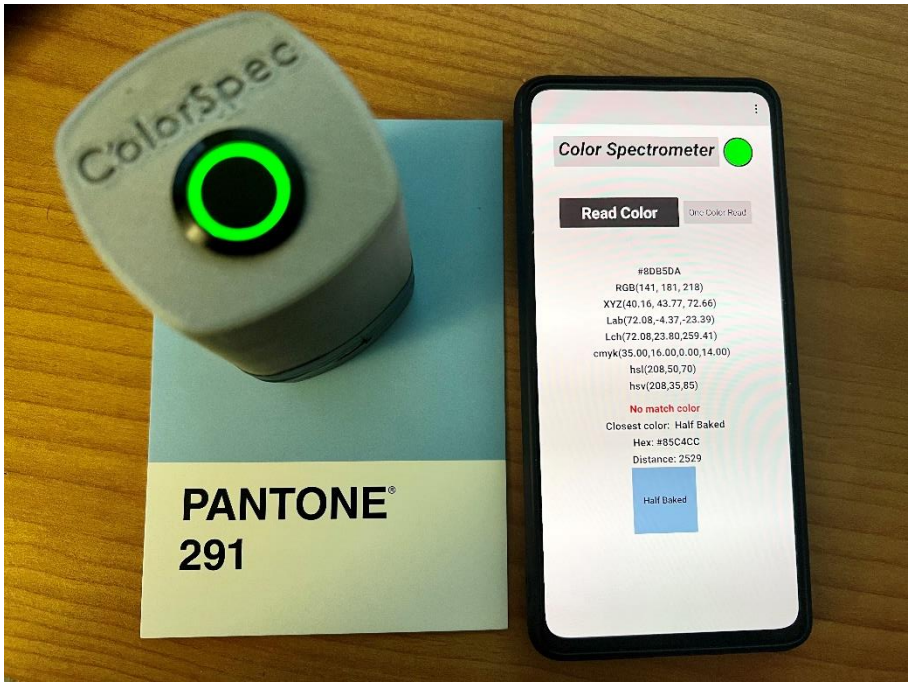


Figure 22: reading a color sample and showing the result

3.5.3. Color spaces conversion

After mapping the sensor output to the RGB color space, it is also required to convert to the other color spaces;

The RGB will be converted to CMYK, HSL, HSV, and XYZ, which then will be converted to LAB and LCH.

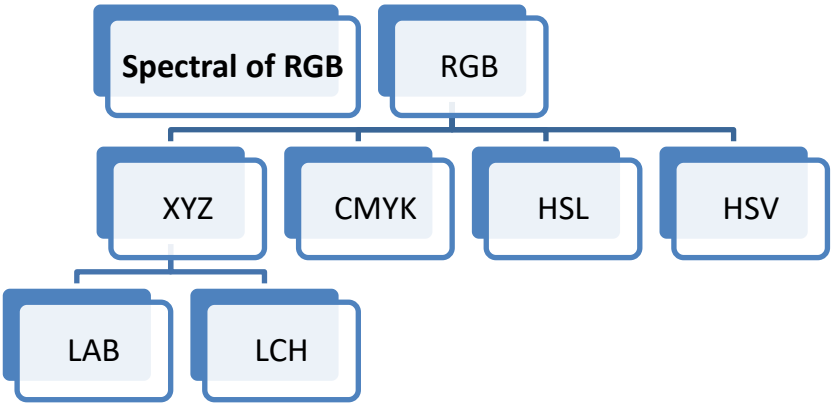


Figure 23: color spaces conversion method

The conversion methods are coded in ArduinoC language based on the formulas explained in the previous sections.

3.5.4. Bluetooth Data frame

When the data is ready, it will be sent to the Mobile phone Via Bluetooth; the data consist of one color measurement described in 7 color spaces; each color space contains three values except the CMYK, which contains four value; at the end, we have 20 value if we neglect the shared values.

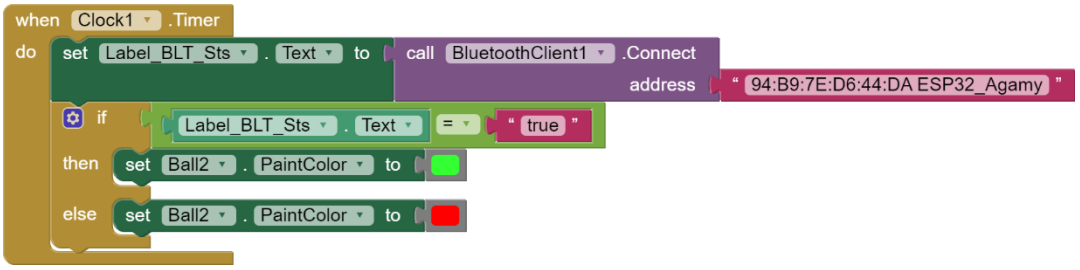
The data will be encapsulated into one text frame; the values are separated by a comma so that it will be easy for the mobile app to take the frame and split the data with the comma.

	RGB			XYZ			LAB			LCH		CMYK				HSL			HSV		
"	R,	G,	B,	X,	Y,	Z,	L,	A,	B,	C,	H,	C,	M,	Y,	K,	H,	S,	L,	S,	V,	"

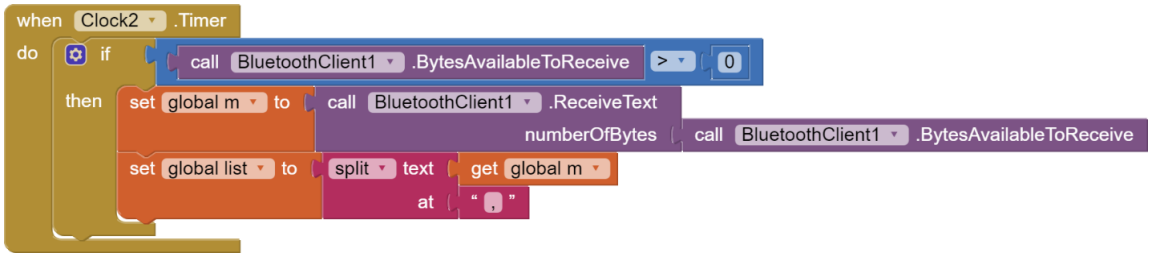
Figure 24: the Data frame from the ESP32 to the mobile APP

3.5.5. Mobile app Algorithm

The mobile app will connect to the ESP Bluetooth module directly using the Address of the module, and it will give an indication with a led (it is essential to pair with the ESP32 Bluetooth from the setting of the Mobile).

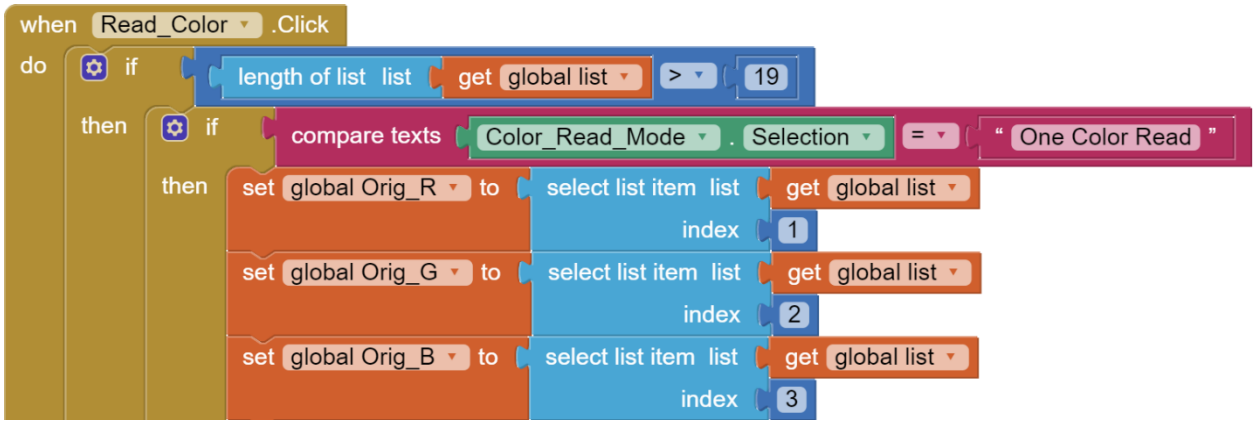


After setting up the connection, the app will check if there is data available to receive and store it in a global variable that will split it into 20 values.

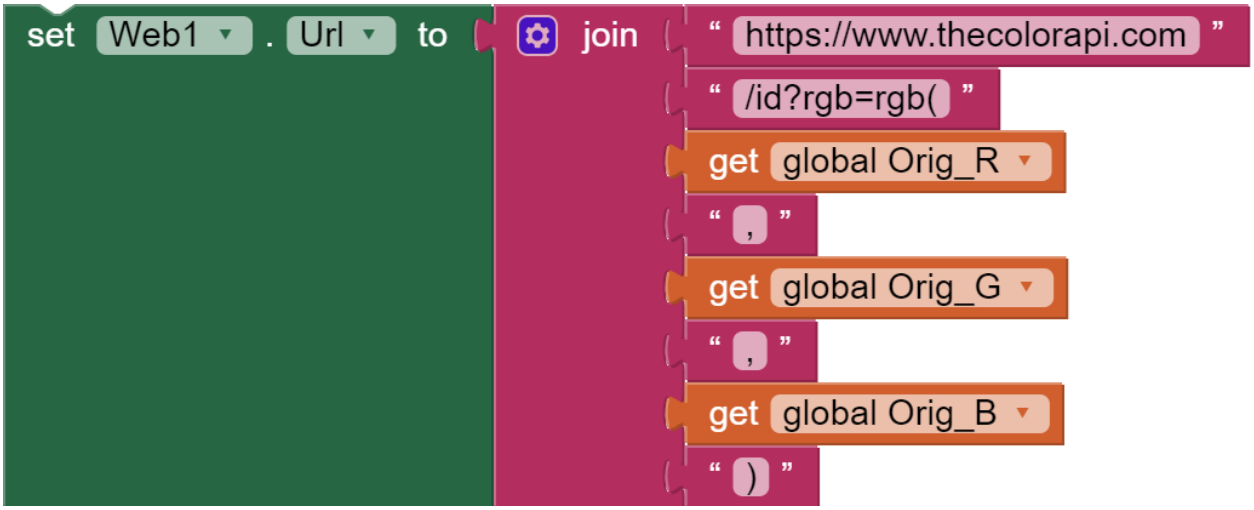


It is also essential to check that we have 20 elements in the list before using this data in the reading modes; the execution of this step is done immediately after clicking the Read Color button to avoid any garbage in the Bluetooth channel.

The 20 elements will be displayed on the screen based on the read mode selected by the user, whether Read only one color or Compare two colors.



An API (application programming interface) will take the RGB code and give the nearest named color to the measured one.



The response code of the app contains vast data about the color, but the only used piece of information is the name of the color and the HEX code conversion of the RGB.

3.6. System mechanical design

The objective of the thesis is to create a portable device, so it was essential to consider this while doing the mechanical design.

The device case is made of PLA (Polylactic acid) using the 3D printing technology; the device's size is 80x55x55cm; it is tiny so that a man can hold it with one hand. It is also light with a weight of 120gm.

The case was designed to contain the electronic components and grantee a good way of fixing each component.

Figure 25: mechanical design of the device

The color sensor output is significantly affected by light and the distance between the sensor and the measured surface.

The case was designed to isolate the sensor from the external light; the sensor is positioned downward and centered in the middle; the distance between the sensor and the device edge is 25mm in this way, no light leakage will occur.

The reading hole, which allows the sensor to emit the light and receive the reflected wavelength, is designed to focus the light on the measured object.

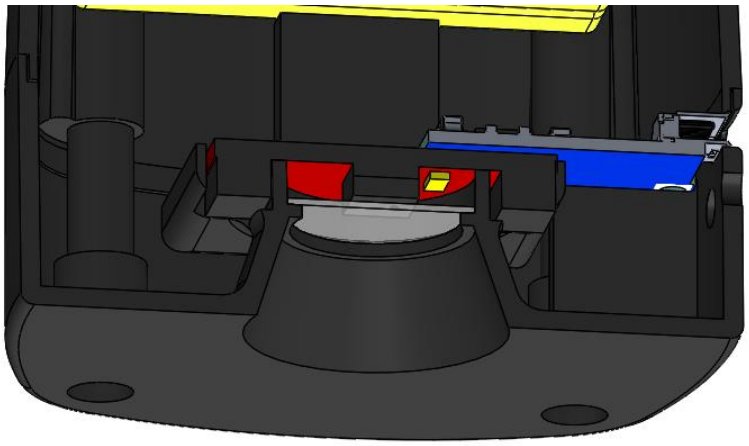


Figure 26: design of device reading hole

The UI (user interface) is positioned in a goon way to make it easy for the user to use it. The ON/OFF button with the indication LED is on the top of the device, and the charging USB port is on the front of the device with two LEDs to indicate the charging stat

Figure 27: design and positioning of user interface

4. Results

This section will show the measurements of ten colors with available RGB data and evaluate the measurement compared to the true one.

4.1. Measurement results

Color	Actual results (R, G, B)	Device Results (R, G, B)
PANTONE 19-4305	47, 52, 53	46, 52, 50
PANTONE 19-3935	62, 68, 101	61, 65, 97
PANTONE 19-3438	121, 71, 132	121, 65, 130
PANTONE 19-0825	246, 195, 36	243, 195, 36
PANTONE 19-1255	229, 118, 31	223, 120, 30
PANTONE 19-1564	220, 68, 58	228, 68, 60
PANTONE 19-1248	176, 88, 58	175, 93, 62
PANTONE 19-5722	63, 128, 112	63, 132, 112
PANTONE 19-4330	0, 130, 180	5, 130, 180
PANTONE 19-6437	123, 182, 101	121, 182, 103

Table 4: measurements results of 10 color samples

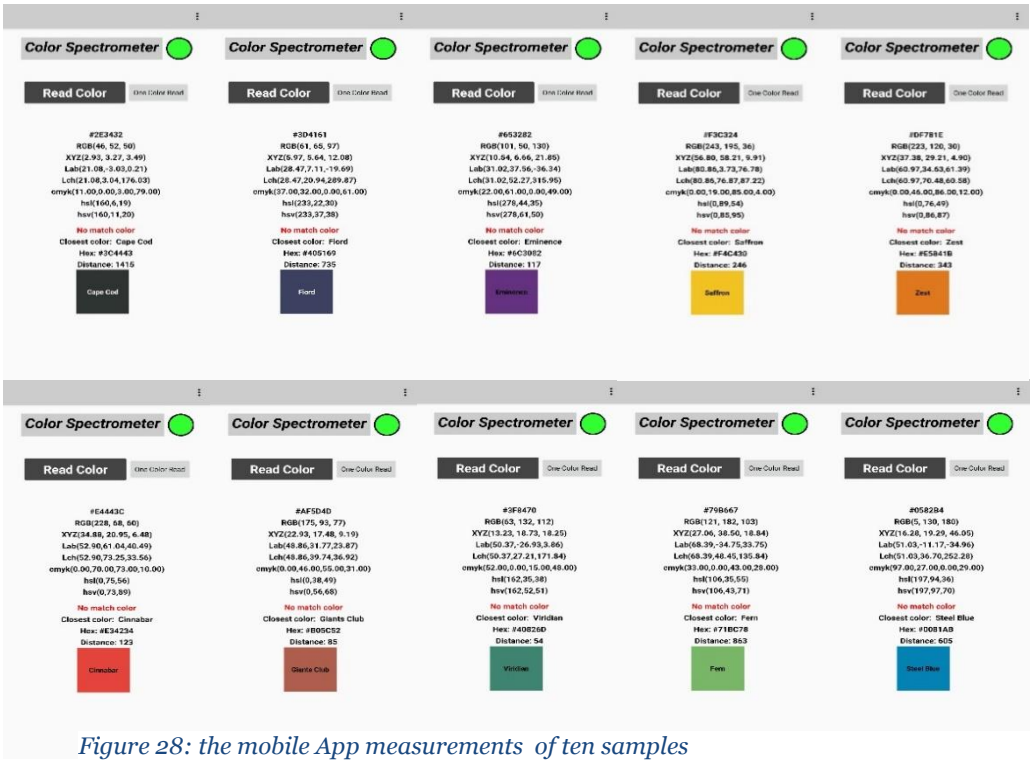


Figure 28: the mobile App measurements of ten samples

4.2. Analysis and discussion

In order to evaluate the output data, a suitable concept is to calculate the distance between the output color of the device and the Actual RGB of the color measured.

$$D = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$$

4.2.1. PANTONE 19-4305

Distance = 3.162

Actual Value	Device output
47, 52, 53	46, 52, 50

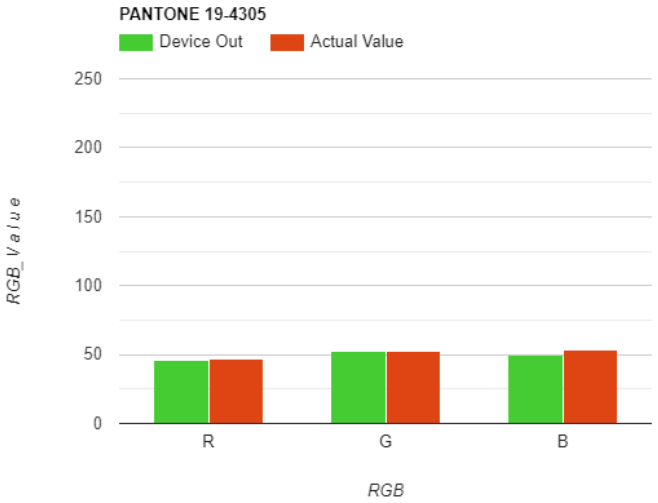


Figure 29: the first sample measurement

4.2.2. PANTONE 19-3935

Distance=5.099

Actual Value	Device output
62, 68, 101	61, 65, 97

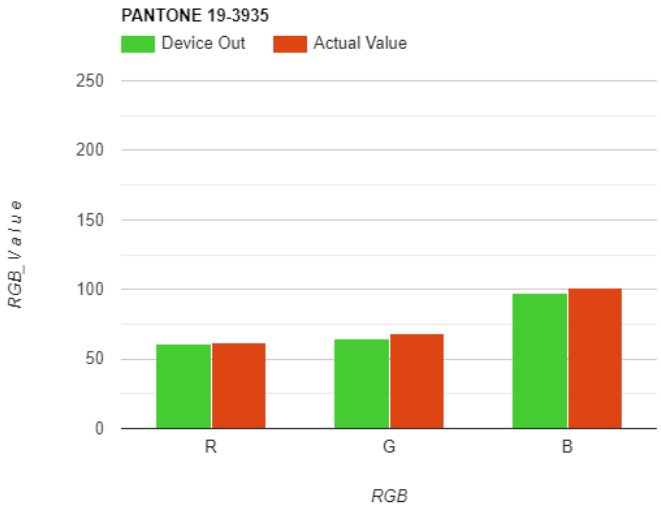


Figure 30: the second sample measurement

4.2.3. PANTONE 19-3438

Distance=6.324

Actual Value	Device output
121, 71, 132	121, 65, 130

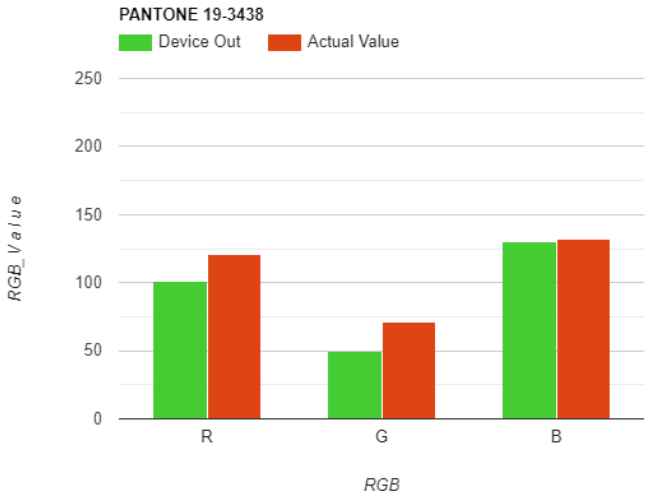


Figure 31: the third sample measurement

4.2.4. PANTONE 19-0825

Distance=3

Actual Value	Device output
246, 195, 36	243, 195, 36

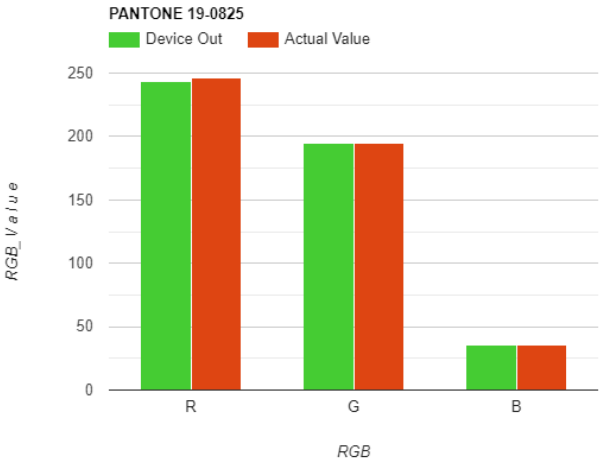


Figure 32: the fourth sample measurement

4.2.5. PANTONE 19-1255

Distance=6.403

Actual Value	Device output
229, 118, 31	223, 120, 30

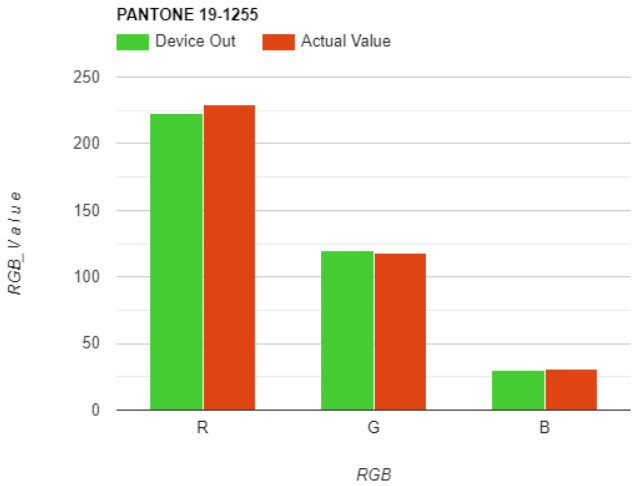


Figure 33: the fifth sample measurement

4.2.6. PANTONE 19-1564

Distance=8.246

Actual Value	Device output
220, 68, 58	228, 68, 60

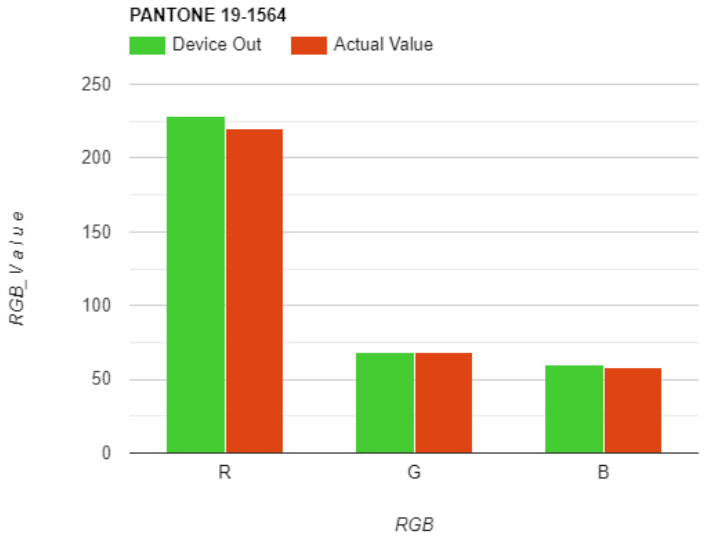


Figure 34: the sixth sample measurement

4.2.7. PANTONE 19-1248

Distance=6.481

Actual Value	Device output
176, 88, 58	175, 93, 62

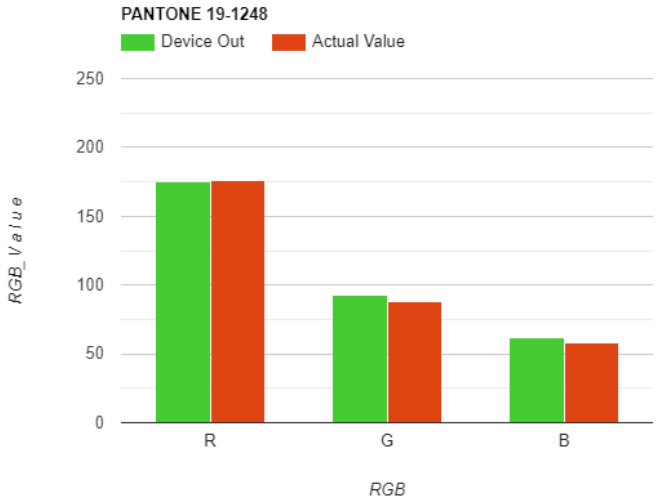


Figure 35: the seventh sample measurement

4.2.8. PANTONE 19-5722

Distance=4

Actual Value	Device output
63, 128, 112	63, 132, 112

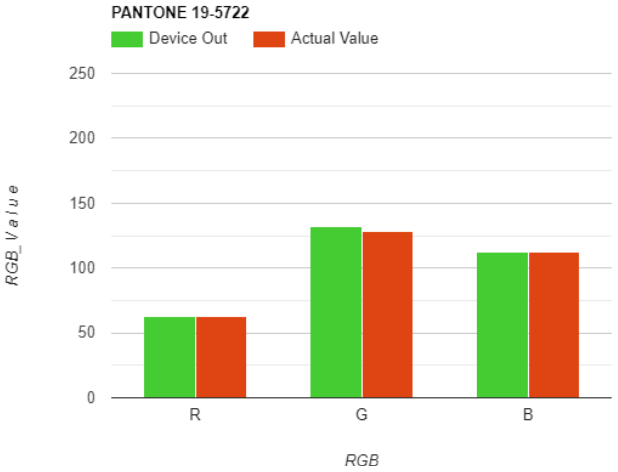


Figure 36: the eighth sample measurement

4.2.9. PANTONE 19-4330

Distance = 5

Actual Value	Device output
0, 130, 180	5, 130, 180

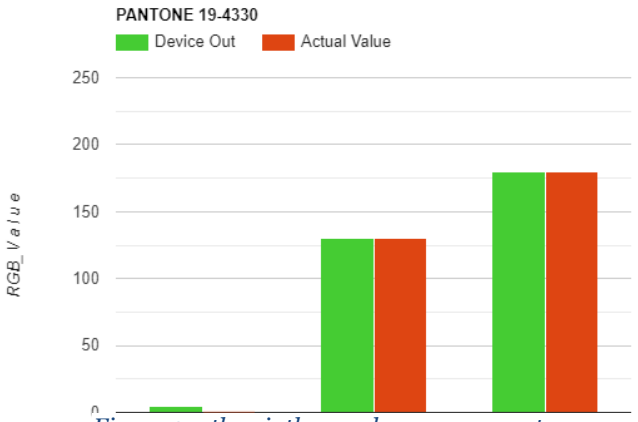


Figure 37: the ninth sample measurement

4.2.10. PANTONE 19-6437

Distance = 2.828

Actual Value	Device output
123, 182, 101	121, 182, 103

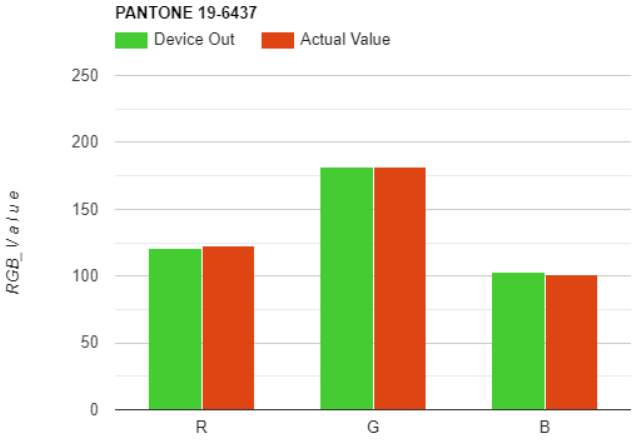


Figure 38: the tenth sample measurement

After taking the reading of 10 colors and calculating the distance with the actual values, the maximum distance is 8.246.

Color	**Disance**
PANTONE 19-4305	3.162
PANTONE 19-3935	5.099
PANTONE 19-3438	6.324
PANTONE 19-0825	3
PANTONE 19-1255	6.403
PANTONE 19-1564	8.246
PANTONE 19-1248	6.481
PANTONE 19-5722	4
PANTONE 19-4330	5
PANTONE 19-6437	2.828

Table 5: color Distance calculation for the ten samples

4.3. Troubleshooting

- One of the most challenging aspects of this project is the fact that it is not possible to evaluate the sensor output or the calibration process until finishing the mechanical design and the electric circuit wiring; because we need to fix the factors that affect the sensor output, which is the amount of light and the suppling voltage of the light sours (built-in LED).
- Secondly, to calibrate the color sensor, a colored material (paper card) with a defined color code is used, but the problem is that the color values measured by the device are not the same. It depends on the position of the sensor on the paper card; of course, it is not a significant difference, but it is crucial to have a precise value for the calibration process. The problem was solved by taking the values of three different positions on the paper and using the average result in the calibration process.

- The software update at the beginning of the implementation process was complex because to upload the updated code, there is a need to disassemble the device, connect the USB micro port to the ESP32, and thin assemble aging the device. This problem was fixed by using an external female USB micro port connected to the ESP inside the device, making it easy to update the software without trouble.

5. Conclusion and future work

5.1. Conclusion

To conclude, the output of this thesis project is a color measurement device that can read the color of any flat surface and display it to the user.
The device is tiny and lightweight; it is also mounted with a powerful rechargeable battery to be used in the industrial environment without the need to attach any cables.

The device is connected to an Android mobile application to show the result immediately; the user can take the read of one color or compare two colors showing the difference; moreover, the data can be shared using any sharing tool installed on the mobile phone.

The app shows the data in & color spaces the RGB, XYZ, LAB, LCH, CMYK, HSL, and HSV, which covers all the needs in the industrial sector.

5.2. Future work

- In this project, the sensor's output is the wavelengths of the red, green, and blue colors reflected from the measured surface; it will be more efficient to use a sensor with a color space output (like AS7261 output is XYZ) to avoid the problems of converting the data.
- Develop an IOS application to be compatible with IOS smartphones.
- Update the mobile Application to monitor the battery status.

6. References

[1] "https://www.sttmedia.com/colormodel-xyz."

[2] "https://www.mblock.com.tw/en/letterdetail/59/CIE%20XYZ%20color%20space?Open=59."

[3] "https://en.wikipedia.org/wiki/Color_space."

[4] "https://www.opticallimits.com/color-spaces."

[5] "https://color.viewsonic.com/explore/content/Color-gamut_6.html."

[6] "https://psychology.fandom.com/wiki/Color_space."

[7] "https://en.wikipedia.org/wiki/SRGB."

[8] "http://dougkerr.net/Pumpkin/articles/CIE_XYZ.pdf."

[9] "https://en.wikipedia.org/wiki/CMYK_color_model."

[10] "https://en.wikipedia.org/wiki/HSL_and_HSV."

[11] "https://www.xrite.com/learning-color-education/other-resources/what-is-a-spectrophotometer."

[12] "https://www.mouser.com/datasheet/2/891/esp-wroom-32_datasheet_en-1223836.pdf."

[13] "http://ai2.appinventor.mit.edu/."

[14] "esp32-wroom-wifi-devkit-v1_schematic".

[15] ams AG, "ams Datasheet AS7262 6-Channel Visible Spectral_ID Device with Electronic Shutter and Smart Interface Benefits Features General Description," 2017.

7. Appendices

Appendix A ESP 32 Code

```
#include "BluetoothSerial.h"
#include "esp_bt_main.h"
#include "esp_bt_device.h"
#include "AS726X.h"
AS726X sensor;
String  data;
String  dataON;
String  dataOFF;
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif
BluetoothSerial SerialBT;
void printDeviceAddress() {
const uint8_t* point = esp_bt_dev_get_address();
for (int i = 0; i < 6; i++) {
   char str[3];
   sprintf(str, "%02X", (int)point[i]);
   Serial.print(str);
   if (i < 5){
   Serial.print(":");}
}
Serial.println();
}
int addr ;
void setup()
{
Wire.begin();
data = "";
Serial.begin(9600);
delay(500);
pinMode(2, OUTPUT);
 sensor.begin();


  SerialBT.begin("ESP32_Agamy");
  Serial.println("This is your address");
  printDeviceAddress();
}
void loop()
{
sensor.setBulbCurrent(0);
sensor.setIntegrationTime(200);
sensor.setMeasurementMode(2);
sensor.setGain(0);
sensor.takeMeasurementsWithBulb();
 // T=sensor.getTemperatureF();
  float R = sensor.getCalibratedRed();
  float B = sensor.getCalibratedBlue();
  float G = sensor.getCalibratedGreen();
  float out1[] = {0,0,38,120,204,246};
  float in1[]  = {2.69,144.52,159.78,306.99,489.21,754.01};
  float val1 = R;
```

```
 float  RR = multiMap(val1, in1, out1, 6);
float out2[] = {0,49,107,130,132,169,195,224,251};
  float in2[]= {6.87,104.22,176.36,295.47,308.07,576.05,973.44,1037.57,1370.83};
  float val2 = G;
 float  GG = multiMap(val2, in2, out2, 9);
float out3[] = {0,69,139,162,180,173,180,220,255};
  float in3[]= {5.22,139.67,271.51,454.25,489.49,616.11,736.20,882.40,1156.51};
  float val3 =B;
 float  BB = multiMap(val3, in3, out3, 9);
//convert RGB to XYz
  double f1 = (RR / 255.000000);
  double f2 = (GG / 255.00000);
  double f3 = (BB / 255.00000);
  if ( f1 > 0.04045 ) {
    f1 = pow( ( ( f1 + 0.0550 ) / 1.0550 ) , 2.4000);
  }
  else              {
    f1 = f1 / 12.9200;
  }
  if ( f2 > 0.04045 ) {
    f2 = pow( ( ( f2 + 0.0550 ) / 1.0550 ) , 2.4000);
  }
  else             {
    f2 = f2 / 12.9200;
  }
  if ( f3 > 0.04045 ) {
    f3 = pow( ( ( f3 + 0.055 ) / 1.055 ) , 2.4000);
  }
  else{
    f3 = f3 / 12.9200;
      }

  f1 = f1 * 100;
  f2 = f2 * 100;
  f3 = f3 * 100;
  double x = ((f1 * 0.4124 ) + (f2 * 0.3576) + (f3 * 0.1805));
  double y = ((f1 * 0.2126 ) + ( f2 * 0.7152) + (f3 * 0.0722));
  double z = ((f1 * 0.0193 ) + (f2 * 0.1192 ) + (f3 * 0.9505));
  //double x = ((f1* 0.4124 )+ (f2 * 0.3576) +(f3 * 0.1805))*0.7562;
  //double y= ((f1 * 0.2126 )+( f2 * 0.7152) + (f3 * 0.0722))*0.7506;
  //double z= ((f1 * 0.0193 )+ (f2 * 0.1192 )+ (f3 * 0.9505))*0.7883;
  //convert XYZ to LAB
  double var_X = x / 95;
  double var_Y = y / 100;
  double var_Z = z / 108;
  if ( var_X > 0.008856 ) {
    var_X = pow( var_X , ( 0.3333333333 ));
  }
  else                  {
    var_X = ( 7.7870 * var_X ) + ( 16.0 / 116.0 );
  }
  if ( var_Y > 0.008856 ) {
    var_Y = pow (var_Y , ( 0.3333333333 ));
  }
  else                    {
    var_Y = ( 7.7870 * var_Y ) + ( 16.0 / 116.0 );
  }
  if ( var_Z > 0.008856 ) {
```

```
    var_Z = pow (var_Z , ( 0.3333333333 ));
  }
  else                    {
    var_Z = ( 7.7870 * var_Z ) + ( 16.0 / 116.0 );
  }
  double l = ( 116 * var_Y ) - 16;
  double a = 500 * ( var_X - var_Y );
  double b = 200 * ( var_Y - var_Z );
  //convert XYZ to LCH
  double h =  atan2  ( b, a);
  if ( h > 0 ) {
    h = ( h / PI ) * 180;
  }
  else             h = 360 - ( abs( h ) / PI ) * 180;
  double c = sqrt( pow (a, 2) + pow(b, 2) );
//convert RGB to CMYK
  float R_dot = (RR / 255.00);
  float G_dot = (GG / 255.00);
  float B_dot = (BB / 255.00);
  float maxRG= max(R_dot,G_dot);
float Kx=(1-max(maxRG,B_dot));
float Cx = (1-R_dot-Kx) / ( 1-Kx) ;
float Yx = ( 1-B_dot-Kx) / ( 1-Kx) ;
float Mx = (1-G_dot-Kx) /(  1-Kx);
int Cperc = Cx * 100;
int Mperc = Mx * 100;
int Yperc = Yx * 100;
int Kperc = Kx * 100;
 //convert RGB to HSL
int Hx;
float Sx;
float Lx;
int temp1;
float S;
float Cmax=max(max(R_dot,G_dot),B_dot);
float Cmin=min(min(R_dot,G_dot),B_dot);
float Delta=Cmax-Cmin;

if(Cmax==R_dot)
{
  temp1=(G_dot-B_dot)/Delta;
  Hx=(60*(temp1%6));
  }
if(Cmax==G_dot)
{Hx=(60*(((B_dot-R_dot)/Delta)+2));}
if(Cmax==B_dot)
{Hx=(60*(((R_dot-G_dot)/Delta)+4));}

Lx=(Cmax+Cmin)/2;

if(Delta==0)
{Hx=0;
Sx=0;
}
else
{Sx=Delta/(1-abs((2*Lx)-1));}
```

```
//convert RGB to HSV

if(Cmax==0)
{S=0;}
else
{S=Delta/Cmax;}
float Vx=Cmax;

int Sxperc=Sx*100;
int Lperc=Lx*100;
int Sperc=S*100;
int Vperc=Vx*100;


//sending the frame to the mobile APP

//Serial.println(String((int)RR) + ',' + String((int)GG) + ',' + String((int)BB) + ',' +
String((double)x) + ',' + String((double)y) + ',' + String((double)z) + ',' + String((double)l) +
',' + String((double)a) + ',' + String((double)b) + ',' + String((double)c) + ',' +
String((double)h) + ','+String((double)Cperc) + ',' + String((double)Mperc) + ',' +
String((double)Yperc) + ',' + String((double)Kperc) + ',' + String((int)Hx) + ',' +
String((int)Sxperc) + ',' + String((int)Lperc) + ',' + String((int)Sperc) + ',' +
String((int)Vperc) + ',');
SerialBT.println(String((int)RR) + ',' + String((int)GG) + ',' + String((int)BB) + ',' +
String((double)x) + ',' + String((double)y) + ',' + String((double)z) + ',' + String((double)l) +
',' + String((double)a) + ',' + String((double)b) + ',' + String((double)c) + ',' +
String((double)h) + ','+String((double)Cperc) + ',' + String((double)Mperc) + ',' +
String((double)Yperc) + ',' + String((double)Kperc) + ',' + String((int)Hx) + ',' +
String((int)Sxperc) + ',' + String((int)Lperc) + ',' + String((int)Sperc) + ',' +
String((int)Vperc) + ',');


      }

// the Lookup table multiMap function

float multiMap(float val, float* _in, float* _out, uint8_t size)
{
 uint8_t pos = 1;
 if(_in[0] < _in[1]) {
   if (val <= _in[0]) return _out[0];
   if (val >= _in[size-1]) return _out[size-1];
   if (val == _in[pos]) return _out[pos];
   while(val > _in[pos]) pos++;
 } else {
   if (val >= _in[0]) return _out[0];
   if (val <= _in[size-1]) return _out[size-1];
   if (val == _in[pos]) return _out[pos];
   while(val < _in[pos]) pos++;
 }
 // interpolate in the right segment for the rest
 return map(val, _in[pos-1], _in[pos], _out[pos-1], _out[pos]);
}
#include "BluetoothSerial.h"
#include "esp_bt_main.h"
#include "esp_bt_device.h"
#include "AS726X.h"
AS726X sensor;
```

```
String  data;
String  dataON;
String  dataOFF;
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;
void printDeviceAddress() {

const uint8_t* point = esp_bt_dev_get_address();

for (int i = 0; i < 6; i++) {
   char str[3];
   sprintf(str, "%02X", (int)point[i]);
   Serial.print(str);
   if (i < 5){
   Serial.print(":");}
}
Serial.println();
}
int addr ;
void setup()
{

Wire.begin();

data = "";


Serial.begin(9600);
delay(500);

pinMode(2, OUTPUT);
 sensor.begin();


  SerialBT.begin("ESP32_Agamy");
  Serial.println("This is your address");
  printDeviceAddress();

}


void loop()
{

sensor.setBulbCurrent(0);
sensor.setIntegrationTime(200);
sensor.setMeasurementMode(2);
sensor.setGain(0);
sensor.takeMeasurementsWithBulb();

 // T=sensor.getTemperatureF();
  float R = sensor.getCalibratedRed();
  float B = sensor.getCalibratedBlue();
  float G = sensor.getCalibratedGreen();
```

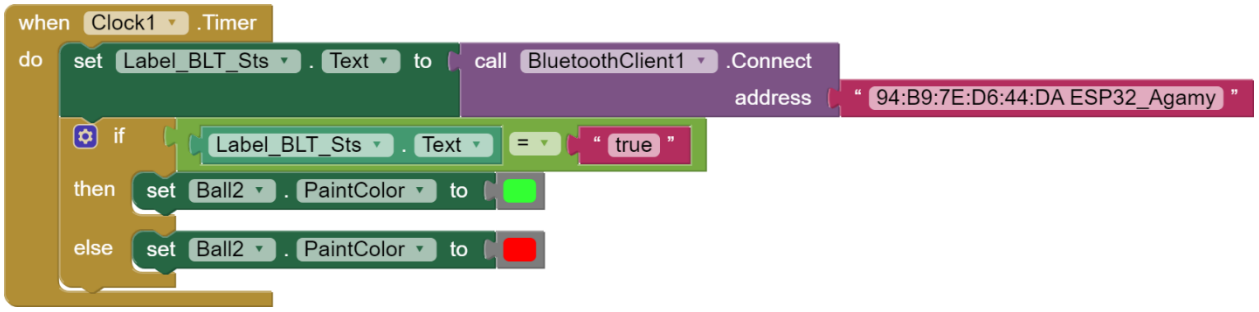
```
  float out1[] = {0,0,38,120,204,246};
  float in1[]  = {2.69,144.52,159.78,306.99,489.21,754.01};
  float val1 = R;
 float  RR = multiMap(val1, in1, out1, 6);

float out2[] = {0,49,107,130,132,169,195,224,251};
  float in2[]= {6.87,104.22,176.36,295.47,308.07,576.05,973.44,1037.57,1370.83};
  float val2 = G;
 float  GG = multiMap(val2, in2, out2, 9);

float out3[] = {0,69,139,162,180,173,180,220,255};
  float in3[]= {5.22,139.67,271.51,454.25,489.49,616.11,736.20,882.40,1156.51};
  float val3 =B;
 float  BB = multiMap(val3, in3, out3, 9);



//convert RGB to XYz

  double f1 = (RR / 255.000000);
  double f2 = (GG / 255.00000);
  double f3 = (BB / 255.00000);

  if ( f1 > 0.04045 ) {
    f1 = pow( ( ( f1 + 0.0550 ) / 1.0550 ) , 2.4000);
  }
  else              {
    f1 = f1 / 12.9200;
  }
  if ( f2 > 0.04045 ) {
    f2 = pow( ( ( f2 + 0.0550 ) / 1.0550 ) , 2.4000);
  }
  else             {
    f2 = f2 / 12.9200;
  }
  if ( f3 > 0.04045 ) {
    f3 = pow( ( ( f3 + 0.055 ) / 1.055 ) , 2.4000);
  }
  else{
    f3 = f3 / 12.9200;
      }

  f1 = f1 * 100;
  f2 = f2 * 100;
  f3 = f3 * 100;

  double x = ((f1 * 0.4124 ) + (f2 * 0.3576) + (f3 * 0.1805));
  double y = ((f1 * 0.2126 ) + ( f2 * 0.7152) + (f3 * 0.0722));
  double z = ((f1 * 0.0193 ) + (f2 * 0.1192 ) + (f3 * 0.9505));

  //double x = ((f1* 0.4124 )+ (f2 * 0.3576) +(f3 * 0.1805))*0.7562;
  //double y= ((f1 * 0.2126 )+( f2 * 0.7152) + (f3 * 0.0722))*0.7506;
  //double z= ((f1 * 0.0193 )+ (f2 * 0.1192 )+ (f3 * 0.9505))*0.7883;

  //convert XYZ to LAB
```

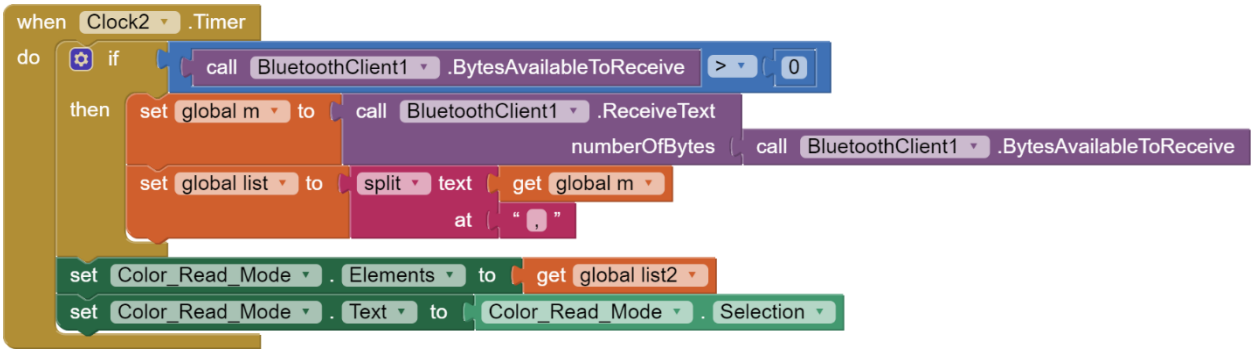
```
 double var_X = x / 95;
 double var_Y = y / 100;
 double var_Z = z / 108;

 if ( var_X > 0.008856 ) {
   var_X = pow( var_X , ( 0.3333333333 ));
 }
 else                  {
   var_X = ( 7.7870 * var_X ) + ( 16.0 / 116.0 );
 }
 if ( var_Y > 0.008856 ) {
   var_Y = pow (var_Y , ( 0.3333333333 ));
 }
 else                    {
   var_Y = ( 7.7870 * var_Y ) + ( 16.0 / 116.0 );
 }
 if ( var_Z > 0.008856 ) {
   var_Z = pow (var_Z , ( 0.3333333333 ));
 }
 else                    {
   var_Z = ( 7.7870 * var_Z ) + ( 16.0 / 116.0 );
 }

 double l = ( 116 * var_Y ) - 16;
 double a = 500 * ( var_X - var_Y );
 double b = 200 * ( var_Y - var_Z );

 //convert XYZ to LCH

 double h =  atan2  ( b, a);
 if ( h > 0 ) {
   h = ( h / PI ) * 180;
 }
 else             h = 360 - ( abs( h ) / PI ) * 180;
 double c = sqrt( pow (a, 2) + pow(b, 2) );



//convert RGB to CMYK

 float R_dot = (RR / 255.00);
 float G_dot = (GG / 255.00);
 float B_dot = (BB / 255.00);
 float maxRG= max(R_dot,G_dot);
float Kx=(1-max(maxRG,B_dot));


float Cx = (1-R_dot-Kx) / ( 1-Kx) ;
float Yx = ( 1-B_dot-Kx) / ( 1-Kx) ;
float Mx = (1-G_dot-Kx) /(  1-Kx);

int Cperc = Cx * 100;
int Mperc = Mx * 100;
int Yperc = Yx * 100;
int Kperc = Kx * 100;

//convert RGB to HSL
```

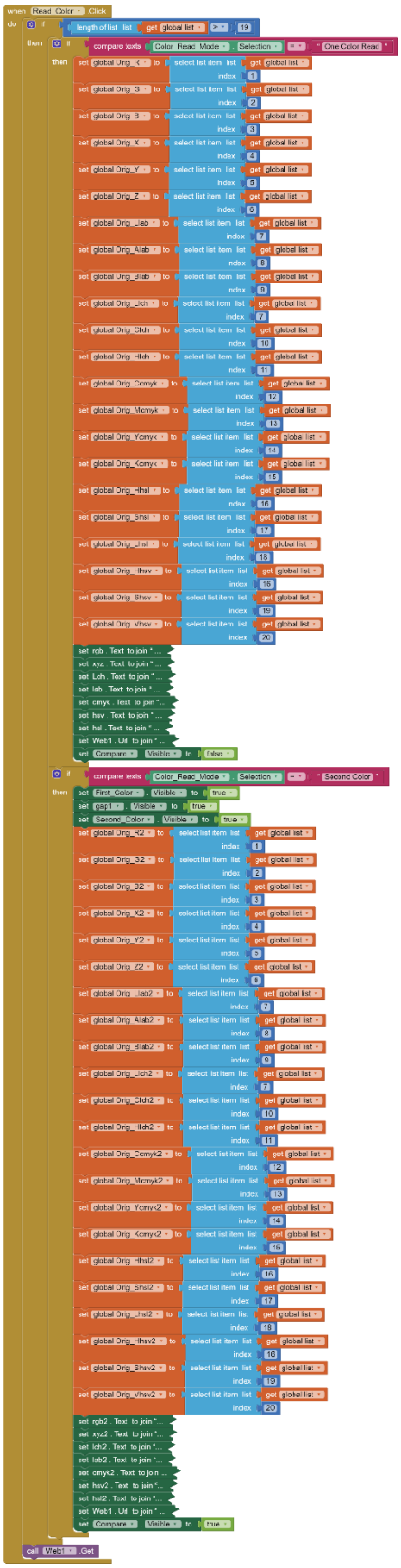
```
int Hx;
float Sx;
float Lx;
int temp1;
float S;
float Cmax=max(max(R_dot,G_dot),B_dot);
float Cmin=min(min(R_dot,G_dot),B_dot);
float Delta=Cmax-Cmin;

if(Cmax==R_dot)
{
  temp1=(G_dot-B_dot)/Delta;
  Hx=(60*(temp1%6));
  }
if(Cmax==G_dot)
{Hx=(60*(((B_dot-R_dot)/Delta)+2));}
if(Cmax==B_dot)
{Hx=(60*(((R_dot-G_dot)/Delta)+4));}

Lx=(Cmax+Cmin)/2;

if(Delta==0)
{Hx=0;
Sx=0;
}
else
{Sx=Delta/(1-abs((2*Lx)-1));}


//convert RGB to HSV

if(Cmax==0)
{S=0;}
else
{S=Delta/Cmax;}
float Vx=Cmax;

int Sxperc=Sx*100;
int Lperc=Lx*100;
int Sperc=S*100;
int Vperc=Vx*100;


//sending the frame to the mobile APP

//Serial.println(String((int)RR) + ',' + String((int)GG) + ',' + String((int)BB) + ',' +
String((double)x) + ',' + String((double)y) + ',' + String((double)z) + ',' + String((double)l) +
',' + String((double)a) + ',' + String((double)b) + ',' + String((double)c) + ',' +
String((double)h) + ','+String((double)Cperc) + ',' + String((double)Mperc) + ',' +
String((double)Yperc) + ',' + String((double)Kperc) + ',' + String((int)Hx) + ',' +
String((int)Sxperc) + ',' + String((int)Lperc) + ',' + String((int)Sperc) + ',' +
String((int)Vperc) + ',');
SerialBT.println(String((int)RR) + ',' + String((int)GG) + ',' + String((int)BB) + ',' +
String((double)x) + ',' + String((double)y) + ',' + String((double)z) + ',' + String((double)l) +
',' + String((double)a) + ',' + String((double)b) + ',' + String((double)c) + ',' +
String((double)h) + ','+String((double)Cperc) + ',' + String((double)Mperc) + ',' +
String((double)Yperc) + ',' + String((double)Kperc) + ',' + String((int)Hx) + ',' +
String((int)Sxperc) + ',' + String((int)Lperc) + ',' + String((int)Sperc) + ',' +
```

```
String((int)Vperc) + ',');


      }

// the Lookup table multiMap function

float multiMap(float val, float* _in, float* _out, uint8_t size)
{
 uint8_t pos = 1;
 if(_in[0] < _in[1]) {
   if (val <= _in[0]) return _out[0];
   if (val >= _in[size-1]) return _out[size-1];
   if (val == _in[pos]) return _out[pos];
   while(val > _in[pos]) pos++;
 } else {
   if (val >= _in[0]) return _out[0];
   if (val <= _in[size-1]) return _out[size-1];
   if (val == _in[pos]) return _out[pos];
   while(val < _in[pos]) pos++;
 }
 // interpolate in the right segment for the rest
 return map(val, _in[pos-1], _in[pos], _out[pos-1], _out[pos]);
}
```

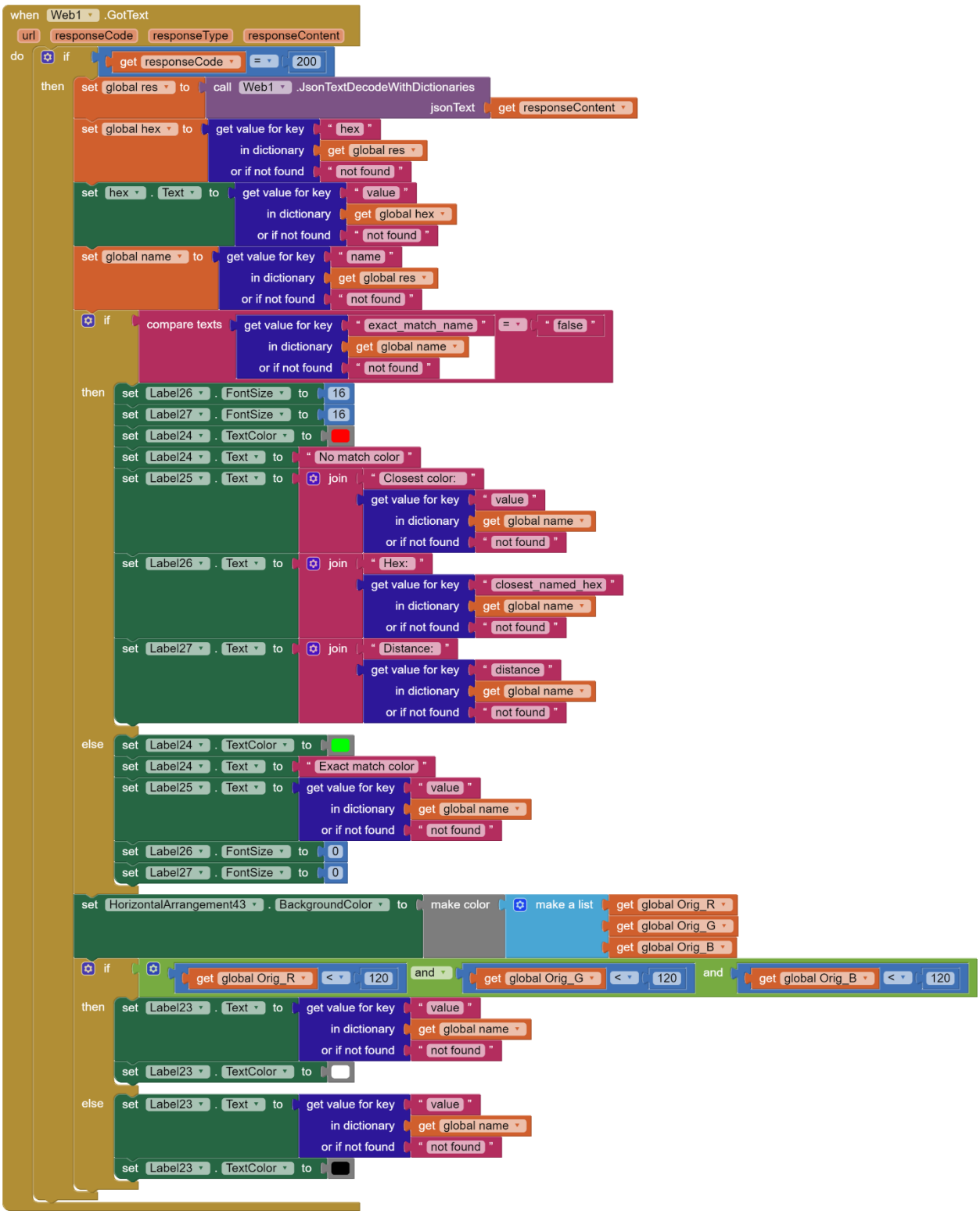
Appendix B Mobile APP Code






when Read_Color .Click
do if length of list get global list > 19
then if compare texts Color_Read_Mode . Selection = One Color Read
then set global Orig_R to select list item list get global list index 1
set global Orig_G to select list item list get global list index 2
set global Orig_B to select list item list get global list index 3
set global Orig_X to select list item list get global list index 4
set global Orig_Y to select list item list get global list index 5
set global Orig_Z to select list item list get global list index 6
set global Orig_Llab to select list item list get global list index 7
set global Orig_Alab to select list item list get global list index 8
set global Orig_Blab to select list item list get global list index 9
set global Orig_Lch to select list item list get global list index 7
set global Orig_Clch to select list item list get global list index 10
set global Orig_Hlch to select list item list get global list index 11
set global Orig_Cmyk to select list item list get global list index 12
set global Orig_Mcmyk to select list item list get global list index 13
set global Orig_Ycmyk to select list item list get global list index 14
set global Orig_Kcmyk to select list item list get global list index 15
set global Orig_Hhsl to select list item list get global list index 16
set global Orig_Shsl to select list item list get global list index 17
set global Orig_Lhsl to select list item list get global list index 18
set global Orig_Hhsv to select list item list get global list index 16
set global Orig_Shsv to select list item list get global list index 19
set global Orig_Vhsv to select list item list get global list index 20
set rgb . Text to join "...
set xyz . Text to join "...
set Lch . Text to join "...
set lab . Text to join "...
set cmyk . Text to join "...
set hsv . Text to join "...
set hsl . Text to join "...
set Web1 . Url to join "...
set Compare . Visible to false
if compare texts Color_Read_Mode . Selection = Second Color
then set First_Color . Visible to true
set cap1 . Visible to true
set Second_Color . Visible to true
set global Orig_R2 to select list item list get global list index 1
set global Orig_G2 to select list item list get global list index 2
set global Orig_B2 to select list item list get global list index 3
set global Orig_X2 to select list item list get global list index 4
set global Orig_Y2 to select list item list get global list index 5
set global Orig_Z2 to select list item list get global list index 6
set global Orig_Llab2 to select list item list get global list index 7
set global Orig_Alab2 to select list item list get global list index 8
set global Orig_Blab2 to select list item list get global list index 9
set global Orig_Lch2 to select list item list get global list index 7
set global Orig_Clch2 to select list item list get global list index 10
set global Orig_Hlch2 to select list item list get global list index 11
set global Orig_Ccmyk2 to select list item list get global list index 12
set global Orig_Mcmyk2 to select list item list get global list index 13
set global Orig_Ycmyk2 to select list item list get global list index 14
set global Orig_Kcmyk2 to select list item list get global list index 15
set global Orig_Hhsl2 to select list item list get global list index 16
set global Orig_Shsl2 to select list item list get global list index 17
set global Orig_Lhsl2 to select list item list get global list index 18
set global Orig_Hhsv2 to select list item list get global list index 16
set global Orig_Shsv2 to select list item list get global list index 19
set global Orig_Vhsv2 to select list item list get global list index 20
set rgb2 . Text to join "...
set xyz2 . Text to join "...
set lch2 . Text to join "...
set lab2 . Text to join "...
set cmyk2 . Text to join ...
set hsv2 . Text to join "...
set hsl2 . Text to join "...
set Web1 . Url to join "...
set Compare . Visible to true
call Web1 .Get


when Compare .Click
do
set gap2 . Visible to true
set Color_difference . Visible to true
set global Orig_R_delta to get global Orig_R - get global Orig_R2
set global Orig_G_delta to get global Orig_G - get global Orig_G2
set global Orig_B_delta to get global Orig_B - get global Orig_B2
set global Orig_X_delta to get global Orig_X - get global Orig_X2
set global Orig_Y_delta to get global Orig_Y - get global Orig_Y2
set global Orig_Z_delta to get global Orig_Z - get global Orig_Z2
set global Orig_Llab_delta to get global Orig_Llab - get global Orig_Llab2
set global Orig_Alab_delta to get global Orig_Alab - get global Orig_Alab2
set global Orig_Blab_delta to get global Orig_Blab - get global Orig_Blab2
set global Orig_Llch_delta to get global Orig_Llch - get global Orig_Llch2
set global Orig_Clch_delta to get global Orig_Clch - get global Orig_Clch2
set global Orig_Hlch_delta to get global Orig_Hlch - get global Orig_Hlch2
set global Orig_Ccmyk_delta to get global Orig_Ccmyk - get global Orig_Ccmyk2
set global Orig_Mcmyk_delta to get global Orig_Mcmyk - get global Orig_Mcmyk2
set global Orig_Ycmyk_delta to get global Orig_Ycmyk - get global Orig_Ycmyk2
set global Orig_Kcmyk_delta to get global Orig_Kcmyk - get global Orig_Kcmyk2
set global Orig_Hhsl_delta to get global Orig_Hhsl - get global Orig_Hhsl2
set global Orig_Shsl_delta to get global Orig_Shsl - get global Orig_Shsl2
set global Orig_Lhsl_delta to get global Orig_Lhsl - get global Orig_Lhsl2
set global Orig_Hhsv_delta to get global Orig_Hhsv - get global Orig_Hhsv2
set global Orig_Shsv_delta to get global Orig_Shsv - get global Orig_Shsv2
set global Orig_Vhsv_delta to get global Orig_Vhsv - get global Orig_Vhsv2
set rgb_delta . Text to j...
set xyz_delta . Text to j...
set lch__delta . Text to ...
set lab_delta . Text to j...
set cmyk_delta . Text to ...
set hsv_delta . Text to j...
set hsl_delta . Text to j...


when Web1 .GotText
url responseCode responseType responseContent
do if get responseCode = 200
then set global res to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
set global hex to get value for key "hex" in dictionary get global res or if not found "not found"
set hex . Text to get value for key "value" in dictionary get global hex or if not found "not found"
set global name to get value for key "name" in dictionary get global res or if not found "not found"
if compare texts get value for key "exact_match_name" in dictionary get global name or if not found "not found" = "false"
then set Label26 . FontSize to 16
set Label27 . FontSize to 16
set Label24 . TextColor to
set Label24 . Text to "No match color"
set Label25 . Text to join "Closest color:" get value for key "value" in dictionary get global name or if not found "not found"
set Label26 . Text to join "Hex:" get value for key "closest_named_hex" in dictionary get global name or if not found "not found"
set Label27 . Text to join "Distance:" get value for key "distance" in dictionary get global name or if not found "not found"
else set Label24 . TextColor to
set Label24 . Text to "Exact match color"
set Label25 . Text to get value for key "value" in dictionary get global name or if not found "not found"
set Label26 . FontSize to 0
set Label27 . FontSize to 0
set HorizontalArrangement43 . BackgroundColor to make color make a list get global Orig_R get global Orig_G get global Orig_B
if get global Orig_R < 120 and get global Orig_G < 120 and get global Orig_B < 120
then set Label23 . Text to get value for key "value" in dictionary get global name or if not found "not found"
set Label23 . TextColor to
else set Label23 . Text to get value for key "value" in dictionary get global name or if not found "not found"
set Label23 . TextColor to

Appendix C API JSON response code

Refernce:

https://www.thecolorapi.com/id?rgb=rgb(25,0,0)

Response:


```
{"hex":{"value":"#190000","clean":"190000"},

"rgb":{"fraction":{"r":0.09803921568627451,"g":0,"b":0},"r":25,"g":0,"b":0,"value":"rgb(25, 0,
0)"},

"hsl":{"fraction":{"h":0,"s":1,"l":0.049019607843137254},"h":0,"s":100,"l":5,"value":"hsl(0,
100%, 5%)"},

"hsv":{"fraction":{"h":0,"s":1,"v":0.09803921568627451},"value":"hsv(0, 100%,
10%)","h":0,"s":100,"v":10},

"name":{"value":"Diesel",":"#130000","exact_match_name":false,"distance":54},

"cmyk":{"fraction":{"c":0,"m":1,"y":1,"k":0.9019607843137255},"value":"cmyk(0, 100, 100,
90)","c":0,"m":100,"y":100,"k":90},

"XYZ":{"fraction":{"X":0.04043137254901961,"Y":0.020843137254901962,"Z":0.001892156862745098},
"value":"XYZ(4, 2, 0)","X":4,"Y":2,"Z":0},

"image":{"bare":"https://www.thecolorapi.com/id?format=svg&named=false&hex=190000",

"named":"https://www.thecolorapi.com/id?format=svg&hex=190000"},

"contrast":{"value":"#ffffff"},"_links":{"self":{"href":"/id?hex=190000"}},"_embedded":{}}
```