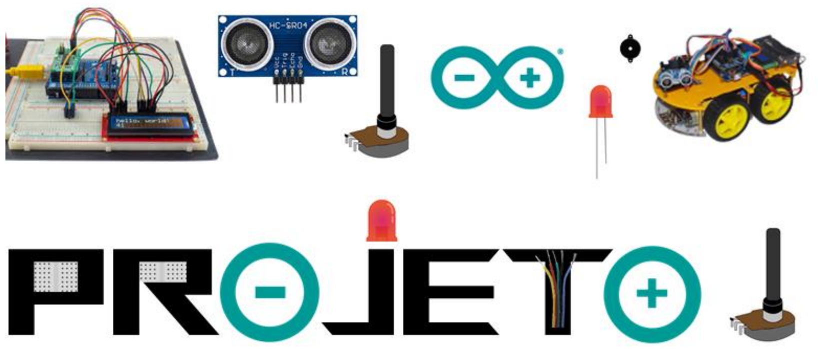

Licenciatura em Engenharia Eletrotécnica e de Computadores

Laboratorial assignment # 2

Getting Started with Arduino: Intelligent LED Effects

Components list:

- Arduino UNO
- 1 USB cable
- 1 white breadboard
- 6 LEDs (any color)
- 6 220Ω resistors
- 1 Potentiometer
- 1 RGB LED (with 4 pins: generates red, green or blue colors)
- Wires
- Software: Arduino IDE

What is Arduino?

Arduino is an open-source electronics platform based on a user-friendly approach for developing hardware- and software-based systems. It is ideal for anyone making interactive projects. It has a very easy and simple way of programming in a C-like based language style with friendly software and hardware (Arduino IDE).

1. Blink an external LED

Plug your Arduino UNO board to your PC using the USB cable.
Mount the system as shown in **Figure 1**. Please be extremely careful with the **polarity** of the LED.

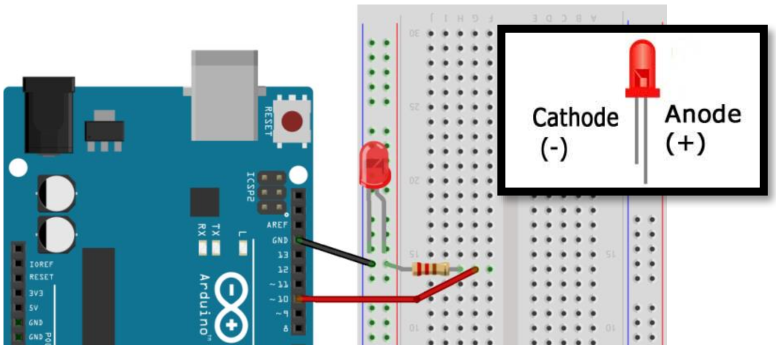

Figure 1. Circuit for blinking an external LED.

Upload the following program to the UNO and observe the LED's behavior.

```
//the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 10 as an output.
  pinMode(10, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(10, HIGH);         // turn the LED on (HIGH)
  delay(1000);                    // wait for a second
  digitalWrite(10, LOW);          // turn the LED off (LOW)
  delay(1000);                   // wait for a second
}
```

1.1. Duty Cycles and LED blinking

a) Change the duty cycle of the blinking to 90%, 45% and 22%.
   Explain the differences and show them to your teacher.
   Note: the duty cycle is the percentage of the period for which the circuit is HIGH.

b) Using the same duty cycles, try shorter periods (e.g., 4miliseconds instead of 2 seconds).
   Is it still blinking? Can you explain the new effect that you observe?

```
Examples:                                              In code:

1) Duty cycle of 80%
       LED HIGH     1600 milliseconds                  digitalWrite(10, HIGH);
       LED LOW      400 milliseconds                   delay(1600);
                                                       digitalWrite(10, LOW);
                                                       delay(400);

2) Duty cycle of 10%
       LED HIGH     200 milliseconds                   digitalWrite(10, HIGH);
       LED LOW      1800 milliseconds                  delay(200);
                                                       digitalWrite(10, LOW);
                                                       delay(1800);
```

1.2. Blink quicker and quicker

a) Mount a circuit that starts by blinking the LED for a period of 4 seconds (duty cycle of 50%) and then, in each `loop()` call it shall divide the period by half.
b) Upload to the UNO your circuit and test it.
c) What have you observed? Can you explain why the LED seems to have turned ON permanently?

```
//Global variables
int time;

void setup() {
  // initialize digital pin 10 as an output.
  pinMode(10, OUTPUT);
  time=2000;
}

void loop() {
  digitalWrite(10, HIGH);         // turn the LED on (HIGH)
  delay(time);                    // wait for half a period
  digitalWrite(10, LOW);          // turn the LED off (LOW)
  delay(time);                    // wait for half a period
  time=time/2;                    // half of period for the next time
}
```

1.3. Controlling LED intensity with analogWrite()

This time we will output an 8-bit analog value (between 0 and 255) to control the LED's lightening intensity through one of the PWM pins of the Arduino Uno by using the `analogWrite()` function.

Based on the previous exercise, define an **intensity** global variable and initialize it at 255 in the **setup** function. Decrement the intensity in your **loop** function and use `analogWrite(3, intensity)` to control the LED intensity with pin 3. Use a delay of 10 miliseconds. When the intensity value reaches 0, reset it to 255. Upload and test your program.

2. LED String and Chase effect

2.1. Array of LEDs with Different Intensity

Observe **Figure 2** and mount a string of 6 LEDs in the board, by connecting all negative poles of the LEDs (shorter wire) to the GND (0V) and the positive poles (longer one) to a 220Ω resistor and then to the 6 digital inputs.

Since we will be using the `analogWrite()` function, you will need to use the 6 PWM pins (3, 5, 6, 9, 10 and 11) instead of those in the figure. Now set the first LED on the right with an intensity of 64, and each consecutive LED with half the intensity of the previous one. Upload and test your program.

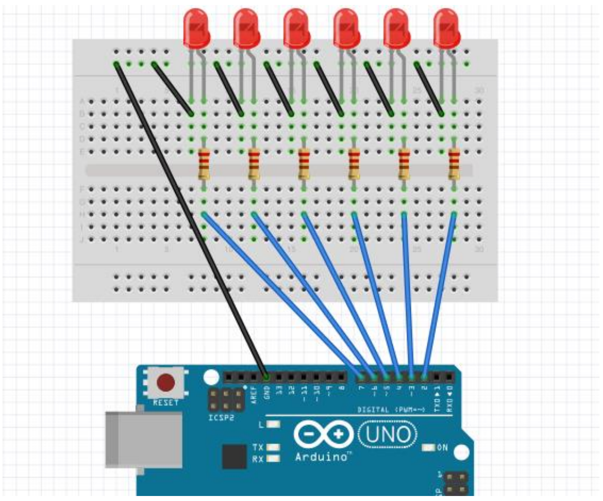

Figure 2. LED string circuit.
Use PWM pins (3,5,6,9,10,11) instead.

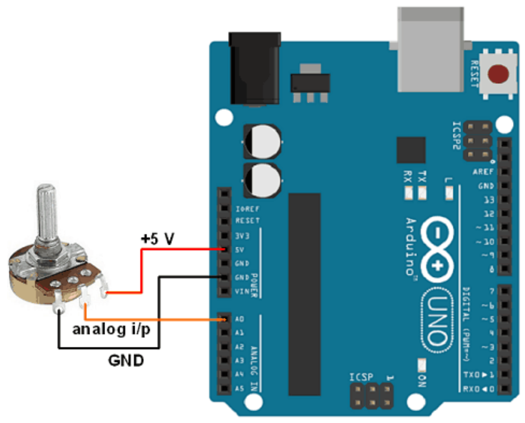

Figure 3. Potentiometer circuit.

2.2. Interactive Intensity effect

A potentiometer consists of an adjustable resistor whose value varies as function of a rotating button from 0Ω to its maximum (in the case of this work 10 kΩ). Plug the potentiometer as shown in **Figure 3** by choosing the analog pin you want to read from, such as A0.

The analog pins of the Arduino UNO can read a voltage value between 0V and +5V and translate it to a 10-bit digital value between 0 and 1023, using the `analogRead()` function.

Please check the builtin `AnalogReadSerial` example in the Arduino IDE (File > Examples > Basics).

Write a program that reads the potentiometer value (`sensorValue` between 0 to 1023) and interactively changes the intensity of the first LED on the right by `sensorValue/4`. All other LEDs intensity should automatically change according to the previous exercise.

Upload and test your program.

2.3. Interactive LED Chase effect

The chase effect is known for being the effect of lighting a single LED in a cyclic way, creating the illusion of the light moving along the string of LEDs. The light "moves" in one direction until the end of the string and then starts from that position and moves back in the opposite direction. This behavior is repeated indefinitely. Write a program to read the potentiometer value and produce a LED chase effect where the delay between each LED transition (in milliseconds) corresponds to the sensor value read.

Double the intensity values of the LEDs every time you complete a cycle (back and forth). Start with intensity values of 1 and reset when you reach maximum intensity.

Upload and test your program.

3. RGB LED

A color can be generated from the combination of other base colors. This is the basic principle used in monitors, tablets and other electronic equipment that displays images to the user. In this work, we will exploit and demonstrate this property using an RGB (red, green and blue) LED, by mixing the 3 primary colors to produce any color we wish. **Table 1** shows the different colors that you can achieve when turning on and off certain combinations of colors (for example, green and blue produce cyan).

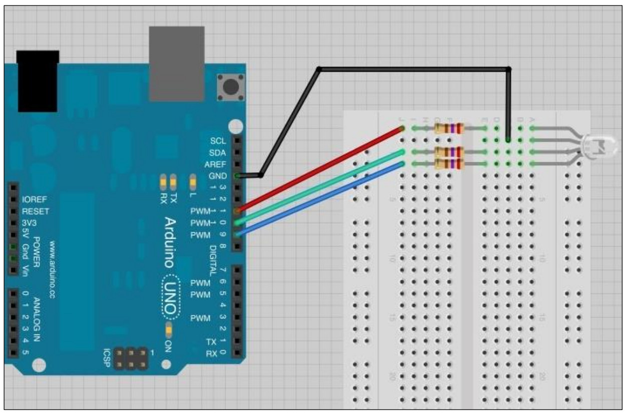

Figure 4. An RGB LED has 3 pins for generating red, green or blue lights, or the combined effect of them. The common pin 2 (the longest pin) should be connected to ground (GND).

Red	Green	Blue	Color
ON	OFF	OFF	Red
OFF	ON	OFF	Green
OFF	OFF	ON	Blue
ON	ON	OFF	Yellow
OFF	ON	ON	Cyan
ON	OFF	ON	Magenta
ON	ON	ON	White

Table 1. How to combine colors using the digitalWrite() function.

Note: the break in the middle of the breadboard indicates that there is an <u>electrical interruption</u> between lines in the upper and lower parts of the board, and also in the two top and bottom lines; please have this in mind when mounting your circuit.

3.1. RGB LED colors with digitalWrite()

Observe the circuit in **Figure 4**. Implement it and write a program that is able to control the lights in an RGB LED using the `digitalWrite()` function. The operating mode should be as follows:

- Each primary color (red, green and blue) should blink for 3 seconds (½ second cycle; duty cycle 50%).
- Then, secondary colors (yellow, cyan and magenta) should start lightening for periods of 2 seconds.
- Finally, you should lighten and blink all the 3 colors simultaneously indefinitely (½ second cycle; duty cycle 50%). What's the obtained color? Why?

Upload and test your program.

When reaching the final state of the program you can use the Arduino's <u>reset button to start again</u>.

Note: the light generated by the LED can be quite intensive. Please use a small paper tape around the LED to decrease the light's intensity. Make sure you have tested all the possible combinations in Table 1.

3.2. Mood Lamp Effect with analogWrite()

You can now change your program so that the RGB LED may cycle through all rainbow colors (as depicted in the example of **Figure 5**), turning each primary and secondary color on during 0.5 seconds. Once a rainbow cycle is completed, the LED should start blinking with random colors for periods of 2 seconds each.

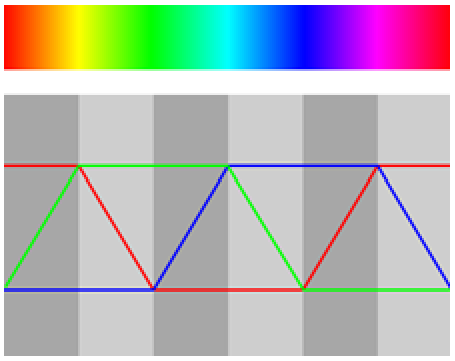

Figure 5. RGB combinations for generating the 7 different rainbow colors in Table 1.
Note the order: Red, Yellow, Green, Cyan, Blue, Magenta and back to Red.

Red	Green	Blue	Color
255	0	0	Red
0	255	0	Green
0	0	255	Blue
255	255	0	Yellow
0	255	255	Cyan
255	0	255	Magenta
255	255	255	White

Table 2. How to combine colors using the analogWrite() function.

Instead of what we did in the previous work using the `digitalWrite()` function, this time we will output an analog value between 0 and 255 (see Table 2) to control the LED's lightening intensity by using the `analogWrite()` command. Make sure you are using PWM pins 9, 10 and 11. Note that the transitions between colors should be smooth, e.g. using a for/while cycle. Upload and test you program.

3.3. Mood Lamp Effect with a Potentiometer

Change the previous exercise so that all timings (including the rainbow cycle and interval for turning each color on) are interactively controlled by a value read by a potentiometer.

Use 100 miliseconds as minimum time unit (corresponding to a potentiometer value of 0) and 2146 miliseconds as maximum time unit (corresponding to a potentiometer value of 1023) for the duration of the primary and secondary colors.