KY -016 RGB SENSOR INTERFACING WITH RUGGED BOARD

Presented By
D.Mohan Kumar
SBCS India pvt ltd

Table of Contents

1.Introduction

2.0 HardWare Components

2.1 KY-016 RGB Sensor

2.1.1RGB LED Module

2.1.2RGB Sensor Pin Configuration

2.2 Rugged Board

3.0 software components

3.1 MINICOM

4.0 KY-016 RGBSensor Interfacing to Rugged Board

5.0 Source Code

5.1 Rugged Board Code Using Sys/Class

6.0 Introduction to MRAA

6.1 Key features of MRAA include

6.2 MRAA Code

7.0 Expected Output

8.0 Real Time Applications

9.0 Referance links

10.0 Project Conclusion

1.0 Introduction

The KY-016 RGB module is a popular color sensor module used in electronics projects, particularly with microcontrollers like Arduino. This module is designed to detect and measure the intensity of red, green, and blue light, allowing it to identify and distinguish different colors.

- The KY-016 module typically consists of a small PCB (Printed Circuit Board) with an RGB LED and a light-dependent resistor (LDR) for each color channel (red, green, and blue). These components work together to measure the ambient light's color composition.

- The module works on the principle that different colors absorb and reflect light differently. The RGB LED emits light, and the LDRs detect the reflected light intensity for each color. The module then generates analog signals proportional to the intensity of red, green, and blue light.

- The KY-016 module usually provides three analog output signals, one for each color channel (R, G, B). These signals can be read by a microcontroller (e.g., Arduino) to determine the color composition in the environment.

- The module typically has four pins: VCC (power), GND (ground), OUT (analog output), and an optional LED pin for powering an onboard LED. Connect the VCC and GND to the power and ground of your microcontroller, respectively. The OUT pin is connected to an analog input pin to read the color values.

- By reading the analog values from the three color channels, you can interpret the color information. Different colors will produce varying combinations of intensity in the red, green, and blue channels.

- While the KY-016 is suitable for basic color detection, it may not provide highly accurate color measurements. For more precise color sensing applications, specialized color sensors or spectrometers might be more appropriate.

2.0 HardWare Components:-

2.1 KY-016 RGB Sensor :-

- An RGB LED color sensor is a type of electronic device that combines RGB l**ight-emitting diodes** (LEDs) with a color sensor to detect and analyze colors in a given environment or on a specific object.

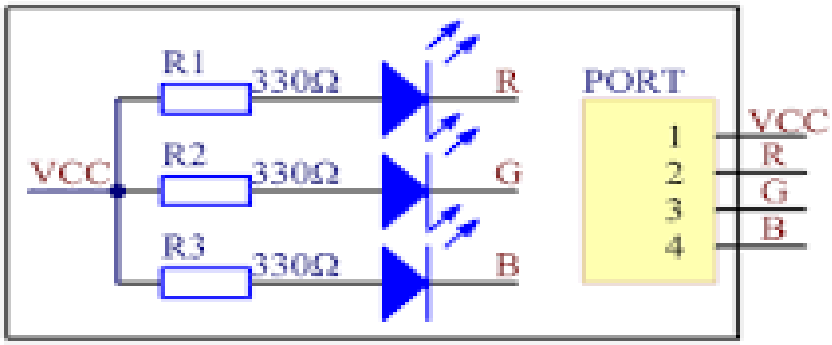
- The **KY-016 F**ull Color RGB LED emits a wide range of different colors by mixing red, green and blue light. Compatible with many popular microcontrollers like Arduino, Raspberry Pi and ESP32.

- Every pixel in a display is composed of red, green and blue (RGB) subpixels that light up at **different intensities to create different colors**. This color space is a color representation method used in electronic displays, like televisions, computer monitors, digital cameras and various types of lighting.

- RGB is the best choice for anything online, like your website, ad posters or banners, and logos. RGB colors are generated through an additive method, and the three colors combined together make white.

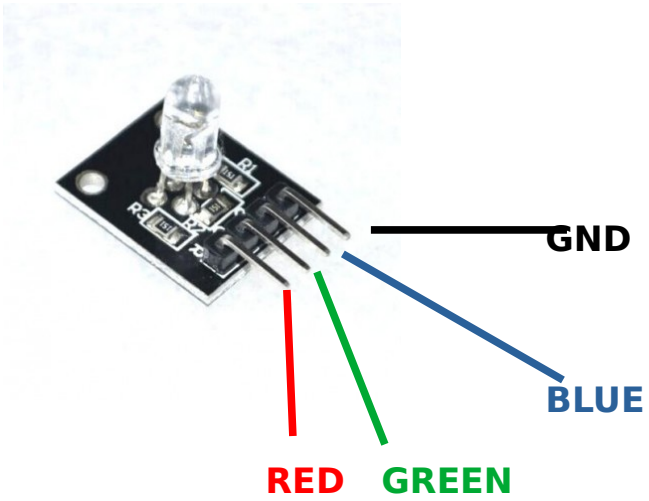
- There is nothing special about RGB lighting and LED lighting, light is light, and as long as the intensity of visible light is within safe limits, it will not harm anyone's eyes. If anything, RGB lights are like LED lights in that they are safer because they do not produce UV light.

2.1.1RGB LED Module :-



➢ The three primary colors, red, green, and blue, can be mixed and compose all kinds of colors by brightness, so you can make an RGB LED emit colorful light by controlling the circuit. RGB LEDs can be categorized into common anode LED and common cathode LED. In this module, we use a common anode RGB LED.

2.1.2RGB Sensor Pin Configuration :-



➢ The KY-016 RGB Sensor four Pins Are Avalible in Module . **GND**,RED,GREEN And BLUE.

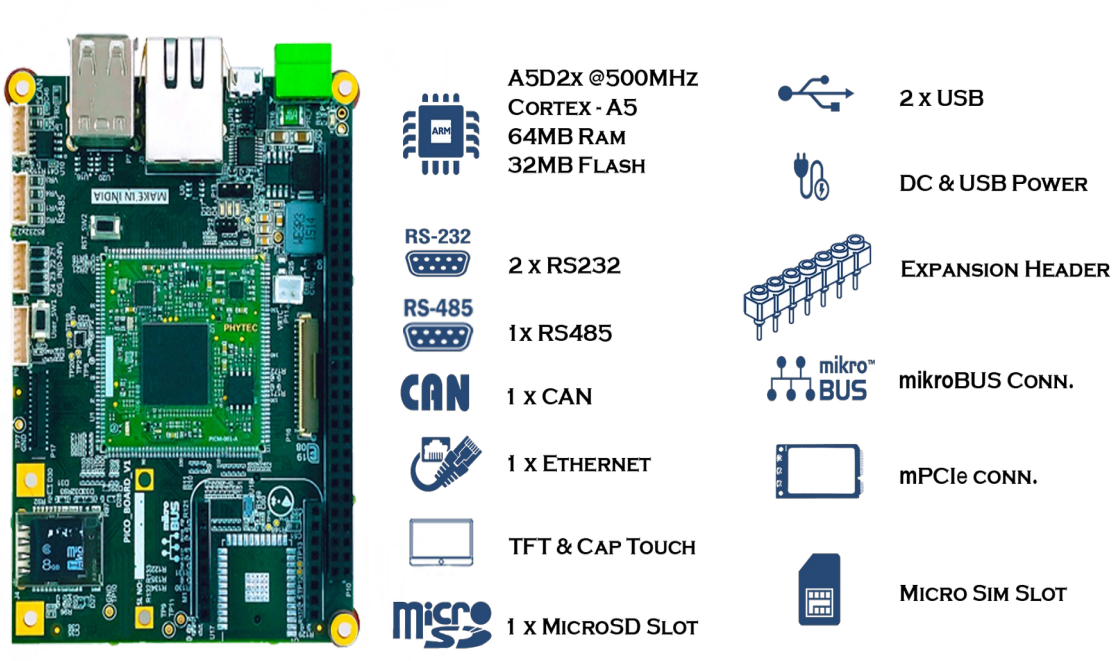
- **GND** (Ground): Connect this pin to the ground of your circuit.

- RED :- the red wire of the KY-016 RGB sensor is connected to the output pin on the board, it suggests that the red component of the color information is transmitted through the OUT pin.

- GREEN :- the Green Wire of the KY-016 RGB Sensor is Connected to the output pin on the Board ,It suggests that the Green Component of the Color information is transmitted through the OUT pin.

- BLUE:- the Blue Wire of the KY-016 RGB Sensor is Connected to the output pin on the Board ,It suggests that the Blue Component of the Color information is transmitted through the OUT pin.

2.2 Rugged Board :-



The term "rugged board" generally refers to a type of single-board computer (SBC) or embedded computing platform designed for use in

harsh or demanding environments. These boards are engineered to withstand challenging conditions such as extreme temperatures, vibrations, shock, humidity, and other environmental stresses. Rugged boards find applications in a variety of industries, including military, aerospace, industrial automation, transportation, and outdoor monitoring systems.

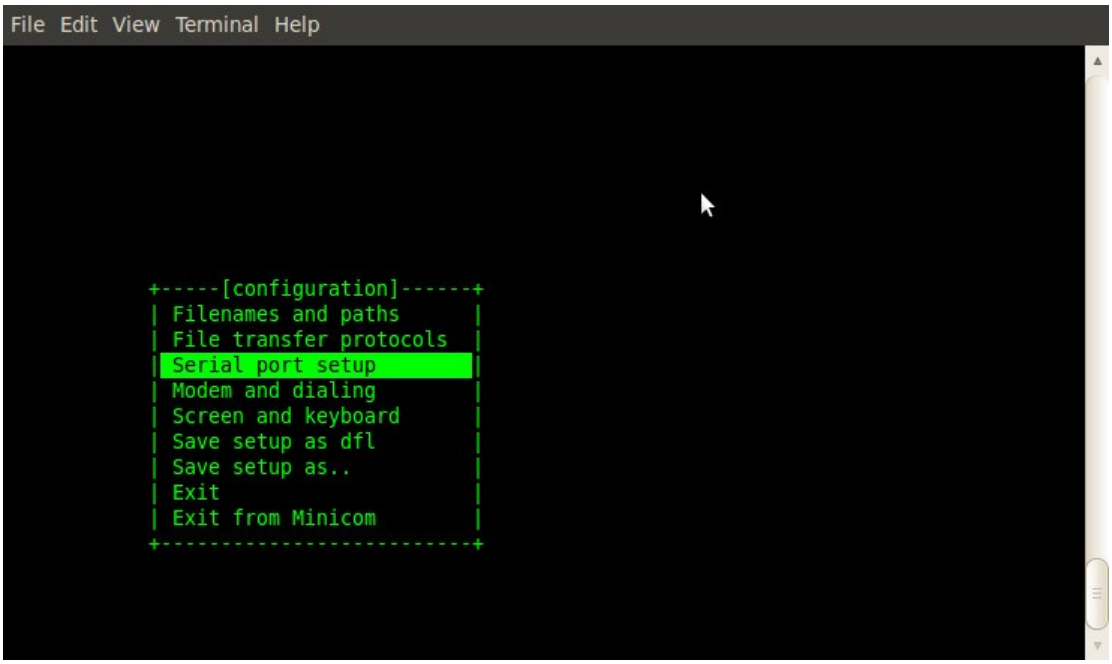
- Rugged boards are built to endure harsh conditions and maintain operational reliability in environments where standard electronic equipment might fail.
- Rugged boards are designed to operate over a wide temperature range, ensuring functionality in extreme cold or hot environments. They often have protective coatings to resist moisture, dust, and other environmental factors.
- Rugged boards often support a broad input voltage range, allowing them to work with different power sources and ensuring stability in scenarios where power fluctuations may occur.
- Rugged boards come equipped with processors suitable for embedded applications.

3.0 software components:-

3.1 MINICOM:-

- Minicom is a text-based serial communication program that is widely used in Unix-like operating systems for interacting with devices over serial ports. It is particularly popular for debugging and configuring devices, especially in projects involving microcontrollers, embedded systems, and other hardware components.

sudo minicom -s



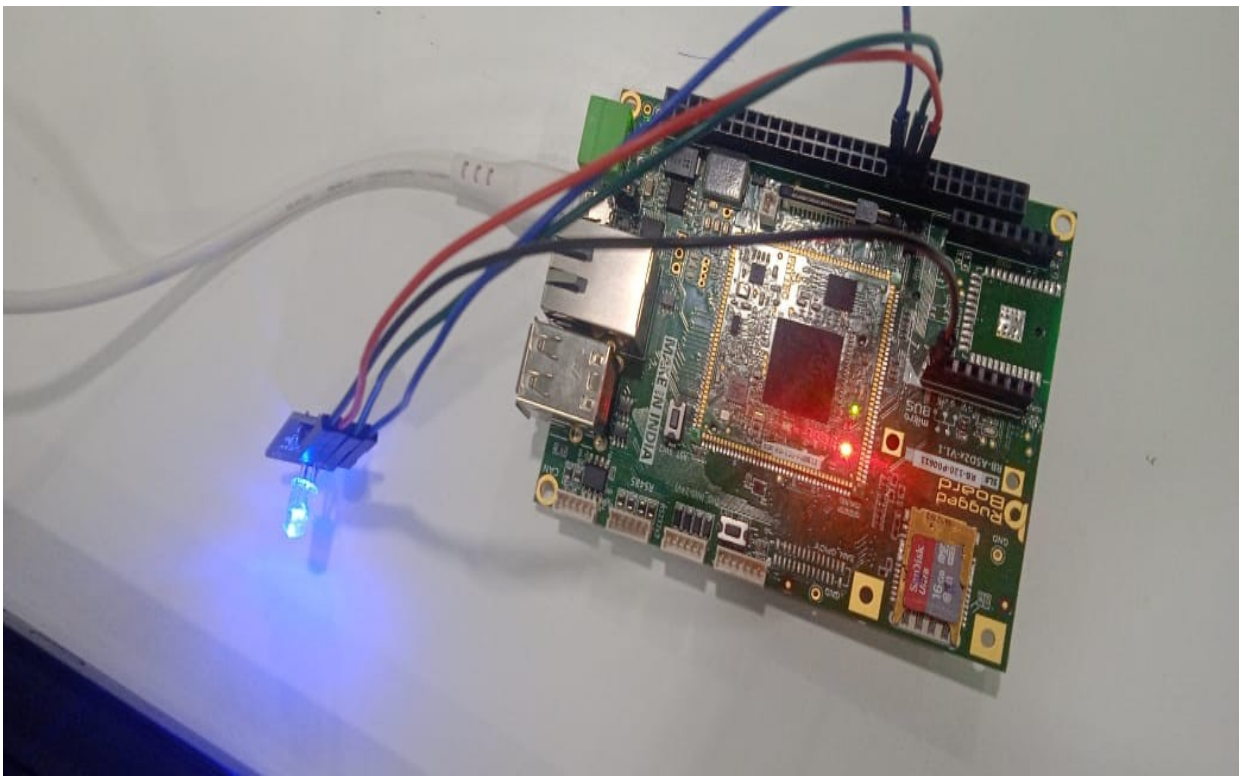
➢ Minicom also supports file transfer using protocols like Xmodem or Ymodem. This feature can be useful for tasks such as updating firmware or transferring files between your computer and the connected device.

```
minicom -D /dev/ttyUSB0
```

4.0 KY-016 RGBSensor Interfacing to Rugged Board :-

➢ If you are connecting an RGB sensor to a "rugged board," the specific pin configuration may depend on the type and model of the rugged board you are using. Rugged boards are often designed for harsh environments and may have different pin layouts or interfaces.

➢ Without knowing the exact specifications of the rugged board, it's challenging to provide precise details. However, I can give you a general guide on how you might approach connecting an RGB sensor to a microcontroller or rugged board:



- **GND** (Ground): Connect this pin to the Sensor Ground to Rugged Board Ground Connection.

- RED :- The red wire of the KY-016 RGB sensor is connected to the output pin on the Rugged board, Analog pin is PA29 is output.

- Green :- The Green wire of the KY-016 RGB sensor is connected to the output pin on the Rugged board, Analog pin is PA31 is output.

- Blue :- The Blue wire of the KY-016 RGB sensor is connected to the output pin on the Rugged board, Analog pin is PA13 is output.

5.0 Source Code

5.1 Rugged Board Code Using Sys/Class:-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
```

```
char export_path[100] = "/sys/class/gpio/export";
char direction_path_red[100] = "/sys/class/gpio/";
char direction_path_green[100] = "/sys/class/gpio/";
char direction_path_blue[100] = "/sys/class/gpio/";
char value_path_red[100] = "/sys/class/gpio/";
char value_path_green[100] = "/sys/class/gpio/";
char value_path_blue[100] = "/sys/class/gpio/";

void setupPin(int pin) {
    int fd_export = open(export_path, O_WRONLY);
    if (fd_export == -1) {
        perror("Error opening export");
        exit(EXIT_FAILURE);
    }

    char pin_str[4];
    sprintf(pin_str, "%d", pin);
    write(fd_export, pin_str, strlen(pin_str) + 1);
    close(fd_export);
}

void setupRGB(int red, int green, int blue) {
    setupPin(red);
    setupPin(green);
    setupPin(blue);

    sprintf(direction_path_red + 16, "%s%d%s", "PA", red, "/direction");
    sprintf(direction_path_green + 16, "%s%d%s", "PA", green, "/direction");
    sprintf(direction_path_blue + 16, "%s%d%s", "PA", blue, "/direction");

    sprintf(value_path_red + 16, "%s%d%s", "PA", red, "/value");
    sprintf(value_path_green + 16, "%s%d%s", "PA", green, "/value");
    sprintf(value_path_blue + 16, "%s%d%s", "PA", blue, "/value");

    int fd_direction_red = open(direction_path_red, O_WRONLY);
    int fd_direction_green = open(direction_path_green, O_WRONLY);
    int fd_direction_blue = open(direction_path_blue, O_WRONLY);

    if (fd_direction_red == -1 || fd_direction_green == -1 || fd_direction_blue == -1) {
        perror("Error opening direction");
        exit(EXIT_FAILURE);
    }

    write(fd_direction_red, "out", 4);
    write(fd_direction_green, "out", 4);
    write(fd_direction_blue, "out", 4);

 close(fd_direction_red);
    close(fd_direction_green);
    close(fd_direction_blue);
}
```

```
int main() {
    setupRGB(29,31,13);  // Assuming GPIO pins PA29, PA31, and PA13 for red, green, and blue, respectively

   int fd_red = open(value_path_red, O_WRONLY);
   int fd_green = open(value_path_green, O_WRONLY);
   int fd_blue = open(value_path_blue, O_WRONLY);

   if (fd_red == -1 || fd_green == -1 || fd_blue == -1) {
      perror("Error opening value");
      exit(EXIT_FAILURE);
   }

  while (1) {
      // Turn on the red LED
      write(fd_red, "1",2); 2);
      usleep(500000);  // Sleep for 500 ms

      // Turn off the red LED
      write(fd_red, "0", 2);

      // Turn on the green LED
      write(fd_green, "1", 2);
      usleep(500000);  // Sleep for 500 ms

      // Turn off the green LED
      write(fd_green, "0", 2);

      // Turn on the blue LED
      write(fd_blue, "1", 2);
      usleep(500000);  // Sleep for 500 ms

   }

   close(fd_red);
   close(fd_green);
   close(fd_blue);

   return 0;
   }
```

6.0 Introduction to MRAA :-

- MRAA (pronounced em-ra) is a low-level library written in C that provides a consistent API (Application Programming Interface) for interfacing with various hardware platforms, including platforms such as Intel Galileo, Intel Edison, and other embedded

systems. MRAA is primarily used in the context of IoT (Internet of Things) and embedded systems development.

➢ The primary goal of MRAA is to abstract the complexity of interacting with various I/O (Input/Output) interfaces and sensors, providing a unified interface regardless of the underlying hardware. It allows developers to write code that can be easily ported across different hardware platforms without significant modifications.

6.1 Key features of MRAA include:

- **GPIO (General Purpose Input/Output) Support:** MRAA provides a consistent interface for working with GPIO pins, allowing developers to read and write digital values.
- **Analog I/O Support:** MRAA supports analog input and output, making it easier to interface with analog sensors and devices.
- **PWM (Pulse Width Modulation):** MRAA allows you to generate PWM signals, which is useful for tasks such as controlling the brightness of LEDs or the speed of motors.
- **I2C and SPI Support:** MRAA provides functions for communicating over I2C and SPI buses, common communication protocols used in embedded systems.
- **UART (Universal Asynchronous Receiver-Transmitter):** MRAA supports serial communication, which is essential for interfacing with devices that communicate over UART.

6.2 MRAA Code :-

```
#include <stdio.h>
#include <unistd.h>
#include <mraa.h>

mraa_gpio_context gpio_red, gpio_green, gpio_blue;

void setupPin(int pin) {
   mraa_gpio_context context = mraa_gpio_init(pin);
   if (context == NULL) {
      fprintf(stderr, "Error initializing GPIO %d\n", pin);
      exit(EXIT_FAILURE);
   }

   mraa_result_t result = mraa_gpio_dir(context, MRAA_GPIO_OUT);
   if (result != MRAA_SUCCESS) {
      fprintf(stderr, "Error setting direction for GPIO %d\n", pin);
      mraa_result_print(result);
      exit(EXIT_FAILURE);
   }
}
```

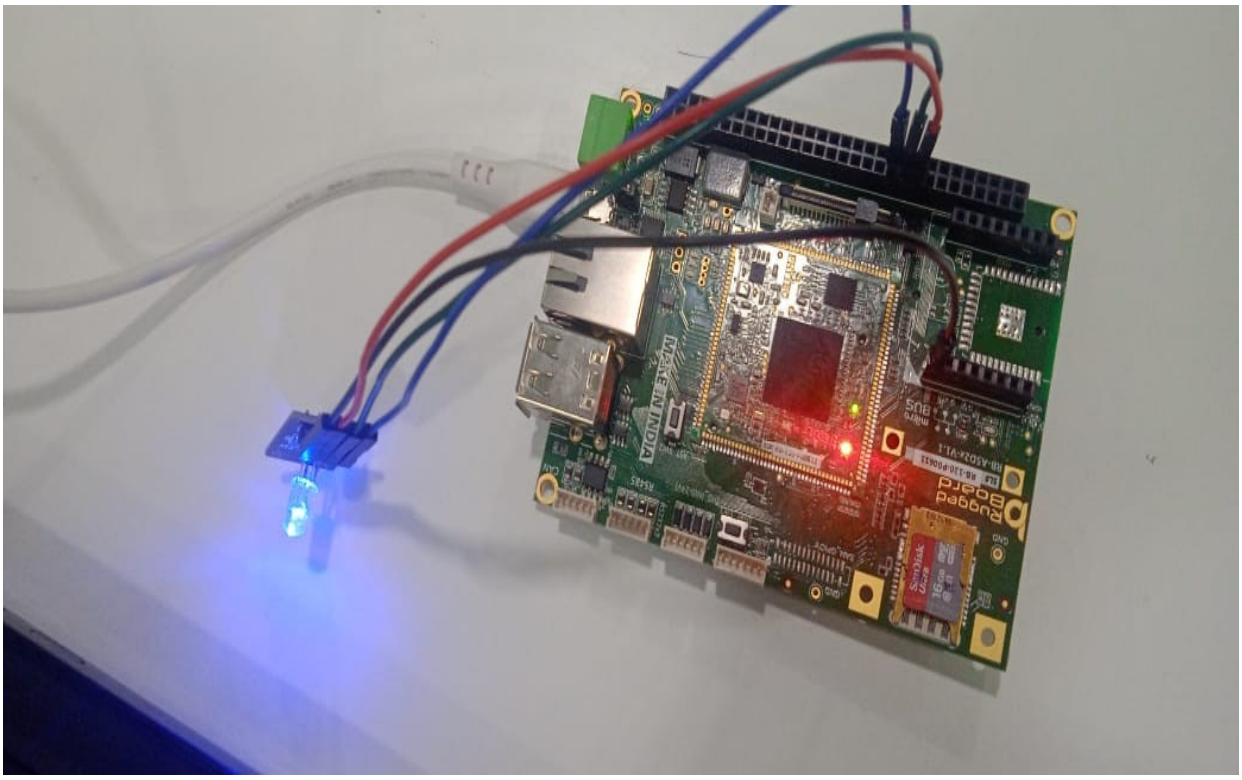
```
void setupRGB(int red, int green, int blue) {
    setupPin(red);
    setupPin(green);
    setupPin(blue);

    gpio_red = mraa_gpio_init(red);
    gpio_green = mraa_gpio_init(green);
    gpio_blue = mraa_gpio_init(blue);

    if (gpio_red == NULL || gpio_green == NULL || gpio_blue == NULL) {
        fprintf(stderr, "Error initializing GPIOs\n");
        exit(EXIT_FAILURE);
    }
}
    mraa_gpio_dir(gpio_red, MRAA_GPIO_OUT);
    mraa_gpio_dir(gpio_green, MRAA_GPIO_OUT);
    mraa_gpio_dir(gpio_blue, MRAA_GPIO_OUT);
}

int main() {
    setupRGB(40, 42, 44);

    while (1) {
        // Turn on the red LED
        mraa_gpio_write(gpio_red, 1);
        usleep(500000);  // Sleep for 500 ms

        // Turn off the red LED
        mraa_gpio_write(gpio_red, 0);

        // Turn on the green LED
        mraa_gpio_write(gpio_green, 1);
        usleep(500000);  // Sleep for 500 ms

        // Turn off the green LED
        mraa_gpio_write(gpio_green, 0);

        // Turn on the blue LED
        mraa_gpio_write(gpio_blue, 1);
        usleep(500000);  // Sleep for 500 ms

        // Turn off the blue LED
        mraa_gpio_write(gpio_blue, 0);
        usleep(500000);  // Sleep for 500 ms
    }

    mraa_gpio_close(gpio_red);
    mraa_gpio_close(gpio_green);
    mraa_gpio_close(gpio_blue);

    return 0;
```

}

7.0 Expected Output :-

The Expected output for KY -016 RGB Sensor Interfacing to Rugged Board .

- The KY-016 RGB sensor starts detecting the intensity of red, green, and blue light in the environment.



8.0 Real Time Applications :-

Interfacing a KY-016 RGB sensor with a rugged board for real-time applications can offer several practical uses in various industries and scenarios. Here are some potential real-time applications for this setup:

- **Smart Home and IoT Applications:**
  Integrating the sensor with IoT systems can offer various real-time functionalities:
  - **Smart Lighting:** Adjust room lighting based on the ambient color scheme.
  - **Home Automation:** Trigger actions or alerts based on detected color changes.
- **Data Processing Speed:**

Ensure the rugged board can process color data in real time without significant latency for applications that demand immediate responses.

- **Accuracy and Calibration:** Calibrate the sensor accurately to ensure reliable color detection in varying lighting conditions.
- **Adaptability:** Design the system to adapt to dynamic changes in color and environmental conditions.

➢ **Security and Surveillance:**

Using the sensor for color detection can have applications in security systems:

- **Intrusion Detection:** Detect unauthorized access by analyzing changes in detected colors in specific areas.
- **Object Recognition:** Identify objects or vehicles based on specific color characteristics in surveillance scenarios.

9.0 Referance links :-

1)https://www.phippselectronics.com/using-the-3-colour-led-module-ky-016-with-arduino/

2)https://iot-guider.com/raspberrypi/rgb-3-color-led-module-ky-016-in-raspberry-pi/

3)https://www.simplyiotsensors.com/2021/11/how-to-use-KY-016-RGB-led-with-arduino.html?m=1

4)https://arduinointro.com/articles/projects/mixing-colors-in-arduino-using-the-ks-and-ky-rgb-led-module

10.0 Conclusion :-

In conclusion, the KY-016 RGB Sensor Interfacing to a Rugged Board project has proven to be a valuable exploration into the realm of real-time color sensing in challenging environments. The successful integration of the KY-016 RGB sensor with the rugged board opens up a myriad of applications, from industrial automation to environmental monitoring. The successful connection and seamless integration of the KY-016 RGB sensor with the rugged board showcased the adaptability and compatibility of these components.The project demonstrated the capability of the system to perform real-time color sensing, providing immediate feedback on changes in the environment.