Laboratory of Optoelectronics and Photonics

Wrocław University of Science and Technology

Department of Electronic and Photonic Metrology

RGB component detector

(The project was carried out as part of the Optoelectronics 2-project course)

Gr. 23

Composition of the group

Radosław Mierzwa

Pavlo Kostushevych

Mateusz Gwioździk

WROCLAW 2024

Table of contents

1. Admission

This project of the RGB component detector was completed as part of the Optoelectronics 2 course during one semester. The report is a comprehensive documentation of the RGB component detector project, which is the subject of our research. In the content of the report, we explore the main aspects of the project, in accordance with the assumptions made at the planning stage.

This project aims to measure the RGB components of the environment and to use this data in practice. As part of the report, we present the design process, the technologies used, key components and the results achieved.

This report is a full dive into the design of an RGB component detector, designed to combine theory with practice.

Subject of the Report: This report focuses on a comprehensive discussion of the design, implementation and testing process of the RGB component detector. From theoretical foundations to a detailed description of the hardware and software part.

Contents of the report:

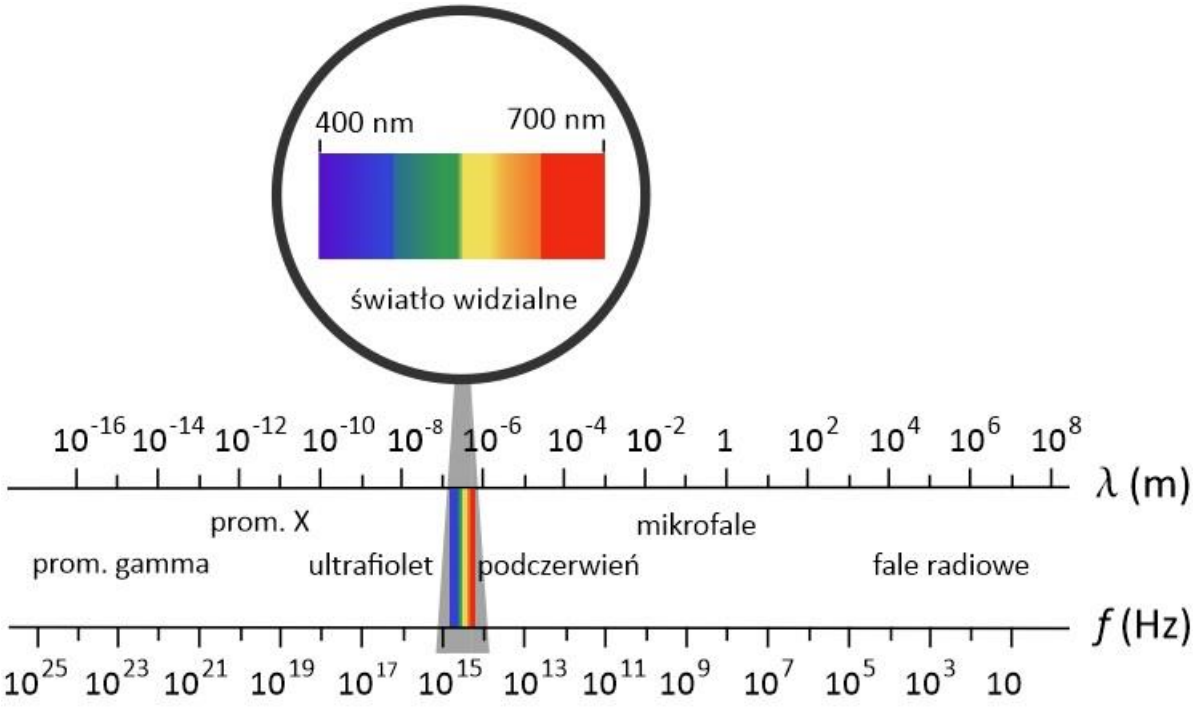
- **Theoretical Introduction:** Familiarize yourself with the fundamental concepts related to RGB color detection and a theoretical understanding of the technologies used.
- **Design Assumptions:** Understand the main goals and assumptions of the project that guided the process of its creation.
- **Description of the Hardware Part:** Review a detailed overview of the components of the detector, analyze the electronic architecture and the connections between the individual components.
- **Description of the software part:** Find out what software platforms have been used, what software algorithms support the functioning of the detector.
- **Device Commissioning and Testing:** Follow the commissioning process, trace the test steps, as well as analyze the results and adjust the device to your expectations.
- **Device Specifications:** Get a complete picture of the detector's technical performance, understand its performance characteristics and measurement precision.
- **Summary:** Evaluate the project's achievements in the context of the assumptions, think about practical applications of the detector, and develop similar projects in the future.

2. Theoretical introduction

Light is a type of electromagnetic radiation that is visible to the human eye. It is a form of energy that manifests itself in the form of electromagnetic waves of specific wavelengths. The electromagnetic spectrum includes different types of waves, such as radiowave, microwave, infrared, visible light, ultraviolet, X-ray, and gamma rays.

The visible light that is perceived by our eyes occupies only a small part of the entire electromagnetic spectrum. The visible light spectrum covers a wavelength range of about 380 to 750 nanometers (Figure 1). Within this wavelength range, different colors are visible to the human eye, with purple having a shorter wavelength and red having a longer wavelength.

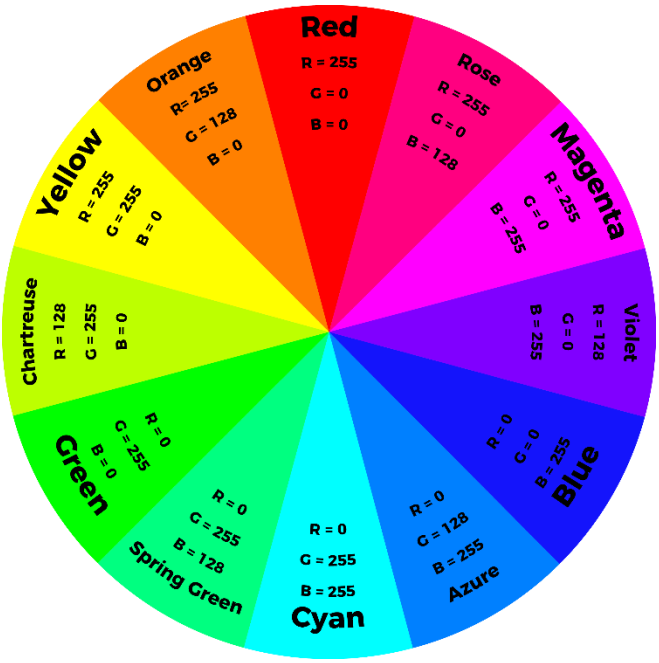
Light can have various sources such as the Sun, artificial sources, fluorescent lamps, fluorescent lamps, light-emitting diodes (LEDs), etc. There are many applications of light in everyday life and in various fields, such as lighting, photography, medicine (light therapy), telecommunications (fiber optics), physics (electromagnetic wave research), or display technology.



Drawing 1 Spectrum of electromagnetic waves [1].

Color spaces are an unambiguous description of a color presented in a coordinate system. Depending on the selected model, the system can be two- or three-dimensional.

RBG is an additive color space model in which color is described as the sum of the components of red, green, and blue (Figure 2). In the RGB model, each parameter can reach values in the range of 0-255, i.e. it is described on 8 bits. This way of writing allows you to describe over 16 million colors. RGB values are often stored in the hexadecimal system, e.g. white, which occurs when all 3 components reach the maximum value, in the hexadecimal system will be described as #FFFFFF.



Drawing 2 Model barw RGB [2].

3. Design assumptions

The project involved the creation of an RGB component detector, the results of which were to be presented on a liquid crystal display. The entire device was to be powered by batteries, which ensured portability and independence from the power source. In addition, it was planned that the RGB LED would emit light with a color similar to the analyzed color, which further facilitated visual monitoring of the measurement results. In this way, the design not only served as a colour detector, but also ensured readability and practicality of use thanks to the use of an LCD display and mobility thanks to battery power.

The project involved the expansion of the RGB component detector, the goal of which was to present the measurement results both on a serial monitor in the Arduino platform and on an additional LCD display. A key element of the concept was to power the entire device from a 9V battery, which was to ensure portability and independence from an external energy source.

In pursuit of functional and aesthetic excellence, we have introduced innovative elements, such as the RGB LED, which emits light with a color similar to the analyzed color. This advanced solution not only increases the readability of measurement results, but also facilitates visual monitoring of the analysis process.

3.1 Functional assumptions

* RGB Component Measurement: The main goal of the project is to be able to precisely measure the RGB color components in the tested environment.

*Displaying Measurement Results on Arduino Serial Monitor: Projecting measurement results onto a serial monitor on the Arduino platform, allowing the user to monitor and analyze data in real time.

*Secondary LCD Display: Integration of an additional LCD display for increased readability of measurement results and increased user interaction with the detector.

*Battery Powered for Portability: Use battery power (9V) to ensure portability and independence from a fixed power source.

*Additional RGB LED for Visualization of Results: Using the RGB LED to emit light with a color similar to the color being analyzed, which will facilitate visual monitoring of measurement results.

*Stability and Precision of Measurements: Providing stability and precision in the RGB component measurement process for effective color identification.

*Battery Life: Designed to be portable, ensure adequate battery life.

*Ambient Resistance: Designing the detector to withstand changing environmental conditions such as lighting.

*Functional Expandability: Designed the system to be able to add functional extensions in the future, allowing for possible upgrades and expansions of the RGB component detector.

*An important aspect is that the LCD display was powered directly from the Arduino platform, which in turn drew power from a 9V battery. This integrated approach to power eliminates the need for an additional power source for the LCD display, while increasing the mobility and practicality of the RGB component detector.

*Thanks to the use of a serial monitor in Arduino, an LCD display powered directly from the Arduino platform, an RGB diode and battery power, the project not only effectively meets the assumptions of a color detector, but also stands out for its professional design, excellent readability of measurement results, practicality of use and a high degree of mobility.

3.2 Design assumptions of the RGB component detector design:

*RGB LED with Color Resistors 220ohm Red and Green, Resistor 440 Ohm LED Blue:

The use of an RGB diode with resistors of 220 Ohm for red and green and a resistor of 440 Ohm for blue in order to precisely control the brightness and intensity of individual color components of blue introduces a very large range, so we used a larger resistor to limit the current.

*Arduino Platform as a Control Chip:

Use of the Arduino platform as a central control system, enabling the detector to be programmable and integrated with various modules.

*9V battery powered:

Use of a 9V battery as a power source, which ensures the portability of the detector and makes it independent of a constant power source.

*Color sensor module TCS3200:

The use of a color sensor module TCS3200 to accurately measure RGB components in the environment, which ensures precise and reliable measurement results.

*Suitable electrical connections:

Careful planning and execution of electrical connections between individual elements, ensuring stable operation of the detector.

*Adequate protection of electronics:

Protect the electronics from potential surges and interference to ensure the durability and reliability of the detector.

*Integration of the RGB LED Assembly with the LCD Display:

Seamless connection of the RGB LED to the LCD display to enable real-time visualization of measurement results.

*Application of Serial Monitor Interface in Arduino:

Implementation of the serial monitor interface in Arduino in order to present the measurement results in real time and facilitate analysis by the user.

*Battery Usage Optimization:

Optimization of the detector program and implementation of the sleep function to minimize power consumption, which translates into longer battery life.

Cooperation with the TCS3200 module:

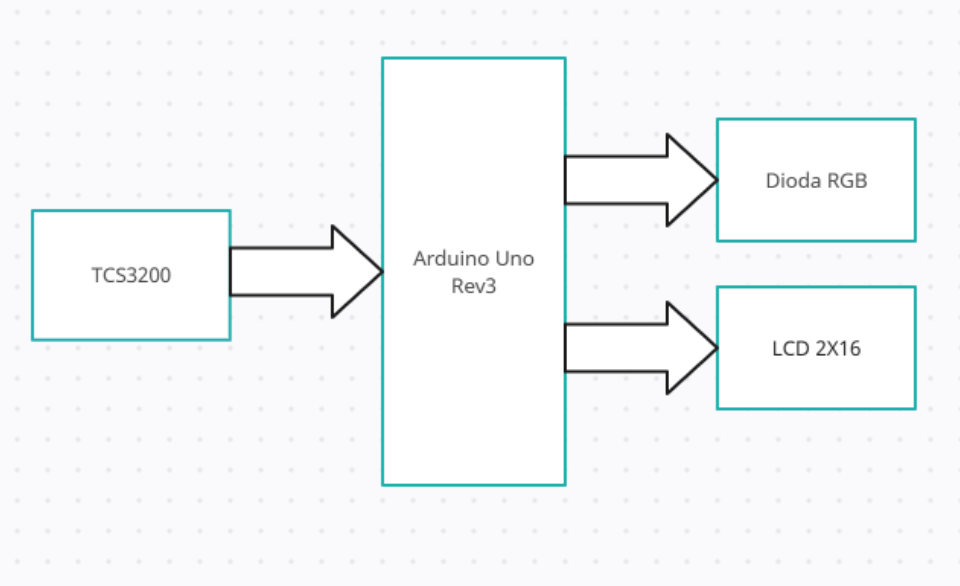
Coordinated collaboration between the Arduino platform and the TCS3200 module, enabling precise control and reading of color sensor data.

*Simple and Durable Mechanical Design:

Creating a simple and durable physical enclosure of the detector, which will be easy to use and protect all components from damage.

4. Description of the hardware part

The presented block diagram (Figure 3) shows the visual representation of individual components and their mutual relations in the design of the RGB component detector. This neat flowchart serves as a clear mapping of the structure of the hardware part, showing the harmonious integration of individual components, allowing for an easier understanding of the complexity of the device.



Drawing 3 Block diagram of the RGB component detector.

The device can be divided into 4 blocks, which are a sensor, a microcontroller, an LCD and an RGB LED.

a) Flowchart description:

The TCS3200 color sensor transmits data about the RGB intensity to the Arduino microcontroller, which precisely regulates the RGB diode through PWM signals, emitting light similar to the analyzed color. At the same time, the microcontroller transmits the finished measurement results to the 2x16 LCD display, enabling clear data presentation.

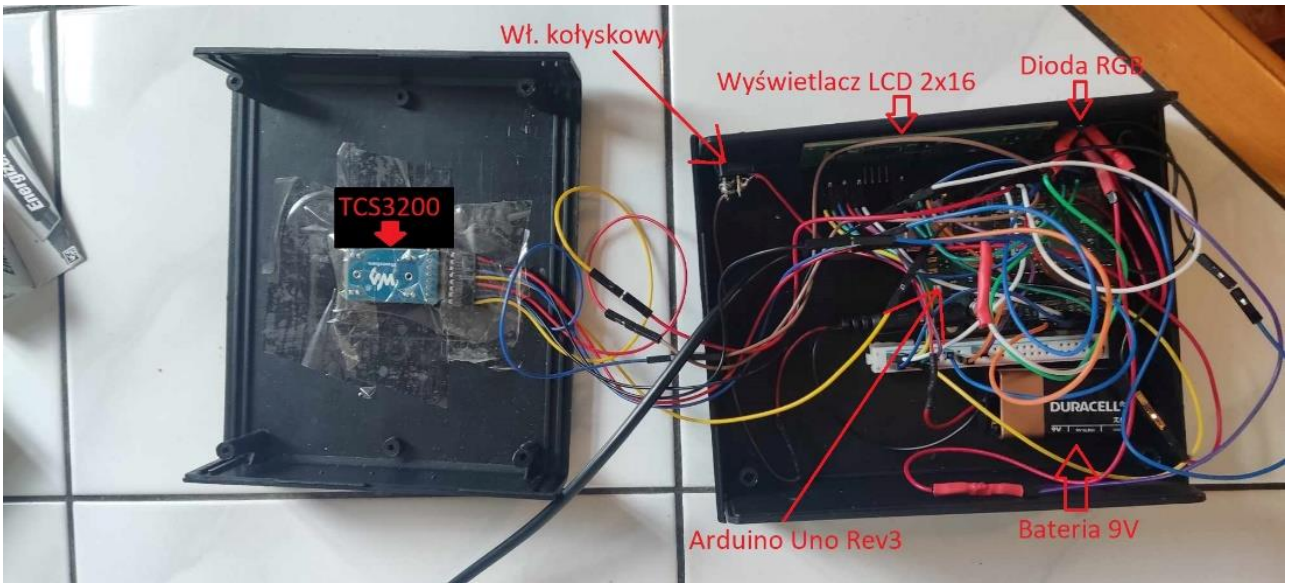
b) Photographs of the actual layout:

Figures 4-7 show the Front View of the Device (Figure 4), the Top View of the Device (Figure 5), the Bottom View of the Device (Figure 6), and the Inside View of the Device (Figure 7).

Drawing 4 Front view of the device

Drawing 5 Top view of the device

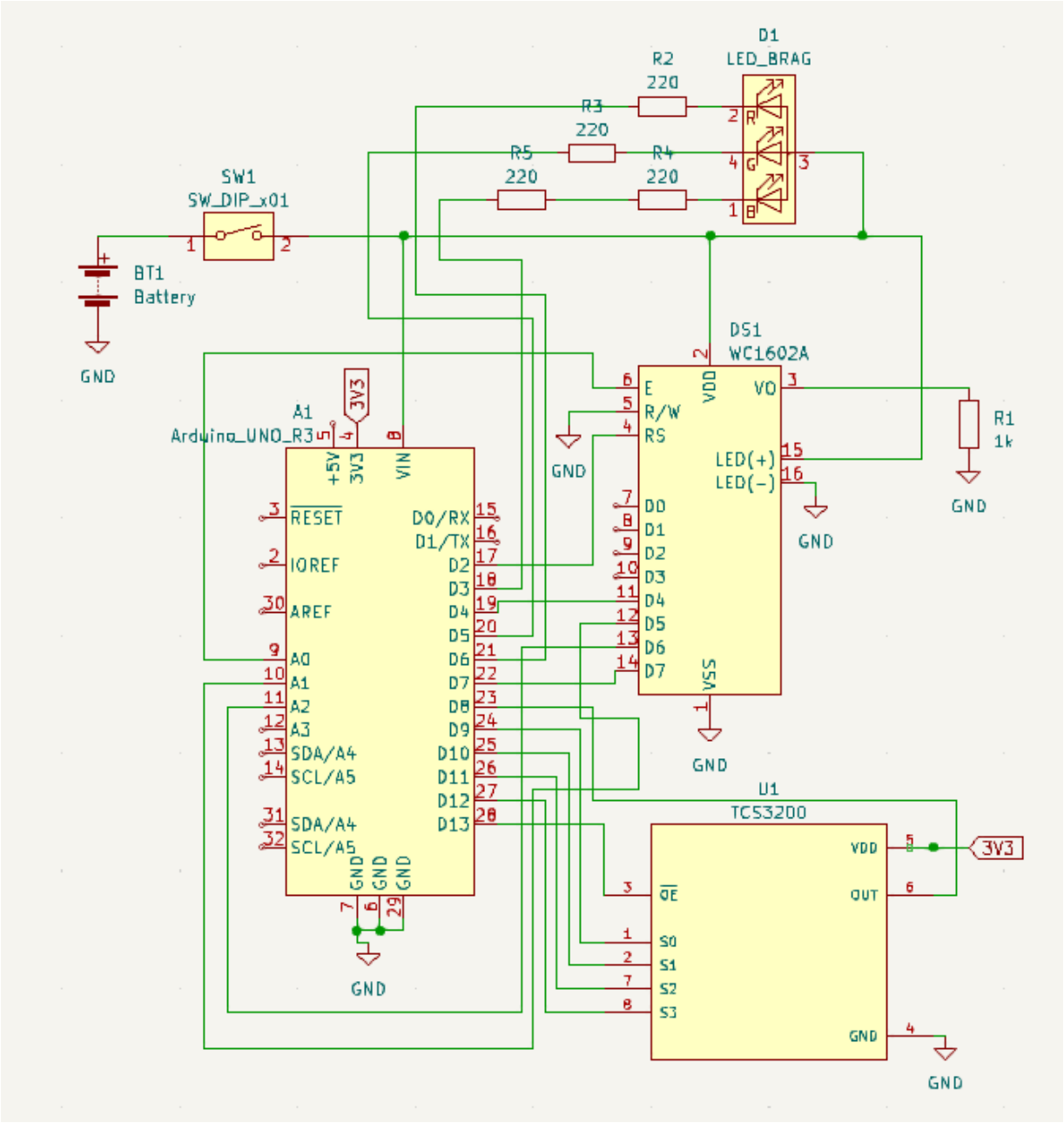
Drawing 6 View of the bottom of the device



Drawing 7 View of the inside of the device

4.1 Schematic diagram

The schematic diagram shown (Figure 8) illustrates the complex structure of the RGB component detector. The list of components includes, m.in, a TCS3200 color sensor module, an RGB LED and an LCD display. The diagram shows how the individual electronic components are connected to each other, showing the integral architecture of the device.



Drawing 8 Schematic diagram of the RGB component detector

a) Description of the schematic diagram:

Arduino UNO rev3 (A1):

It receives and processes data from the TCS3200 color sensor and controls the RGB LED.

It acts as a central control unit.

TCS3200 (U1):

It receives light from the environment, measuring the components of the colors: red, green and blue.

It uses pins S0-S3 to set the measurement mode and outputs an output signal on the OUT pin.

LCD (DS1):

It uses the LiquidCrystal library to communicate with Arduino.

It receives data from Arduino and presents the measurement results and the color name on a two-line LCD display.

Dioda RGB (D1):

It consists of three diodes (red, green, blue) emitting light of different colors.

It receives signals from the Arduino and regulates the intensity of light emission depending on the measurement results.

Electrical connections:

The individual pins of the Arduino (A1) and components (U1, DS1, D1) are connected according to the connection description, ensuring stable operation of the system.

Arduino Pins (A1) for Components:

A1_VIN: Arduino power supply from 5V.

A1_3V3: Power supply for the sensor TCS3200.

A1_A0, A1_A1, A1_A2: LCD control.

A1_D2, A1_D4, A1_D5, A1_D6, A1_D7: Connections to the LCD display.

A1_D8: Receive the signal from the TCS3200 sensor.

A1_D9, A1_D10, A1_D11, A1_D12: Control the measurement mode of the TCS3200 sensor.

A1_GND: Common mass of the system.

Piny LCD (DS1):

DS1_E: Display control.

DS1_D4, DS1_D5, DS1_D6, DS1_D7: Data connections to the display.

DS1_RS: Display mode control.

Piny TCS3200 (U1):

U1_OUT: Color sensor output signal.

U1_S0, U1_S1, U1_S2, U1_S3: Control the measurement mode.

U1_OE: Turning off the sensor output.

U1_VDD: Power supply to the sensor.

Piny Dioda RGB (D1):

RED, GREEN, BLUE: Control the emission intensity of the individual RGB LEDs.

b) List of elements

Table 1 A list of elements used in the project.

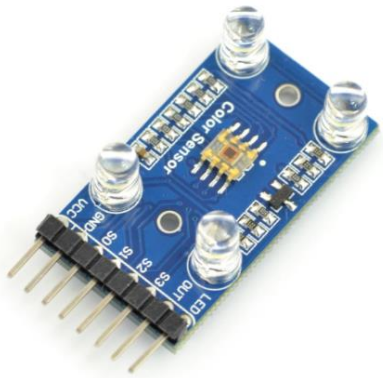
Symbol	Element Description	Pieces	value	unit
DS1	LCD 2x16	1	-	-
A1	Arduino uno rev.3	1	-	-
D1	Dioda RGB	1	-	-
BT1	6LR61 Alkaline Battery	1	9	V
R2,R3,R4,R5	Resistor	4	220	Oh
R1	Resistor	1	1	kΩ
U1	TCS3200	1	-	-
SW1	Rocker switch	1	-	-

4.2 Key Elements

This section details the key elements of the RGB component detector design, as well as their important role in ensuring the functionality and efficiency of the entire system. A professional approach to the description of individual components of the system allows you to understand their function and cooperation, which translates into precision and reliability of color measurements.

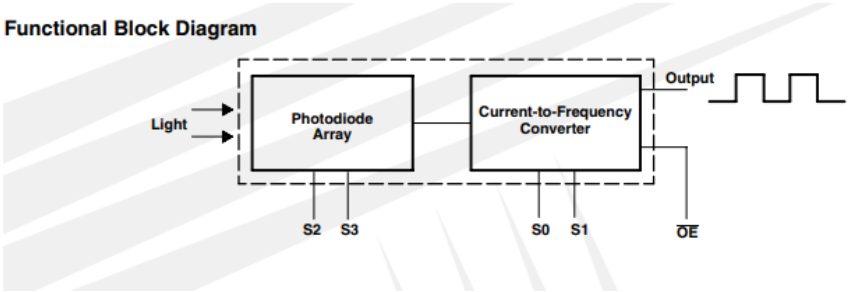
4.2.1 Color Sensor

The module with a color sensor allows you to measure RGB components, the output is the frequency depending on the light intensity of the selected color. Operated with any microcontroller or development kit, including Arduino. The RGB TCS3200 component sensor was used in the design (Figure 9).



Drawing 9 RGB component sensor TCS3200 [3].

The TCS3200 color sensor is a programmable light-to-frequency converter that combines configurable silicon photodiodes and a current-to-frequency converter in a single monolithic CMOS integrated circuit. The output is a square wave (with a duty cycle factor of 50%) and the frequency is proportional to the light intensity (light intensity).



Drawing 10 Sensor function block [4].

*TCS3200 Color Sensor Specification

- Supply voltage: 2.7 V to 5.5 V
- Pins: goldpin
- Simple communication with the microcontroller (frequency readout)
- Integrated 4 LEDs to illuminate the test item
- Dimensions: 36 x 20 mm (without connectors)

The module is powered by a voltage in the range of 2.7 V to 5.5 V, which makes it suitable for both 3.3 V and 5 V systems. The output is a signal whose frequency depends on the light intensity selected by means of pins S2 and S3, the color.

*Module connection

A detailed description of the pins is available in the tables later in the description. The pins are popular goldpin connectors that allow the sensor to be connected to the main module by means of connection cables (included). A description of the pins can be found in the table.

Module connection

PIN	OPIS
VCCIO	Zasilanie od 2,7 V do 5,5 V.
GND	Masa układu.
LED	Kontrola 4 diod LED.
OUT	Wyjście częstotliwości koloru.
S0/S1	Wejścia służące do skalowania częstotliwości wyjściowej. Opis poniżej.
S2/S3	Wejścia służące do wyboru typu fotodiody. Opis poniżej.

Drawing 11 Description of the sensor module pins TCS3200 [5].

*Color sensor configuration

With the help of configuration inputs S0 and S1, a prescaler (divider) of the output frequency signal is selected, thanks to which we can select the appropriate frequency range for the microcontroller used. Inputs S2 and S3 are used to select the type of measurement photodiode: red, green, blue or the clear option, i.e. without filtering (all RGB components will be measured at the same time).

Output Frequency Divider Selection

S0	S1	SKALA CZĘSTOTLIWOŚCI WYJŚCIOWEJ W STOSUNKU DO FO
L		Czujnik wyłączony
L	H	2 % 10-12 kHz
H	L	20 % 100-120 kHz
H	H	100 % 500-600 kHz

Drawing 12 Selecting the frequency scaling of the sensor output signal [6].

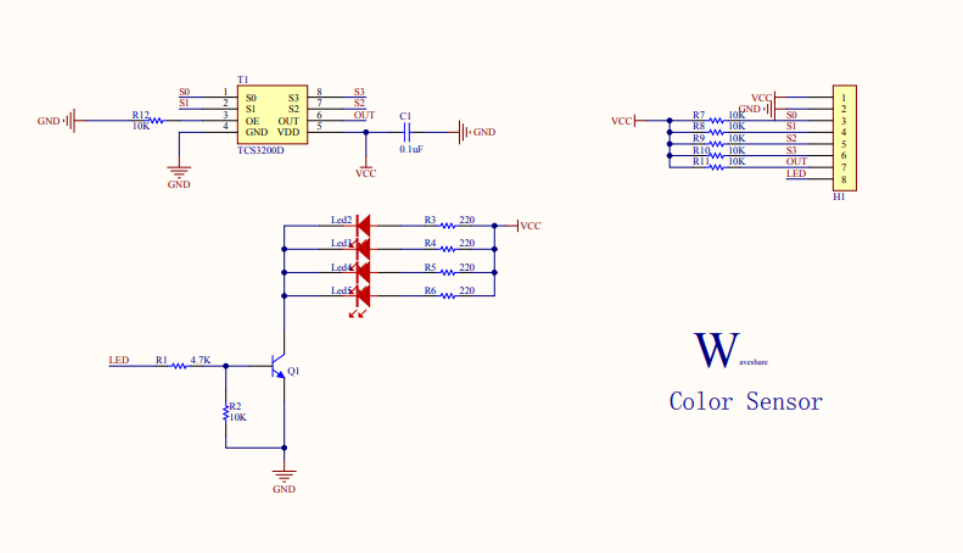
Selecting the photodiode (filtration) type

S0	S1	RODZAJ FOTODIODY
L	L	czerwona (R)
L	H	niebieska (B)
H	L	clear (RGB) (bez filtracji)
H	H	zielona (G)

L (Low) - stan niski

H (High) - stan niski

Drawing 13 Selection of the photodioy type [7].



Drawing 14 Schematic diagram of the sensor [8].

*The role of the sensor in the design.

The role of the TCS3200 color sensor in our RGB component detector project comes down to precise measurement of light intensity in three fundamental color components: red (R), green (G) and blue (B). The main tasks and role of the sensor include:

1.Color intensity measurement:

The TCS3200 sensor is responsible for the acquisition of data on the intensity of light in individual RGB components. This allows you to precisely determine the proportions and intensity of individual colors on the analyzed surface.

2.Color Filtration:

By using color filters, the sensor selectively captures wavelengths of light, allowing for concentrated measurement of RGB components. These filters are crucial for accurate color identification.

3.Analog-to-digital conversion (ADC):

The sensor uses a built-in analog-to-digital converter (ADC) to convert photodiode readings into digital signals. This makes it easier for the Arduino platform to further process the data.

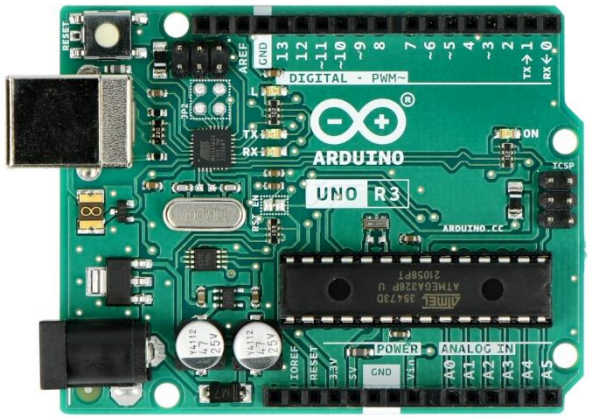
4.RGB LED Sync:

The TCS3200 sensor is synchronized with the RGB LED to ensure consistent measurements. This means that RGB measurements are performed simultaneously, resulting in precise measurement results.

In short, the role of the TCS3200 sensor in our project is to collect accurate data on the intensity of the RGB components, which enables effective color identification and meeting the functional objectives of the RGB component detector.

4.2.2 Microcontroller

The project uses the ATmega328P microcontroller built into the Arduino Uno rev3 evaluation board (Figure 15).



Drawing 15 Arduino UNO rev3 evaluation board with ATmega328P processor [9].

***Technical Features:**

-Clock frequency: Operates at 16 MHz as standard.

-Program Memory: It has 32 KB of Flash memory for user programs.

-RAM: It comes with 2 KB of RAM to store data while the program is running.

-EEPROM Memory: It has 1 KB of EEPROM memory to store data permanently.

-Number of Pins: 28 programmable I/O pins are available.

-Serial Communication: It has UART, SPI, and I2C interfaces, which allows it to communicate with other devices.

-Maximum current 50mA.

***What is:**

The microcontroller block is responsible for receiving the color intensity frequency of individual photodiodes from the sensor through a digital input or analogue-to-digital converter. The microcontroller used in the device is an ATmega328P built into the Arduino Uno rev3 evaluation board. It is an 8-bit microcontroller clocked at 16MHz. ATmega328P is a popular microcontroller manufactured by Microchip Technology, and previously by Atmel Corporation. It is widely used in electronic projects and prototypes, especially in Arduino platform applications. Here are some key highlights regarding the ATmega328P:

***How it works:**

A microcontroller works on the principle of programming, which means that its behavior is controlled by a previously written program that is uploaded to the microcontroller's memory. This program executes a series of instructions, reacting to input and performing appropriate actions on the outputs.

***Features of the Arduino Block in the RGB Detector Project:**

1. Data Collection: Arduino receives signals from the TCS3200 module, analyzes the color components (red, green, and blue), and converts them into numerical values.

2. RGB LED Operation Control: The microcontroller controls the RGB LED, adjusting the intensity of each color component based on the data read, which allows the measurement results to be visually displayed.

3. Communication with the LCD Display: Arduino manages the LCD display, presenting the current measurement results and the color name to the user in an easy-to-read form.

4. Power Optimization: The implementation of the sleep function and program optimization are designed to minimize power consumption, which translates into longer battery life of the detector.

5. Results-Based Decisions: Based on the read values of the color components, Arduino makes decisions about the color name, which allows for a more complex analysis of the color spectrum.

***I/O support:**

The Arduino block supports inputs from the TCS3200 color sensor and RGB LEDs, controlling the signals and passing data to the appropriate pins.

***LCD Communication:**

Arduino is responsible for transmitting data to the LCD display, which allows you to visualize the measurement results and interact with the user.

***Color Sensor Module Power Supply:**

The microcontroller manages the power supply of the TCS3200 module, initiates measurements and controls the duty cycle of the RGB component detector.

***Serial Monitor Interface:**

The Arduino block allows the use of a serial monitor interface, which allows you to monitor and debug the detector's operation through messages sent to the computer's serial port.

***Reading data:**

Arduino peaks data from various sources, such as the module TCS3200 This data is then processed according to specific program algorithms.

***Data processing:**

The microcontroller processes the collected data based on a previously written program. In the case of an RGB component detector, the processing includes the analysis of color intensity measurements and possible decision-making based on this data.

***Output Control:**

Based on the results of the analysis, Arduino controls the RGB diode and sends the information to the LCD display, allowing the results to be visualized for the user.

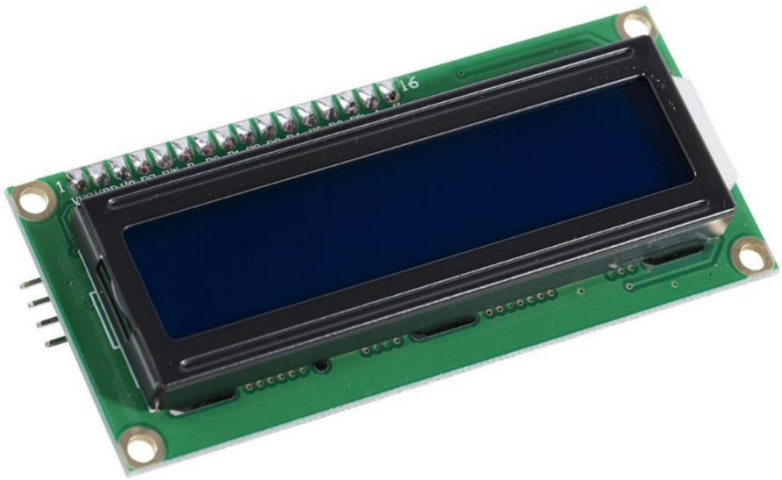
***Power Management:**

The microcontroller manages the power supply to the entire chip, including the RGB LED, TCS3200 module, and other connected components, to optimize power consumption.

The Arduino microcontroller block is the central control element in the RGB component detector design, determining the cooperation of individual components and the implementation of the project's functional assumptions.

4.2.3 2x16 LCD Display

The project uses an alphanumeric LCD display by justPi. Presents values on two lines, 16 characters each. It is powered by 5 V. It is characterized by extremely simple operation, high availability and numerous support for many microcontrollers. The signs are displayed on a blue background.



Drawing 16 LCD display [10].

***What is:**

The 2x16 Liquid Crystal Display (LCD) is a 2-line 16-character LCD display with a built-in controller for serial communication. In the RGB detector design, it acts as a user interface, presenting measurement results as well as diagnostic information.

***How it works:**

The display receives serial signals from the Arduino microcontroller and uses them to control liquid crystal pixels, creating characters and symbols on the screen.

***Features of LCD Display Block in RGB Detector Project:**

1. Presentation of Measurement Results: The LCD display is responsible for presenting the results of RGB component measurements in real time, allowing the user to evaluate the analyzed colors in real time.

2. Diagnostic Messages: In the event of errors or exceptions, the display informs the user of the system status by displaying appropriate diagnostic messages.

3. User Interaction: Allows you to interact with you, for example, by displaying calibration instructions, battery status, or other messages regarding the operation of the RGB component detector.

***Reading data:**

The display receives data from the microcontroller through a serial communication protocol, presenting the information in text form on the screen.

***Text Processing and Formatting:**

It takes care of formatting data and text to present measurement results in a readable and understandable form. It can also format diagnostic messages to make the information easier for the user to understand.

***Backlight Control:**

Equipped with backlight control, which allows it to ensure adequate contrast and readability in various lighting conditions (we used a 1 kΩ resistor).

***Integration with Arduino:**

It communicates with the Arduino microcontroller via serial wires, which allows data and instructions to be sent to the display.

***Blue backlight:**

In the case of the RGB component detector, the display is characterized by a blue backlight, which harmonizes with the overall design and aesthetics of the detector.

***Simple and Effective Information Presentation:**

It is an effective way of presenting data, especially where readability and immediate availability of information are crucial.

The 2x16 LCD serial display block therefore plays an important role in user interaction, presenting the measurement results and enabling effective diagnostics and operation of the RGB component detector.

4.2.4 RGB LED

The design uses an RGB diode with a common anode.

Drawing 17 RGB LED [11].

***What is:**

The 5mm RGB Common Anode LED is a tri-color diode that emits light in red, green and blue. It has a common anode, which means that the anode of the RGB diode is connected together, and the control of individual colors is done by changing the voltage level at the cathodes.

***How it works:**

It works on the principle of electroluminescence of semiconductors. Each color (red, green, blue) is generated by excitation of the corresponding semiconductor material, which causes light to be emitted.

***RGB LED Features in RGB Detector Design:**

Color Mixing: The RGB LED allows you to mix primary colors (red, green, blue) in different proportions, resulting in a wide range of colors.

Brightness Control: By adjusting the voltage applied per diode (PWM), you can control the brightness of the emitted light, which is important in the RGB component detector design.

***Color parameters:**

Red:

Emitted wavelength: 625 nm

Brightness: 100 mcd

Beam angle: 30°

Operating Parameters:

Current If: 25 mA

Voltage Vf: 2.0V

Green:

Emitted wavelength: 525 nm

Brightness: 800 mcd

Beam angle: 30°

Operating Parameters:

Current If: 25 mA

Voltage Vf: 3.5V

Blue:

Emitted wavelength: 470 nm

Brightness: 20 mcd

Beam angle: 30°

Operating Parameters:

Current If: 25 mA

Voltage Vf: 3.5V

4.2.5 Battery

The project uses an alkaline energizer max plus 9V 6LR61 (650 mAh) battery

Drawing 18 The battery used in the device [12].

***What is:**

The 9V battery is the power source for the whole RGB component detector project. It provides the electrical energy necessary for the operation of the Arduino microcontroller, RGB LEDs, LCD display and other components.

*** Battery Type:**

Battery type: 9V (We assume that this is the battery powering the entire system).

*** How it works:**

The 9V battery supplies the constant electrical voltage needed to power all components in the project. In the case of the RGB detector, the battery is a portable source, which allows it to be independent of the power source.

*** Why you need it:**

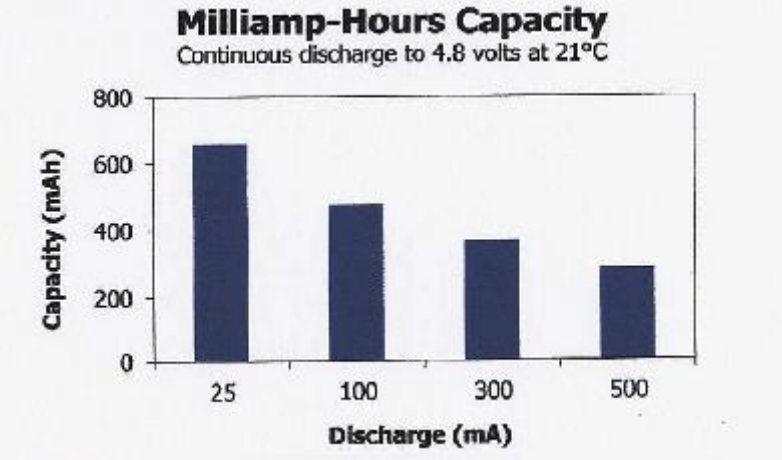
A 9V battery is necessary to enable portable and power-source-independent operation of the RGB component detector. It provides electricity for the Arduino microcontroller, RGB LEDs, LCD display and other connected components.

*** Battery Parameters:**

Voltage: 9V (Average battery voltage 9V during use).

Battery Type: Alkaline or other suitable type.

Capacity: 650 mAh at 100 mA capacity about 500 mAh (Figure 19)



Drawing 19 Dependence of battery capacity on discharge current [13].

Type: Alkaline

Safe maximum current draw: 100-150 mAh.

4.2.6 Rocker switch

A rocker switch was used in the project.

Drawing 20 Rocker switch used in the project [14].

***What is:**

The 250V 6A rocker switch is a component that acts as a switch in the electrical circuit of the RGB component detector project. It allows the device to be switched on and off in a controlled manner.

***Technical Specifications:**

Operating voltage: 250V

Working Current: 6A

Switch type: Rocker (mechanical, with an on/off switch in the form of a lever).

***How it works:**

A rocker switch is a mechanical device whose lever allows an electrical circuit to be physically connected or disconnected. The device is switched on and off by moving a lever.

***Why you need it:**

The rocker switch is necessary to enable controlled power on and off of the RGB component detector. It provides easy access to control of the device and allows you to save energy when the device is not in use.

***Application in RGB Detector Project:**

The rocker switch is a key part of the RGB component detector design, allowing easy power management of the device. Its placement on the detector housing makes it easy to access and operate.

5. Description of the program part

5.1 Development environment

To write the code, compile it and implement the program on the microcontroller, the C programming language was used in the Arduino IDE environment, created by the microcontroller manufacturer. Arduino IDE is an advanced tool that offers the user a wide range of possibilities, including access to

ready-made sketches, libraries and an extensive forum where you can get help if you need it. The code listing has been placed in the add-ons.

5.2 Algorithm of the program

The following diagram (Figure 21) shows the algorithm on which the design code is based.

Drawing 21 Diagram showing the program's algorithm

5.3 Code

In this chapter, the following blocks of the program will be described in detail:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(2, A0, 4, A1, A2, 7);
#define S0 9
#define S1 10
#define S2 11
#define S3 12
#define OUT 8
#define LED 13

#define RED 6
#define GREEN 5
#define BLUE 3
```

Drawing 22 A piece of code responsible for defining pins.

In the presented code fragment (Figure 22), the pins of the Arduino UNO rev3 microcontroller are defined, which are used in the project. Digital pins 2, 4 and 7 as well as analog pins A0, A1 and A2 are used to operate the LCD. The TCS3200 color sensor is controlled by digital pins 8 to 13, while PWM digital pins 3, 5, and 6 are used to control the RGB LED.

```
long int HALF_PERIOD = 0;
int RED_VALUE = 0;
int GREEN_VALUE = 0;
int BLUE_VALUE = 0;
int RED_LED_VALUE = 0;
int GREEN_LED_VALUE = 0;
int BLUE_LED_VALUE = 0;
String COLOR_NAME;
```

Drawing 23 A piece of code responsible for declaring variables.

Then (Figure 23) the variables used in the project are declared. To store the data read from the sensor, a variable HALF_PERIOD has been created, to which the value read from the sensor using **the pulseIn()** function is written in the further part of the code. The int **RED_VALUE**, **GREEN_VALUE** and **BLUE_VALUE variables** will store the RGB components of the color being tested, while the RED_LED_VALUE, **GREEN_LED_VALUE** and **BLUE_LED_VALUE** variables will contain values that will later be supplied to the RGB LED. String **COLOR_NAME** will store the color name based on the content of the RGB members.

```
23  void setup() {
24
25    pinMode(S0, OUTPUT);
26    pinMode(S1, OUTPUT);
27    pinMode(S2, OUTPUT);
28    pinMode(S3, OUTPUT);
29    pinMode(LED, OUTPUT);
30    pinMode(OUT, INPUT);
31    lcd.begin(16, 2);
32
33    digitalWrite(S0, HIGH);
34    digitalWrite(S1, LOW);
35
36    digitalWrite(LED,HIGH );
37
38    Serial.begin(9600);
39  }
```

Drawing 24 A piece of code when initializing the program.

The **setup()** function (Figure 24) assigns the pins connected to the sensor a mode (**OUTPUT** or **INPUT)** and calls a function to start the display. S0 pin is assigned HIGH and S1 is **assigned LOW** to set the sensor return frequency scaling to 20%. Assigning a high state to the LED pin causes the LEDs attached to the sensor module to light up. We also start sending information to a serial monitor at a bit rate of 9600 bps.

```
43    digitalWrite(S2, LOW);
44    digitalWrite(S3, LOW);
45    HALF_PERIOD = pulseIn(OUT, LOW);
46    RED_VALUE = 255-HALF_PERIOD;
47    if (RED_VALUE < 0) {
48      RED_VALUE = 0;
49    }
50
51    Serial.print("R = ");
52    Serial.print(RED_VALUE);
53    Serial.print("   ");
54
55    delay(100);
```

Drawing 25 A piece of code responsible for reading the red component.

According to the sensor documentation, TCS3200 setting the LOW value on pins S2 and S3 sets the sensor to read the red value. The sensor returns a square wave signal (duty cycle = 50%) with a frequency depending on the intensity of the component under test. With **the pulseIn() function** , we read the duration of the low signal, i.e. 1/2 of the period of the signal under study. (Figure 25)

- **digitalWrite(S2, LOW); oraz digitalWrite(S3, LOW);**
    - They set a specific mode of operation of the sensor. In this case, both settings to LOW mean that the sensor is in red reading mode.
- **HALF_PERIOD = pulseIn(OUT, LOW);**

- Measures the duration of pulses with a low logic state on the OUT pin. This value is then used to determine the intensity of the received light, which in turn corresponds to the red part of the spectrum.

- **RED_VALUE = 255 - HALF_PERIOD;**
  - Reverses the read value because the TCS3000 sensor returns a larger value for darker colors. This value is then assigned to the **variable RED_VALUE**.
- **if (RED_VALUE < 0) { RED_VALUE = 0; }**
  - Prevents the read value from being below zero. If the value was negative, it is set to zero.
- **Serial.print("R = "); Serial.print(RED_VALUE); Serial.print(" ");**
  - Prints the read red color value to the Serial Monitor in the Arduino environment.
- **delay(100);**
  - A short pause to avoid reading the next colors too quickly.

The same steps for the green (**GREEN_VALUE**) (Figure 26) and blue (**BLUE_VALUE**) (Figure 27) colors are repeated, with corresponding changes to the pin settings (S2 and S3) of the sensor.

```
59  digitalWrite(S2, HIGH);
60  digitalWrite(S3, HIGH);
61
62  HALF_PERIOD = pulseIn(OUT, LOW);
63  GREEN_VALUE = 255-HALF_PERIOD;
64  if (GREEN_VALUE < 0) {
65    GREEN_VALUE = 0;
66  }
67
68  Serial.print("G = ");
69  Serial.print(GREEN_VALUE);
70  Serial.print("   ");
71  delay(100);
```

Drawing 26 A piece of code responsible for reading the green component.

```
74  digitalWrite(S2, LOW);
75  digitalWrite(S3, HIGH);
76
77  HALF_PERIOD = pulseIn(OUT, LOW);
78  BLUE_VALUE = 255-HALF_PERIOD;
79  if (BLUE_VALUE < 0) {
80    BLUE_VALUE = 0;
81  }
82
83  Serial.print("B = ");
84  Serial.print(BLUE_VALUE);
85  Serial.println("   ");
86  delay(100);
```

Drawing 27 A piece of code responsible for reading the blue component.

```
88  RED_LED_VALUE=RED_VALUE;
89  GREEN_LED_VALUE=GREEN_VALUE;
90  BLUE_LED_VALUE=BLUE_VALUE;
```

Drawing 28 A piece of code that is responsible for the variables used to control the RGB LEDs.

In this code snippet (Figure 28), the values read from the color sensor **(RED_VALUE, GREEN_VALUE, BLUE_VALUE**) are assigned to the variables that will be used to control the RGB LED (RED_LED_VALUE, GREEN_LED_VALUE, BLUE_LED_VALUE). These variables are responsible for setting the light intensity of the RGB LED, which as a result affects the color emitted by the LED.

The next code snippet (Figures 29-40) contains a series of if-else conditions that check the values read from the color sensor (red, green, and blue) and assign them the appropriate color names. In addition, in some cases, some modifications are made to the values for RGB LEDs to achieve the desired color effect. Here is a description of each condition:

```
93    if(RED_VALUE < 30 && GREEN_VALUE < 30 && BLUE_VALUE < 30)
94    {
95      COLOR_NAME = "Black";
96    }
```

Drawing 29 The code snippet responsible for the condition for the color black.

```
97    else if(RED_VALUE > 165 && GREEN_VALUE > 190 && BLUE_VALUE > 190)
98    {
99      COLOR_NAME = "White";
100   }
```

Drawing 30 The code snippet responsible for the condition for the color white.

```
101   else if(RED_VALUE >= 0 && RED_VALUE < 20 && GREEN_VALUE > 130 && GREEN_VALUE < 180 && BLUE_VALUE > 180 && BLUE_VALUE < 255)
102   {
103     COLOR_NAME = "Blue";
104   }
```

Drawing 31 The code snippet responsible for the condition for the color blue.

```
105     else if(RED_VALUE > 45 && RED_VALUE < 160 && GREEN_VALUE > 170 && GREEN_VALUE < 210 && BLUE_VALUE > 195 && BLUE_VALUE < 225)
106   {
107     COLOR_NAME = "Light Blue";
108   }
```

Drawing 32 The code snippet for the condition for light blue.

```
109     else if(RED_VALUE > 140 && RED_VALUE < 225 && GREEN_VALUE >= 0 && GREEN_VALUE < 20 && BLUE_VALUE >= 0 && BLUE_VALUE < 28)
110   {
111     COLOR_NAME = "Red";
112     BLUE_LED_VALUE=0;
113   }
```

Drawing 33 A piece of code responsible for the condition for the red color, as well as the modification of the value for the RGB LED.

```
114   else if(RED_VALUE > 160 && RED_VALUE < 205 && GREEN_VALUE > 60 && GREEN_VALUE < 115 && BLUE_VALUE > 25 && BLUE_VALUE < 90)
115   {
116     COLOR_NAME = "Orange";
117     RED_LED_VALUE=160;
118     GREEN_LED_VALUE=37;
119     BLUE_LED_VALUE=0;
120   }
```

Drawing 34 A piece of code responsible for the condition for the orange color, as well as the modification of the value for the RGB LED.

```
121   else if(RED_VALUE > 40 && RED_VALUE < 90 && GREEN_VALUE >= 0 && GREEN_VALUE < 25 && BLUE_VALUE >= 0 && BLUE_VALUE < 20)
122   {
123     COLOR_NAME = "Brown";
124     RED_LED_VALUE=70;
125     GREEN_LED_VALUE=12;
126     BLUE_LED_VALUE=0;
127   }
```

Drawing 35 A piece of code responsible for the condition for the brown color, as well as the modification of the value for the RGB LED.

```
128   else if(RED_VALUE > 0 && RED_VALUE < 50 && GREEN_VALUE > 130 && GREEN_VALUE < 170 && BLUE_VALUE > 60 && BLUE_VALUE < 115)
129   {
130     COLOR_NAME = "Green";
131     RED_LED_VALUE=0;
132     GREEN_LED_VALUE=75;
133     BLUE_LED_VALUE=0;
134   }
```

Drawing 36 A piece of code responsible for the condition for the green color, as well as the modification of the value for the RGB LED.

```
135   else if(RED_VALUE > 120 && RED_VALUE < 170 && GREEN_VALUE > 170 && GREEN_VALUE < 210 && BLUE_VALUE > 100 && BLUE_VALUE < 150)
136   {
137     COLOR_NAME = "Light Green";
138     RED_LED_VALUE=54;
139     GREEN_LED_VALUE=130;
140     BLUE_LED_VALUE=18;
141   }
```

Drawing 37 A piece of code responsible for the condition for the light green color, as well as the modification of the value for the RGB LED.

```
142   else if(RED_VALUE > 175 && RED_VALUE < 205 && GREEN_VALUE > 120 && GREEN_VALUE < 165 && BLUE_VALUE > 160 && BLUE_VALUE < 200)
143   {
144     COLOR_NAME = "Pink";
145     RED_LED_VALUE=148;
146     GREEN_LED_VALUE=7;
147     BLUE_LED_VALUE=50;
148   }
```

Drawing 38 A piece of code responsible for the condition for the color pink, as well as the modification of the value for the RGB LED.

```
149   else if(RED_VALUE > 180 && RED_VALUE < 220 && GREEN_VALUE > 180 && GREEN_VALUE < 220 && BLUE_VALUE > 100 && BLUE_VALUE < 140)
150   {
151     COLOR_NAME = "Yellow";
152     RED_LED_VALUE=66;
153     GREEN_LED_VALUE =55;
154     BLUE_LED_VALUE=0;
155   }
```

Drawing 39 A piece of code responsible for the condition for the color yellow, as well as the modification of the value for the RGB LED.

```
156   else
157   {
158     COLOR_NAME = "Different Color";
159   }
```

Drawing 40 A piece of code responsible for the default condition (Different Color).

```
161   analogWrite(RED, 255-RED_LED_VALUE);
162   analogWrite(GREEN, 255-GREEN_LED_VALUE);
163   analogWrite(BLUE, 255-BLUE_LED_VALUE);
```

Drawing 41 A piece of code responsible for controlling the RGB LED.

This code snippet (Figure 41) uses **the analogWrite()** functions to control the RGB LEDs depending on the color they detect. Here's how it works:

analogWrite(RED, 255-RED_LED_VALUE);

Red RGB LED intensity control. The RED_LED_VALUE value was previously calculated based on the color reading. The **analogWrite()** function ranges from 0 (no light) to 255 (full brightness). Since 0 is assumed to be full brightness and 255 to be no light, the value **of 255 – RED_LED_VALUE** is used to reverse this logic.

analogWrite(GREEN, 255-GREEN_LED_VALUE);

Analogous to the red LED, this line of code controls the intensity of the green RGB LED.

analogWrite(BLUE, 255 - BLUE_LED_VALUE);

Analogous to the red and green LEDs, this line of code controls the intensity of the blue RGB LED.

```
166   lcd.setCursor(0, 0);
167   lcd.print("R=");
168   lcd.print(RED_VALUE);
169   lcd.print("G=");
170   lcd.print(GREEN_VALUE);
171   lcd.print("B=");
172   lcd.print(BLUE_VALUE);
173   lcd.setCursor(0, 1);
174   lcd.print(COLOR_NAME);
175   delay(300);
176   lcd.clear();
```

Drawing 42 A piece of code responsible for displaying information on the LCD display.

This code snippet (Figure 42) uses the following functions:

- **lcd.setCursor(0, 0);**

Sets the cursor on the LCD screen to (0, 0), which indicates the first character of the first line of the screen.

- **lcd.print("R=");**

Displays the text "R=" on the LCD screen.

- **lcd.print(RED_VALUE);**

Displays the red color (**RED_VALUE**) value on the LCD screen, which was previously read from the sensor.

- **lcd.print("G=");**

Displays the text "G=" on the LCD screen.

- **lcd.print(GREEN_VALUE);**

Displays the green color value (**GREEN_VALUE**) read from the sensor on the LCD screen.

- **lcd.print("B=");**

Displays the text "B=" on the LCD screen.

- **lcd.print(BLUE_VALUE);**

Displays the blue color value (**BLUE_VALUE**) read from the sensor on the LCD screen.

- **lcd.setCursor(0, 1);**

Moves the cursor on the LCD screen to (0, 1), which indicates the first character of the second line of the screen.

- **lcd.print(COLOR_NAME);**

Displays the name of the identified color (**COLOR_NAME) on the LCD screen**, which was previously assigned in the if-else conditions.

- **delay(300);**

It causes a delay on the LCD screen for 300 milliseconds, which allows the user to see the information on the screen for a period of time.

- **lcd.clear();**

Clears the LCD screen to prepare it for the next batch of data. This is used so that the new data does not overlap with the previous data.

6. Test measurements

6.1 Conditions for safe use of the device

To avoid damage to health, including burns or electric shock, the following recommendations should be followed.

1. Temperature:
   - Avoid extreme temperature conditions that can affect the accuracy of measurements.
2. Moisture Protection:
   - Ensure that the detector is protected from moisture, especially if used in a high humidity environment.
3. Purity:
   - Keep the detector clean to avoid interference from contamination on the detector surface.
4. Electrical Safety:
   - Check that the detector is electrically safe and that it meets all electrical safety standards.
5. Transport and storage:
   - During transport and storage, take care to protect the detector against shocks and mechanical damage.

6.2 Device Tests

Using the device, the RGB component values for several basic colors were measured. The results of the measurements are described in the table below.

Table 2 Colors obtained with the detector.

Colour	R	G	B	Colour obtained	True color	R	G	B
Black	0	0	0			0	0	0
Blue	0	162	198			87	172	247
Brown	73	19	0			88	63	38
Green	23	154	89			52	137	69
Light Blue	93	198	210			103	196	215
Light Green	155	191	133			173	218	83
Orange	180	81	31			245	138	67
Pink	191	143	186			245	155	197
Red	152	0	0			198	56	41
White	197	212	218			255	255	255
Yellow	198	192	114			230	208	6

Based on the results obtained, it can be concluded that the values obtained by the sensor are close to the original colors, but they are darkened.

The photos show the color readings that were obtained with our device on sheets of different colors. Figures 43 to 53 show the readings for black, blue, brown, green, light blue, light green, orange, pink, red, white, and yellow. Pay special attention to the color of the LED, which reflects the color of the card

Drawing 43 Readout for black.

Drawing 44 Readout for blue.

Drawing 45 Readout for brown.

Drawing 46 Readout for green.

Drawing 47 Light Blue Readout

Drawing 48 Light Green Readout

Drawing 49 Reading for orange

Drawing 50 Readout for pink

Drawing 51 Readout for red.

Drawing 52 Readout for white.

Drawing 53 Readout for yellow.

Device specifications

Assuming that the chip consumes 90 mA and the battery has a capacity of 500 mAh at 9V, we can estimate that the chip should work continuously for about 5.56 hours. This calculation results from dividing the battery capacity by the current consumption:

$$\frac{500mAh}{90mA} \approx 5{,}56\ h.$$

However, it is worth remembering that these are theoretical calculations and the actual operating time may vary depending on many factors, such as energy conversion efficiency, ambient temperature or battery condition.

Table 3 Appliance Rating Parameters.

Parameter	Symbol	Min.	Type.	Max.	Unit
Supply voltage	Ucc	5	5	12	V
Power consumption	Icc	60	85	90	but
Length	X	-	138	-	mm
Width	And	-	155	-	mm
Height	With	-	58	-	mm
Red component value	R	0	-	255	-
Green component value	G	0	-	255	-
Blue component value	B	0	-	255	-

7. RGB component detector user manual:

Starting the device

To run the RGB component detector, follow these steps:

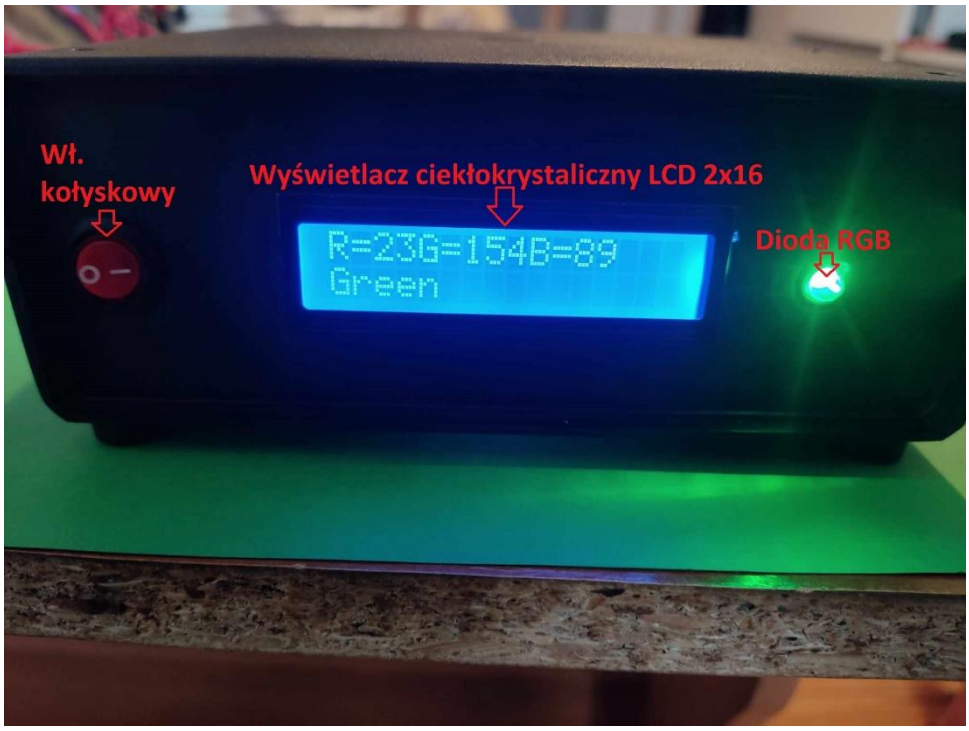
Key Setting: Locate the device switch and turn it to the "ON" position. This key is responsible for turning on the detector.

Color Testing: Place the detector on the color to be tested. The RGB component detector will automatically examine and record the color components, presenting the results on the LCD display.

Shutting down the device

To safely disable the RGB component detector, follow these guidelines:

Key Setting: Find the device switch and move it to the "OFF" position. This step protects the detector from further battery drain and turns off the device.



Drawing 1 Front image of the device

8. Summary

During the implementation of the project, we gained numerous valuable skills and expanded our knowledge in the field of optoelectronics, which was the main goal of this semester-long project. All project assumptions were successfully implemented, and work on it was carried out smoothly and systematically throughout the semester.

The fulfilled assumptions include an RGB component detector, the use of an LCD display for clear presentation of results, battery power for mobility and energy independence, and an RGB diode emitting light close to the analyzed color, ensuring precise measurements. In addition, the design incorporates an ergonomic design, adapted for comfortable use.

During the project, we encountered several significant challenges that we had to confront with the effectiveness of the RGB component detector. One of the main problems was the process of adapting the RGB LED to our expectations. It turned out that achieving the desired intensity and colour of light in the three colour components required precise tuning of the parameters, which was a technical challenge.

Another important issue was to adapt the color sensor to the environment. Placing the sensor directly under the detector housing was intended to minimize the influence of external factors. An additional problem turned out to be determining the optimal distance between the sensor and the tested element. The illuminating diodes, if they were too close, introduced significant errors, while too long distances made the measurements ineffective.

Solving these problems required repeated experimentation, testing, and iteration. Different combinations of parameters had to be used, as well as the electronic and mechanical systems of the detector had to be adapted.

In the context of possible modifications, we are considering potential improvements. These can include adding wireless connectivity such as Bluetooth or Wi-Fi, allowing you to remotely monitor and control the detector using your smartphone or computer. Expanding the user interface on the display would allow access to various operating modes, settings or measurement history. In addition, the design could be weatherproofed, allowing for safe use in a variety of environmental conditions. The introduction of a solar panel or other renewable energy sources to charge the battery would increase the energy independence of the detector.

Bibliography

*Knowledge:

1. https://www.rapidtables.com/web/color/RGB_Color.html
2.https://botland.com.pl/czujniki-swiatla-i-koloru/4932-czujnik-koloru-przetwornik-swiatlo-czestotliwosc-tcs3200-waveshare-9520-5904422374211.html
3.https://botland.com.pl/index.php?controller=attachment&id_attachment=185
4.https://www.arduino.cc/reference/en/language/functions/math/map/
5.https://audiodesign.info.pl/diody-led-przewlekane/DIODA-LED-5mm-RGB-WA-.html
6.https://www.piekarz.pl/40247-bateria-energizer-6lr61-energizer-max-plus-9v-blister/
7.https://botland.com.pl/wyswietlacze-alfanumeryczne-i-graficzne/19732-wyswietlacz-lcd-2x16-znakow-niebieski-ze-zlaczami-justpi-5903351243131.html?cd=18298825138&ad=&kd=&gad_source=1&gclid=CjwKCAiAkp6tBhB5EiwANTCx1H-GfvNyHQdA5LCNkZ9w4_wtmfIEy_63p6gFy2m4YMoWI2nRjxxbARoCMSAQAvD_BwE
8.https://efizyka.net.pl/fale-elektromagnetyczne-widmo-fal-elektromagnetycznych
9.https://botland.com.pl/czujniki-swiatla-i-koloru/4932-czujnik-koloru-przetwornik-swiatlo-czestotliwosc-tcs3200-waveshare-9520-5904422374211.html
10.https://forbot.pl/blog/kurs-arduino-silniki-pwm-serwomechanizm-zewnetrzne-biblioteki-id3913
11.https://html.alldatasheet.com/html-pdf/785114/ENERGIZER/6LR61/484/1/6LR61.html
12.https://www.bing.com/ck/a?!&&p=31ddef02894de022JmltdHM9MTcwNTYyMjQwMCZpZ3VpZD0zMTIzMTBlMC1iNjg4LTY0MWItMGY5Zi0wMmM1YjdmMDY1YjcmaW5zaWQ9NTIyNw&ptn=3&ver=2&hsh=3&fclid=312310e0-b688-641b-0f9f-02c5b7f065b7&psq=wikipedia+rgb&u=a1aHR0cHM6Ly9wbC53aWtpcGVkaWEub3JnL3dpa2kvUkdC&ntb=1

*Photos:

[1] – https://dmfizyka.online/?p=347

[2] –https://www.pikpng.com/pngvi/xhhoio_alt-text-rgb-led-color-mixing-chart-clipart/

[3] –https://botland.com.pl/czujniki-swiatla-i-koloru/4932-czujnik-koloru-przetwornik-swiatlo-czestotliwosc-tcs3200-waveshare-9520-5904422374211.html

[4] – https://www.mcselec.com/index.php?option=com_content&task=view&id=329&Itemid=105

[5] – https://botland.com.pl/czujniki-swiatla-i-koloru/4932-czujnik-koloru-przetwornik-swiatlo-czestotliwosc-tcs3200-waveshare-9520-5904422374211.html

[6] –https://botland.com.pl/czujniki-swiatla-i-koloru/4932-czujnik-koloru-przetwornik-swiatlo-czestotliwosc-tcs3200-waveshare-9520-5904422374211.html

[7] –https://botland.com.pl/czujniki-swiatla-i-koloru/4932-czujnik-koloru-przetwornik-swiatlo-czestotliwosc-tcs3200-waveshare-9520-5904422374211.html

[8] –https://botland.com.pl/czujniki-swiatla-i-koloru/4932-czujnik-koloru-przetwornik-swiatlo-czestotliwosc-tcs3200-waveshare-9520-5904422374211.html

[9]–https://allegro.pl/oferta/arduino-uno-rev3-a000066-oryginal-8203566183

[10]–https://archiwum.allegro.pl/oferta/wyswietlacz-lcd-1602-2x16-hd44780-blue-arduino-i7384561661.html

[11] –https://botland.com.pl/diody-led-rgb/543-dioda-led-5mm-rgb-wsp-anoda-5-szt-5903351244176.html

[12] –https://www.piekarz.pl/40247-bateria-energizer-6lr61-energizer-max-plus-9v-blister/

[13] –https://electronics.stackexchange.com/questions/382584/running-12v-pc-fan-with-9v-battery

[14] –https://www.masterled.pl/wlacznik-kolyskowy-okragly-12v-podswietlany-duplikat-2.html

*Own materials:

1.Pictures of Testing Machine Drawing (43-53)
2.Flow Chart - Figure 3
3.Schematic diagram - Figure 8
4. Program algorithm - Figure 21
5.Photographs of the device Drawing (4-7)

Extras

Listing code for the project:

```
1 /*
2 * Source code for the "RGB component detector" project.
3 * Version 19.01.2024
4 * Authors: Radosław Mierzwa , Pavlo Kostushevych , Mateusz Gwioździk
5 Circuit Connection Description:
7 A1 - Arduino UNO rev3
8 U1 - TCS300
9  DS1 - LCD
10 D1 - dioda RGB
11
12  +5V A1_VIN
13  A1_3V3 U1_VDD
14  A1_A0 DS1_E
15  A1_A1 DS1_D5
16  A1_A2 DS1_D6
17  A1_D2 DS1_RS
18  A1_D3 D1_B
19  A1_D4 DS1_D4
20  A1_D5 D1_G
```

```
21  A1_D6 D1_R
22  A1_D7 DS1_D7
23  A1_D8 U1_OUT
24  A1_D9 U1_S0
25  A1_D10 U1_S1
26  A1_D11 U1_S2
27  A1_D12 U1_S3
28  A1_D12 U1_OE
29  A1_GND GND
30  */
31
32 // Declaration of pins connected to the LCD
33 #include <LiquidCrystal.h>
34  LiquidCrystal lcd(2, A0, 4, A1, A2, 7);
35
36 // Declaration of pins connected to the TCS3200 sensor
37  #define S0 9
38  #define S1 10
39  #define S2 11
40  #define S3 12
41  #define OUT 8
42  #define LED 13
43
44 // Declaration of pins connected to the RGB LED
45  #define RED 6
46 #define GREEN 5
47  #define BLUE 3
48
49 // Declaration of variables used in the project
50  long int HALF_PERIOD = 0;
51 int RED_VALUE = 0;
52 int GREEN_VALUE = 0;
53 int BLUE_VALUE = 0;
54 int RED_LED_VALUE = 0;
55 int GREEN_LED_VALUE = 0;
```

```
int BLUE_LED_VALUE = 0;
 String COLOR_NAME;

 void setup() {

// Declaration of the mode of operation of the pins connected to the
sensor
 pinMode(S0, OUTPUT);
 pinMode(S1, OUTPUT);
 pinMode(S2, OUTPUT);
 pinMode(S3, OUTPUT);
 pinMode(LED, OUTPUT);
 pinMode(OUT, INPUT);

// Initialization of the display and its dimensions
 lcd.begin(16, 2);

// Setting the frequency scaling of the sensor output signal to 20%
digitalWrite(S0, HIGH);
digitalWrite(S1, LOW);

// Lighting up the diodes placed in the sensor module
 digitalWrite(LED,HIGH );

// Start serial transfer at 9600 bps
 Serial.begin(9600);
 }

 void loop() {
// START OF RED COMPONENT MEASUREMENTS

// Setting the reading to the red component
digitalWrite(S2, LOW);
digitalWrite(S3, LOW);

```

```
90 // Sensor Output Half Period Reading
91  HALF_PERIOD = pulseIn(OUT, LOW);
92
93 // Reversal of the received value
94  RED_VALUE = 255-HALF_PERIOD;
95
96 // Truncation of values below 0
97  if (RED_VALUE < 0) {
98    RED_VALUE = 0;
99  }
100
101 // Print the red component value on the serial monitor
102  Serial.print("R = ");
103  Serial.print(RED_VALUE);
104  Serial.print("   ");
105
106 // Wait 100 ms
107  delay(100);
108
109 // START OF MEASUREMENTS OF THE GREEN COMPONENT
110
111 // Setting the reading to the green component
112 digitalWrite(S2, HIGH);
113 digitalWrite(S3, HIGH);
114
115 // Sensor Output Half Period Reading
116  HALF_PERIOD = pulseIn(OUT, LOW);
117
118 // Reversal of the received value
119  GREEN_VALUE = 255-HALF_PERIOD;
120
121 // Truncation of the value below 0
122  if (GREEN_VALUE < 0) {
123    GREEN_VALUE = 0;
124  }
```

```
125
126 // Print the green component on the serial monitor
127  Serial.print("G = ");
128  Serial.print(GREEN_VALUE);
129  Serial.print("   ");
130
131 // Wait 100 ms
132  delay(100);
133
134 // START OF CELESTIAL COMPONENT MEASUREMENTS
135
136 // Setting the reading to the blue component
137 digitalWrite(S2, LOW);
138 digitalWrite(S3, HIGH);
139
140 // Sensor Output Half Period Readout
141  HALF_PERIOD = pulseIn(OUT, LOW);
142
143 // Reversal of the received value
144  BLUE_VALUE = 255-HALF_PERIOD;
145
146 // Truncation of the value below 0
147  if (BLUE_VALUE < 0) {
148    BLUE_VALUE = 0;
149  }
150
151 // Print the value of the blue component on the serial monitor
152  Serial.print("B = ");
153  Serial.print(BLUE_VALUE);
154  Serial.println("   ");
155
156 // Wait 100 ms
157  delay(100);
158
```

```
159 // Assigning the variables responsible for the color of the RGB diode
to the obtained values

160  RED_LED_VALUE=RED_VALUE;

161  GREEN_LED_VALUE=GREEN_VALUE;

162  BLUE_LED_VALUE=BLUE_VALUE;

163

164 // Determination of the name of the obtained color based on estimated
limit values and change of the pin values of the RGB diode for hard-to-
obtain colors

165 if(RED_VALUE < 30 && GREEN_VALUE < 30 && BLUE_VALUE < 30)

166  {

167 COLOR_NAME = "Black";

168  }

169  else if(RED_VALUE > 165 && GREEN_VALUE > 190 && BLUE_VALUE > 190)

170  {

171    COLOR_NAME = "White";

172  }

173 else if(RED_VALUE >= 0 && RED_VALUE < 20 && GREEN_VALUE > 130 &&
GREEN_VALUE < 180 && BLUE_VALUE > 180 && BLUE_VALUE < 255)

174  {

175 COLOR_NAME = "Blue";

176  }

177 else if(RED_VALUE > 45 && RED_VALUE < 160 && GREEN_VALUE > 170 &&
GREEN_VALUE < 210 && BLUE_VALUE > 195 && BLUE_VALUE < 225)

178  {

179    COLOR_NAME = "Light Blue";

180  }

181 else if(RED_VALUE > 140 && RED_VALUE < 225 && GREEN_VALUE >= 0 &&
GREEN_VALUE < 20 && BLUE_VALUE >= 0 && BLUE_VALUE < 28)

182  {

183 COLOR_NAME = "Red";

184    BLUE_LED_VALUE=0;

185  }

186 else if(RED_VALUE > 160 && RED_VALUE < 205 && GREEN_VALUE > 60 &&
GREEN_VALUE < 115 && BLUE_VALUE > 25 && BLUE_VALUE < 90)

187  {

188    COLOR_NAME = "Orange";
```

```
189    RED_LED_VALUE=160;
190    GREEN_LED_VALUE=37;
191    BLUE_LED_VALUE=0;
192  }
193 else if(RED_VALUE > 40 && RED_VALUE < 90 && GREEN_VALUE >= 0 &&
GREEN_VALUE < 25 && BLUE_VALUE >= 0 && BLUE_VALUE < 20)
194  {
195 COLOR_NAME = "Brown";
196    RED_LED_VALUE=70;
197    GREEN_LED_VALUE=12;
198    BLUE_LED_VALUE=0;
199  }
200 else if(RED_VALUE > 0 && RED_VALUE < 50 && GREEN_VALUE > 130 &&
GREEN_VALUE < 170 && BLUE_VALUE > 60 && BLUE_VALUE < 115)
201  {
202 COLOR_NAME = "Green";
203    RED_LED_VALUE=0;
204    GREEN_LED_VALUE=75;
205    BLUE_LED_VALUE=0;
206  }
207 else if(RED_VALUE > 120 && RED_VALUE < 170 && GREEN_VALUE > 170 &&
GREEN_VALUE < 210 && BLUE_VALUE > 100 && BLUE_VALUE < 150)
208  {
209    COLOR_NAME = "Light Green";
210    RED_LED_VALUE=54;
211    GREEN_LED_VALUE=130;
212    BLUE_LED_VALUE=18;
213  }
214 else if(RED_VALUE > 175 && RED_VALUE < 205 && GREEN_VALUE > 120 &&
GREEN_VALUE < 165 && BLUE_VALUE > 160 && BLUE_VALUE < 200)
215  {
216 COLOR_NAME = "Pink";
217    RED_LED_VALUE=148;
218    GREEN_LED_VALUE=7;
219    BLUE_LED_VALUE=50;
220  }
```

```
221 else if(RED_VALUE > 180 && RED_VALUE < 220 && GREEN_VALUE > 180 &&
GREEN_VALUE < 220 && BLUE_VALUE > 100 && BLUE_VALUE < 140)
222  {
223    COLOR_NAME = "Yellow";
224    RED_LED_VALUE=66;
225    GREEN_LED_VALUE =55;
226    BLUE_LED_VALUE=0;
227  }
228  else
229  {
230    COLOR_NAME = "Different Color";
231  }
232
233 // Setting the values of the pins connected to the RGB LED
234 analogWrite(RED, 255-RED_LED_VALUE);
235 analogWrite(GREEN, 255-GREEN_LED_VALUE);
236 analogWrite(BLUE, 255 - BLUE_LED_VALUE);
237
238 // Display of RGB component values and color name on the display
239 lcd.setCursor(0, 0);
240  lcd.print("R=");
241  lcd.print(RED_VALUE);
242  lcd.print("G=");
243  lcd.print(GREEN_VALUE);
244  lcd.print("B=");
245  lcd.print(BLUE_VALUE);
246 lcd.setCursor(0, 1);
247  lcd.print(COLOR_NAME);
248
249 // Wait 300 ms
250  delay(300);
251
252 // Clear the display
253  lcd.clear();
254  }
```

255