ACTIVITY 4 - Arduino 4: Analog Output - RGB

Jason Harvey Lorenzo, March 2, 2024
ENGG 122.02, Department of Electronics, Computer, and Communications Engineering
School of Science and Engineering, Ateneo de Manila University
Quezon City, 1108, Philippines
jason.lorenzo@obf.ateneo.edu

***Abstract*—Light's additive property enables the generation of all colors by varying the intensities of red, green, and blue values, its primary components. This paper presents an Arduino-based project utilizing the analogWrite() function to control the color of an RGB LED based on an analog input by varying the LED's RGB values. Each digit of the input integer prompts the LED to blink in a specific color, as outlined in the Color Table. The source code and operational details are provided, along with insights into the Arduino setup and functionality.**

***Index Terms*—Arduino UNO, RGB LED, analog input, pulse width modulation, color visualization**

I. Program Listing

Modern LED screens exploit the additive properties of light, leveraging varying intensities of red, green, and blue (RGB) LEDs to create the perception of diverse colors. This task involves Arduino's analogWrite() function to adjust the color of an RGB LED based on an analog input. Recall that the analogRead() function provides an integer range from 0 to 1023. The objective here is to prompt the RGB LED to blink a particular color corresponding to each digit of the recorded integer. The Color Table in Table I outlines the corresponding color for each digit.

TABLE I
Color Table

Value	Color
0	white
1	red
2	orange
3	yellow
4	green
5	blue
6	indigo
7	violet
8	ocean
9	raspberry

A. Source Code

```
String number = "";
int digitalVolt;

void setup() {
  for(int i=9; i<=11; i++)
    pinMode(i, OUTPUT);
  pinMode(A0, OUTPUT);
  Serial.begin(9600);
}

void setColor(int r, int g, int b){
  analogWrite(11, r);
  analogWrite(10, g);
  analogWrite(9, b);
}

void loop() {
  digitalVolt=analogRead(A0);
  Serial.print("Digital Value: ");
  Serial.println(digitalVolt);
  number = String(digitalVolt+10000);

  for(int i=1; i<=4; i++){
    switch(number[i]){
      case '0': //white
        setColor(255, 255, 255);
        break;
      case '1': //red
        setColor(255, 0, 0);
        break;
      case '2': //orange
        setColor(255, 125, 0);
        break;
      case '3': //yellow
        setColor(255, 255, 0);
        break;
      case '4': //green
        setColor(0, 255, 0);
        break;
      case '5': //blue
        setColor(0, 0, 255);
        break;
      case '6': //indigo
        setColor(63, 0, 255);
        break;
      case '7': //violet
        setColor(125, 0, 255);
        break;
      case '8': //ocean
        setColor(0, 84, 147);
        break;
      case '9': //raspberry
        setColor(255, 0, 125);
```

```
            break;
        }
        delay(1000);
        setColor(0, 0, 0);
        delay(500);
    }
}
```

B. How It Works

In the setup() function, pins 9 to 11 were configured as output pins to enable pulse width modulation (PWM), a feature essential for controlling the intensity of the RGB LEDs. Meanwhile, A0 was linked to the middle pin of a potentiometer, serving as a voltage divider for input. Additionally, the Serial monitor was initialized with a baud rate of 9600.

The setColor() function is where the analogWrite() function is specifically employed. This function adjusts the voltage output from pins 9 to 11 based on the three parameters (RGB values) provided to it.

The loop() function encapsulates the core functionality of the program. Here, the input (voltage detected in integer form) is read from input pin A0 and displayed in the Serial monitor. Subsequently, it is converted into a string. To preserve the most significant bit, 10,000 is added to the integer value (e.g., 0 in 0999). As strings are treated as arrays in C++, a for loop cycles through every character of the number string, excluding the first one (which is guaranteed to be 1 due to the addition of 10,000). Within the for loop, a switch statement evaluates each digit present in the number string. Depending on the digit, setColor() is invoked to set the corresponding color with their respective RGB values, as specified in the Color Table for the digit being processed. A 1-second delay is introduced to ensure visibility before turning off all RGB values for 0.5 seconds, after which the loop repeats.

II. Arduino Setup

As depicted in Fig. 1, the Arduino setup involves an Arduino UNO microcontroller linked to a laptop via a USB cord for power supply and program upload. The potentiometer's left pin is connected to the 5V output, while its right pin is linked to GND. The central pin of the potentiometer is connected to pin A0. Furthermore, the RGB LED's R pin is connected to pin 11, G to pin 10, B to pin 9, and GND to GND. In Fig. 1, the serial monitor displays the output "922", and in the specific scenario captured, the RGB LED emits a raspberry color, indicating the presence of the digit "9".

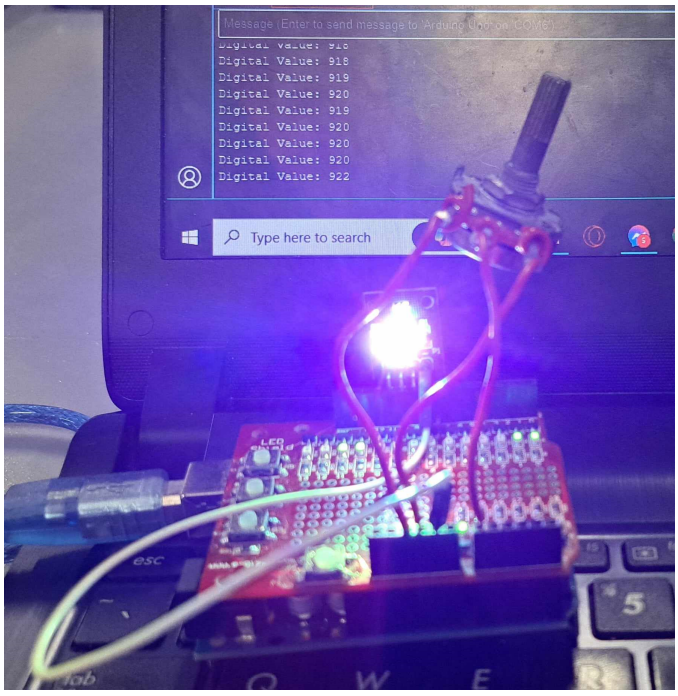

Fig. 1. Arduino Setup with RGB LED, Potentiometer, and the Serial Monitor in the background