Project 1: RGB LED Cycler Solution

Braidan Duffy*

May 24, 2022

Overview

Students will have considered completing this project when they accomplish the following tasks:

1. integrated an RGB LED and button input with their Arduino microcontroller
2. implemented at least 3 functions within their code
   at least one of these functions must include a loop
3. used a switch/case statement to switch between different functions
4. uploaded a compressed file containing:
   a video of the project running with narration
   a neatly made and well-organized schematic
   a neatly organized Arduino code file

Grading

Category	**No Credit**	**Half Credit**	**Full Credit**
Efficacy	Student did not demonstrate their working project	student demonstrated a working project, but it did not meet all of the requirements specified in the project handout	student demonstrated a feature-complete project that accomplished all of the goals specified in the project handout
3 or more functions	Student did not implement enough unique functions		Student implemented enough functions
1 or more loops	Student did not implement enough loops		Student implemented enough loops
Schematic neatness	Student did not provide a schematic, or it is illegible	Student provided a schematic, but it is difficult to read or understand	Student provided a schematic that is easy to read and understand
Mystery extra credit	Student did not request extra credit or reach the necessary thresholds	Stduent successfully implemented either the debounced button input or unique RGB function or something equivalent and justified it adequately	Student successfully implemented both debounced button input or unique RGB function and/or something equivalent and justified it adequately

*B.S. Ocean Engineering 2021
M.S. Ocean Engineering 2023

Guide

Wiring the Breadboard

Students should begin the project by plugging in the RGB LED and button to the breadboard, and the GND pin of the Arduino to the negative (blue) rail of the breadboard; the Arduino's 5V pin should be connected to the positive (red) rail. They should then atttached 1k-ohm resistors to the three short pins of the LED and jump them across the middle of the breadboard, ensuring the two leads are *not* within the same row and the resistor leads are *not* touching. Each of the three resistors should then be connected to the Arduino according to the pinout in Figure 1. The longer pin (cathode) of the LED should be connected to the negative rail of the breadboard using a jumper wire.

The students should be using a 10k-ohm pull-down resistor between one lead of the button and the breadboard's negative rail. A jumper should then connect the button lead that is *on the same column* to the Arduino pin specified in Figure 1 Here, students have the opportunity to earn extra credit by implementing a low-pass DC filter to the button output using some additional capacitors and resistors present in their Arduino kits. The successful implementation of this filter would count as partical extra credit.

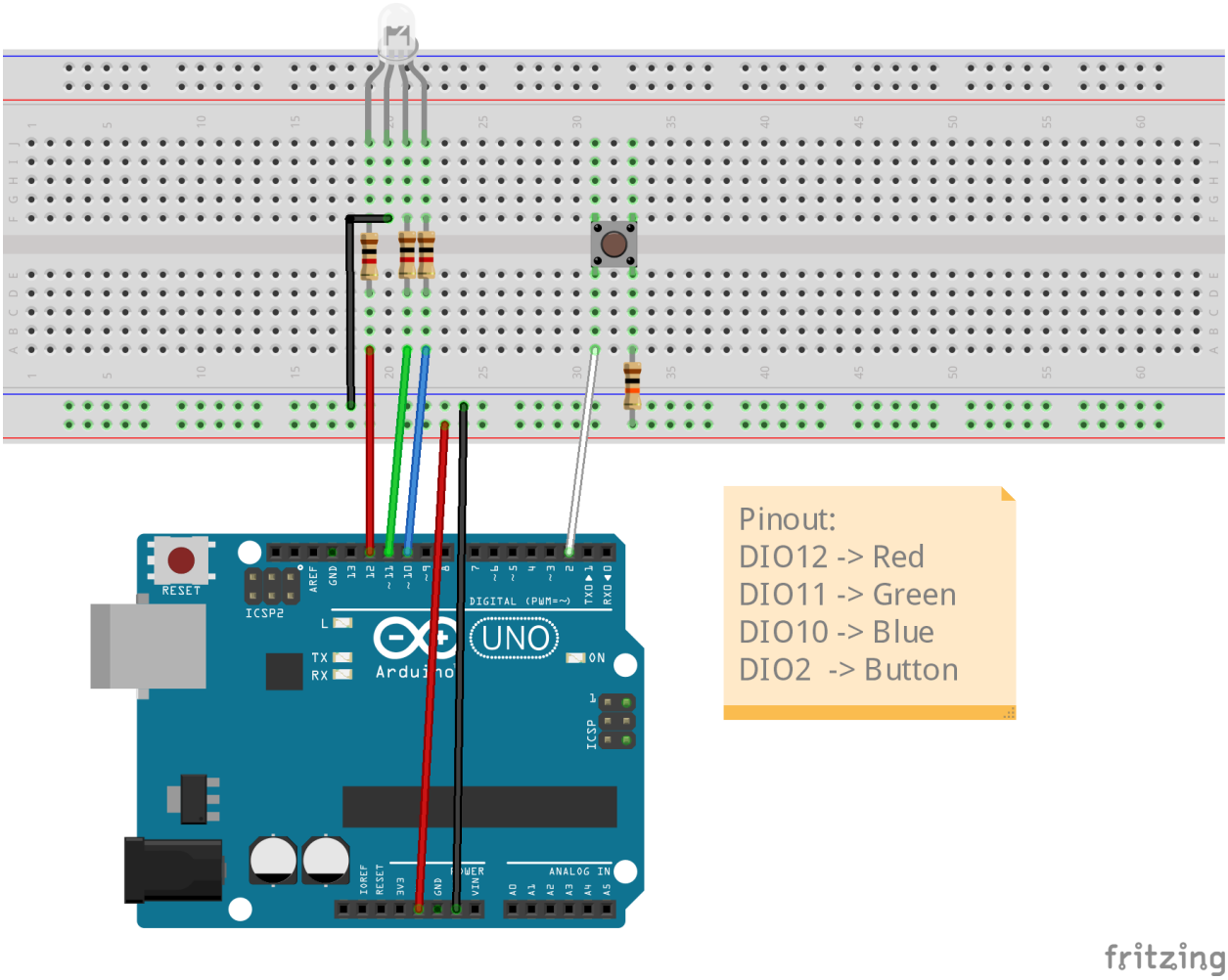

Figure 1: The breadboard layout expected for this project

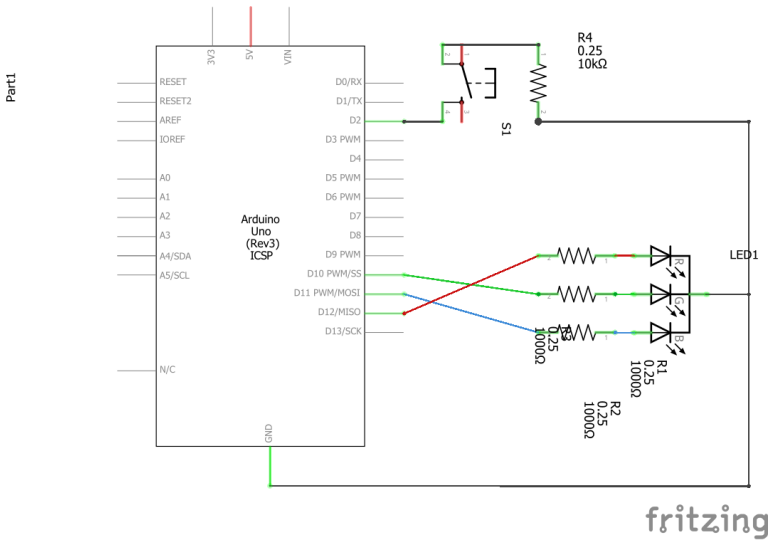

Figure 2: The schematic layout expected for this project

Arduino Code

The student should have a well-organized and commented Arduino code for their project. The code should have multiple functions including `void setup()` and `void loop()` along with their own LED implementations. At least one of the LED functions must include a loop of somekind - either a for-loop or a while-loop that controls the LED's behavior. Inside `setup()`, the student should properly initialize the Arduino pins for inputs and outputs, with appropriately named variables for the pins. Inside `loop()`, the student should have a switch-case statement that checks the LED mode and executes a different LED function for each mode. The same type of function can be called between different modes *so long as they do different things*. For example, LED mode 1 and 2 can both be `breathe()` so long as they do different colors. However, the student must have implemented at least three unique functions in their code. The easiest ones will most likely be: `staticColor()`, `breathe()`, and `rainbow()` - or some variation of that. It is up to the students to determine what they want to implement and how.

The code below shows a basic solution to this project with four unique functions. At the beginning, the pins are defined with clear variable names. The `#define` command is a pre-compiler instruction that essentially maps all calls of the specific variable name to the value. This is handy for declaring constant values in the program, such as pin numbers.

In `setup()`, the Serial port is opened for debugging purposes. Then, the pins are suitably initialized as outputs and inputs. The button input pin is specified as a `INPUT_PULLUP` so that the microcontroller internally pulls the pin high instead of floating. When the button is pressed, the stronger pulldown resistor will for the pin to go LOW, creating a distinct signal that can be used for edge detection or easier button press detection. Here, students can implement an interrupt-based button debouncer and use that to switch the LED modes. If done properly, this would count as part of the extra credit.

In `loop()`, the button input is first checked by determining if the button input pin is being pulled LOW by being pressed. If the button is pressed, then the LED mode is checked to see if it is the last mode specified by `MAX_LED_FUNCS`. If it is the last mode, the counter is reset otherwise, it is incremented. A switch-case statement checks the `ledMode` value and executes a specific function based on the value. The `default` case is present as an implicit fifth LED mode to do nothing and implicitly turn off the LEDs. *This can count as a unique function if implemented properly.* At the bottom of the function is a set of statements to reset the LED state to OFF between state function executions.

`breathe()` uses two for-loops to linearly increase the LED's brightness from OFF to FULL to OFF again. The boolean parameters in the function call determine which LEDs are lit during this function allowing users to breathe either Red, Green, Blue, or White, as they desire. To make the effect more apparent, a `delay()` call is at the bottom of the function to slow the code execution down.

`breatheSine()` uses the same principle as `breathe()` but uses a sine wave to accomplish the effect. By incrementing from 0-180 degrees, the light is brightened and dimmed in a sinusoidal fashion. As before, the boolean parameters in the function call determine which LEDs are lit, allowing for different color combinations.

`rainbow()` builds upon the previous function by cycling all three LEDs in a sinusoidal fashion, but 120 degrees out of phase and shift so the wave amplitudes lie between 0 and 2. This creates a three-phase system that will cycle through most of the color combinations, except black and white. When running, three colored LEDs should clearly get brighter and darker and appear to be "chasing" each other around the diode.

Finally, `staticColor()` just brightens an LED to a constant color and intensity specified in the function arguments. There is a delay at the end of the function to slow down the program execution and allow the program to better detect a button press and increment the LED mode.

```
/**
 * OCE4531 Project 1: RGB LED Cycler Solution
 * @brief This is the solution for OCE4531's first project, the RGB LED cycler.
 * This should not be considered the absolute correct answer that all of the students must have,
 * but it is the ball park they should be within.
 *
 * @author Braidan Duffy
 *
 * @date May 24, 2022
 *
 * @version v1.0
 */

#define BUTTON_PIN 2
#define LED_RED_PIN 12
#define LED_GREEN_PIN 11
#define LED_BLUE_PIN 10
#define MAX_NUM_FUNCS 5
#define PI 3.1415

uint8_t ledMode = 0; // Initialize LED mode state

void setup() {
    Serial.begin(115200);
    while(!Serial); // Wait for serial connection - DEBUG

    pinMode(BUTTON_PIN, INPUT_PULLUP);
    pinMode(LED_RED_PIN, OUTPUT);
    pinMode(LED_GREEN_PIN, OUTPUT);
    pinMode(LED_BLUE_PIN, OUTPUT);
}

void loop() {
    bool isButtonPressed = !digitalRead(BUTTON_PIN);
    if (isButtonPressed) {
        if (ledMode == MAX_NUM_FUNCS-1)
            ledMode = 0;
        else
            ledMode++;
        Serial.println(ledMode); //DEBUG
    }

    switch(ledMode) {
        case 0:
            breathe(true, false, false); // Breathe RED
            break;
        case 1:
            breatheSine(true, false, true); // Breath RED and BLUE
            break;
        case 2:
            rainbow();
            break;
        case 3:
            staticColor(false, false, true, 128); // Static BLUE
            break;
        default:
            delay(250); // Do nothing
            break;
    }

    // Clear LED Colors between cycles
    digitalWrite(LED_RED_PIN, LOW);
    digitalWrite(LED_GREEN_PIN, LOW);
    digitalWrite(LED_BLUE_PIN, LOW);
}

// ==========================
```

```
// === LED MODE FUNCTIONS ===
// ==========================

void breathe(bool r, bool g, bool b) {
    for (uint8_t i=0; i<255; i++) { // Increase brightness
        if (r) analogWrite(LED_RED_PIN, i);
        if (g) analogWrite(LED_GREEN_PIN, i);
        if (b) analogWrite(LED_BLUE_PIN, i);

        delay(5);
    }

    for (uint8_t j=255; j>0; j--) {
        if (r) analogWrite(LED_RED_PIN, j);
        if (g) analogWrite(LED_GREEN_PIN, j);
        if (b) analogWrite(LED_BLUE_PIN, j);

        delay(5);
    }
}

void breatheSine(bool r, bool g, bool b) {
    for (int i=0; i<180; i++) {
        float iRad = i * PI / 180;
        float val = sin(iRad) * 255;

        if (r) analogWrite(LED_RED_PIN, val);
        if (g) analogWrite(LED_GREEN_PIN, val);
        if (b) analogWrite(LED_BLUE_PIN, val);

        delay(5);
    }
}

void rainbow() {
    for (int i=0; i<360; i++) {
        float iRad = i * PI / 180;
        float rVal = (sin(iRad)+1) * 128;
        float gVal = (sin(iRad + 2*PI/3) + 1) * 128;
        float bVal = (sin(iRad + 4*PI/3)+1) * 128;

        analogWrite(LED_RED_PIN, rVal);
        analogWrite(LED_GREEN_PIN, gVal);
        analogWrite(LED_BLUE_PIN, bVal);

        delay(5);
    }
}

void staticColor(bool r, bool g, bool b, uint8_t i) {
    if (r) analogWrite(LED_RED_PIN, i);
    if (g) analogWrite(LED_GREEN_PIN, i);
    if (b) analogWrite(LED_BLUE_PIN, i);

    delay(250);
}
```