RGB COLORS

What is our GOAL for this CLASS?

In this class, we learned the concept of **RGB LED** and fading of LED's using **Pulse width Modulation, Resolution techniques**.

What did we ACHIEVE in the class TODAY?

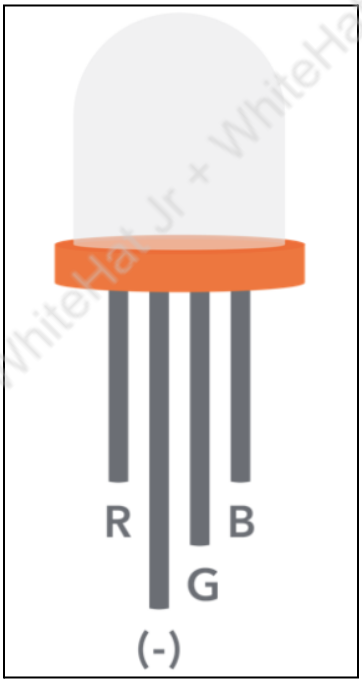
- We learned about the **RGB LED.**
- We learned about **Pulse width Modulation**
- We learned about **Resolution**
- We learned about **Fading of LED**

Which CONCEPTS/ CODING BLOCKS did we cover today?

- We learned concept of **Pulse width Modulation**
- Common Cathode/Anode function

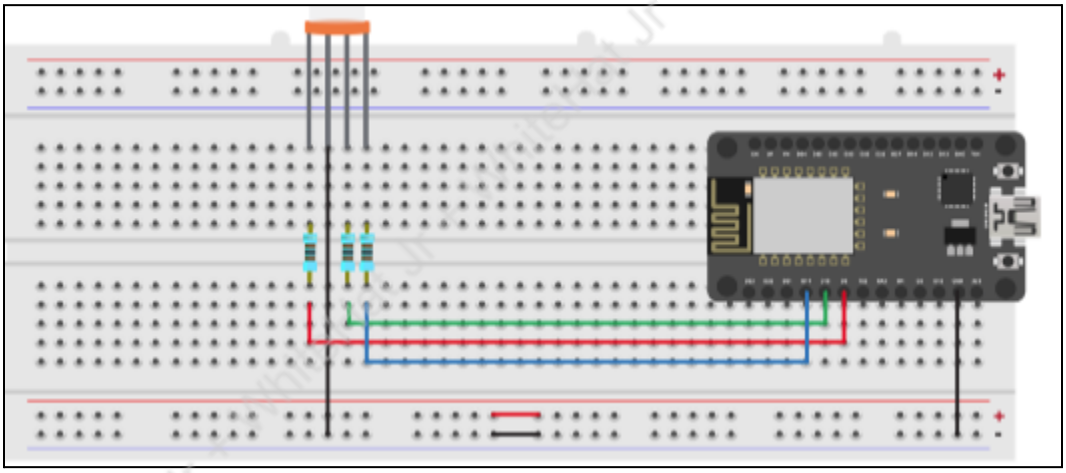
How did we DO the activities?

1. **RGB LED**: **RGB LEDs** have **four** pins—one for each **LED(Red, Green, Blue)** and another for the common anode or cathode.
    - R (red) pin: is to control the red color element
    - G (green) pin: is to control the green color element
    - B (blue) pin: is to control the blue color elements
    - Common Anode and Common Cathode RGB LEDs
    - In a common cathode RGB LED, one pin in the LED shares a common negative connection (cathode).
    - In a common anode RGB LED, one pin in LED share a common positive connection (anode)



2. To build **RGB LED**:
    - Gather the material from the **IoT** kit:
        - 1 x **ESP32**
        - 1 x **USB Cable**
        - 1 x **Breadboard**
        - 4 x **Jumper wires**
        - 1 x **RGB LED**
        - 3 x 330 Ohm **Resistors**
    - Do connections:

- Insert **RGB LED** into the breadboard, insert four legs one by one into the breadboard.
- The three positive leads of the **LEDs** (one red, one green, and one blue) are each connected to the 330- ohm **resistors**.
- The other end of the **resistor** will go to the **ESP32 GPIO** pins, **D5, D18, D19** respectively.
- The common negative connection of the **RGB LED,** which is the second pin from the flat side of the **LED** package and iis also the longest of the four leads.
- This lead will be connected to **ESP 32 GND** pin



- Write the program:
  - Specify to which **GPIO** or **GPIOs** the signal will appear upon.
    - Define **GPIO** signal pins: **LEDR** pins **5, 18, 19**
    - Define **R_channel, G_channel, B_channel 0,1,2**
    - Define **pwm_Frequency 5000**.
    - **PWM** stands for **Pulse Width Modulation.** It is used to check a portion of the time the signal spends on versus the time that the signal spends off. **PWM** on-off pattern can simulate voltages in between the full Vcc of the board (e.g., 5 V/3.3 V on a and off (0 Volts) The duration of **"on time"** is called the **pulse width**.
    - For an **LED**, **PWM** frequency is **5000 Hz**
    - Define **pwm_resolution 8**,8-bit resolution, which means we can control the LED brightness using a value from 0 to 255.

```
#define LEDR 5
#define LEDG 18
#define LEDB 19

#define R_channel 0
#define G_channel 1
#define B_channel 2

#define pwm_Frequency 5000 // pwm frequency
#define pwm_resolution 8 // 8 bit resolution
```

- Initialize using **void setup()** function
    - **ledcAttachPin()** function accepts two arguments.
    - The **first** is the **GPIO** that will output the signal, and the **second** is the channel that will generate the signal.
    - Set channel for all three colors and their PWM frequency along with **pwm_resolution**.

```
void setup() {

  ledcAttachPin(LEDR, R_channel);
  ledcAttachPin(LEDG, G_channel);
  ledcAttachPin(LEDB, B_channel);

  ledcSetup(R_channel, pwm_Frequency, pwm_resolution);
  ledcSetup(G_channel, pwm_Frequency, pwm_resolution);
  ledcSetup(B_channel, pwm_Frequency, pwm_resolution);
}
```

- To execute the main process write the **void loop():**
    - **RGB (Red, Green, Blue)** all colors have 8-bit each. The range for each individual color is 0-255. The combination range is 256*256*256.To display different colors we need to show variation between 0 to 255.
        - **RGB_Color for RED (255,0,0)**
        - **RGB_Color for Green(0.255,0)**
        - **RGB_Color for Blue (0,0,255)**
        - **RGB_Color for Yellow (255,255,0)**
        - **RGB_Color for Cyan (0, 255,255)**
        - **RGB_Color for magenta(255,0,255)**
        - **RGB_Color for pink(255,0,147)**

```
void loop() {
  RGB_Color(255, 0, 0); // RED ccolor
  delay(500);
  RGB_Color(0, 255, 0); // green color
  delay(500);
  RGB_Color(0, 0, 255); // blue color
  delay(500);
  RGB_Color(255, 255, 0); // yellow color
  delay(500);
  RGB_Color(0, 255, 255); // cyan color
  delay(500);
  RGB_Color(255, 0, 255); // magenta color
  delay(500);
  RGB_Color(255, 20, 147); // deep pink color
}
```

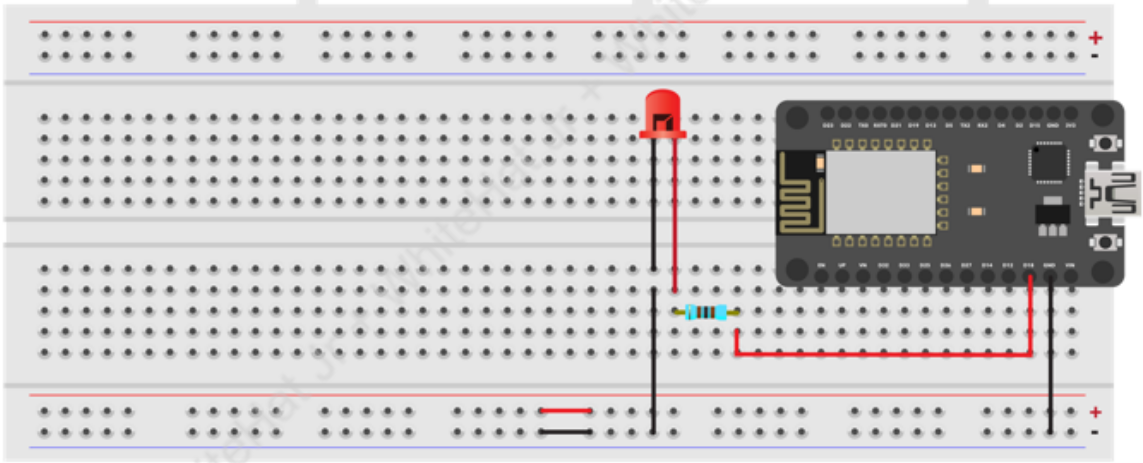
- Make a loop pattern for all described colors using loops.
- The **ledCWrite()** function is used to control the **LED** brightness using **PWM** for **R_channel, G_channel, B_channel**

```
void RGB_Color(int i, int j, int k){
  ledcWrite(R_channel, i);
  ledcWrite(G_channel, j);
  ledcWrite(B_channel, k);
}
```

- Output:
    - Compile and upload the program to **ESP32** board using **Arduino IDE**
    - Verify the program by clicking the **Tick** option
    - Upload the program by clicking the **arrow** option
    - Note: If the port is not selected, insert the USB cable in Computer's port and select the port
    - You will see different colors on the **LED**

3. To Fade the **LED**: Write the program and control the intensity and brightness of a **LED**
    - Gather the material from the **IoT** kit:
        - 1 x **ESP32**
        - 1 x **USB Cable**
        - 1 x **Breadboard**
        - 4 x **Jumper wires**
        - 1 x **RGB LED**
        - 1 x **resistor**

- Do connections:
    - Supply **positive(VCC (+ve))** from the **ESP 32** to the breadboard terminal.
    - Supply **negative(GND (-ve))** from the **ESP 32** to breadboard negative terminal
    - Insert the **Led** into the breadboard
    - Connect the shorter leg of the **LED** to one end of the **resistor** as shown below.
    - Connect another end of the **resistor** with the **GND(0V)** supply of the breadboard.
    - Connect longer Leg of the breadboard with the **ESP 32 GPIO pin 32**



- Write a program:

    - Define **Pins:** Define **GPIO pin for LED**, **led_gpio =32**
    - Define **brightness =0**
    - Define **fadeAmount =5**

```
const byte led_gpio = 32;
int brightness = 0;
int fadeAmount = 5;
```

- Initialize the **setup()**
  - **ledcAttachPin()** function accepts two arguments. The first is the **GPIO** that will output the signal, and the second is the channel that will generate the signal.
  - Set **channel =0**, **frequency=4000** and **resolution = 8**

```
void setup() {
  ledcAttachPin(led_gpio, 0); // assign a led pins to a channel


  ledcSetup(0, 4000, 8); // 12 kHz PWM, 8-bit resolution
}
```

- To execute the main process write the **void loop():**
  We need to control the **LED** brightness in this **main()** function
  - The **ledCWrite()** function is used to control the **LED** brightness, it will take two arguments **channels** and **brightness**
  - Declare the variable **brightness** and change the **brightness** level using **brightness** and **fadeAmount**
  - Write the condition using **if** condition, if **brightness** is less than **0** and more than **255**, then decrease the **fadeAmount** value.
  - Set up **delay** for **30 ms**

```
void loop() {
  ledcWrite(0, brightness); // set the brightness of the LED

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

- **Output**: Compile and upload the program to the **ESP32** board using **Arduino IDE.**
    - Verify the program by clicking the **Tick** option.
    - Upload the program by clicking the **arrow** option
    - Go to **Tools** and select **Serial Monitor**
    - See the **brightness** of the **LED** and **Value**

We learned about **RGB LEDs** and how to fade an **LED**.

What's NEXT?

In the next class, we will learn about ESP32 Server

Expand Your Knowledge

To know more about **LEDs** click here