ELECDUIN

O-UN

WORKSHOP

• Project 2 : Create A Blinking LED Wave

In this project, we'll use FOUR LEDs and will switch them on/off one by one creating a kind of wavelike light pattern.

• *What we do:*

1. Turn on LED 1
2. Wait for 100 milliseconds
3. Turn off LED 1
4. Turn on LED 2
5. Wait for 100 milliseconds
6. Turn off LED 2
7. Turn on LED 3
8. Wait for 100 milliseconds
9. Turn off LED 3
10. Turn on LED 4
11. Wait for 100 milliseconds
12. Repeat infinitely

• *What we need:*

1. Four LED of any color
2. Four 560ohm Resistors
3. Breadboard
4. Arduino and USB Cable
5. Connecting Wires

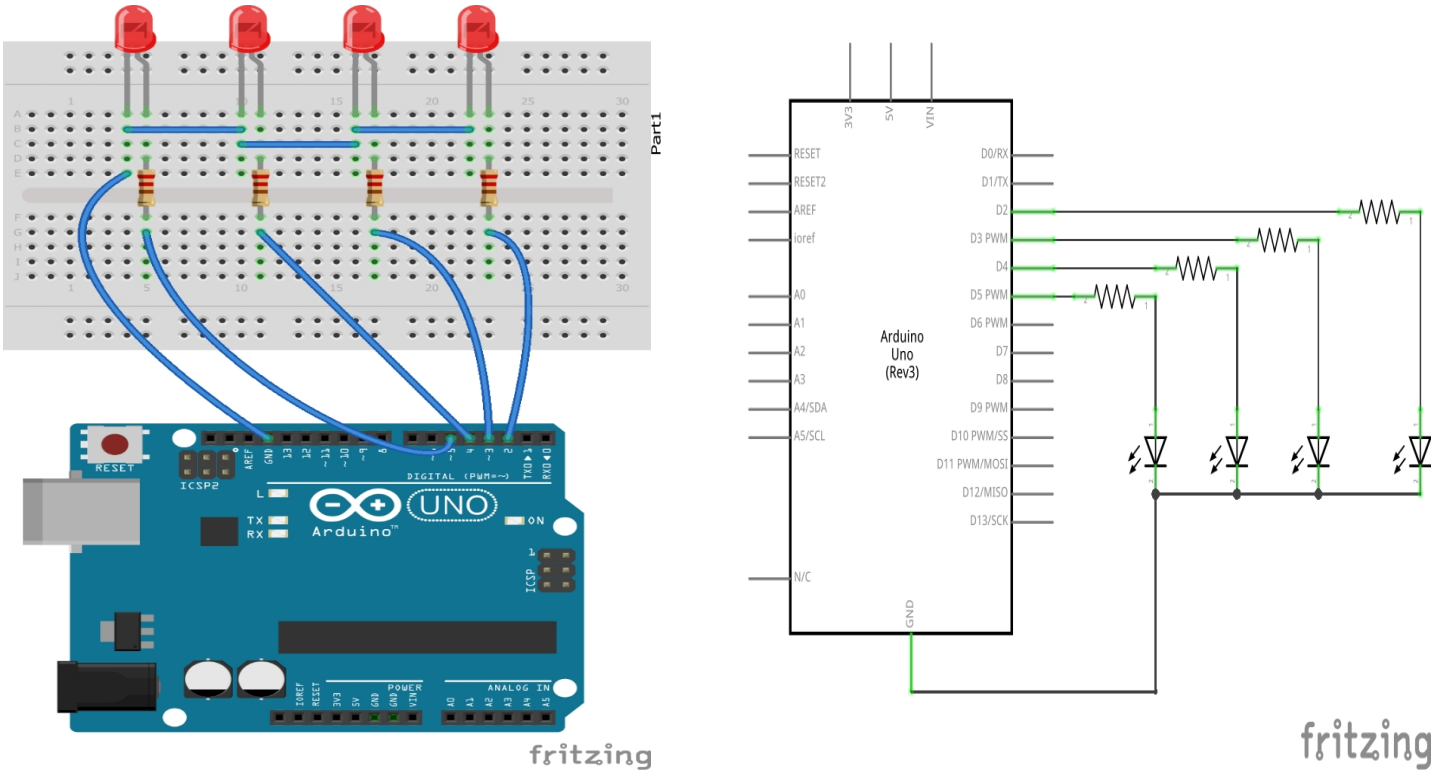
• *The Code:*

```
//Project: Create a blinking LED Wave
void setup()
  {
     pinMode(2, OUTPUT);         // Setting the pin 2 as output
     pinMode(3, OUTPUT);         // Setting the pin 3 as output
     pinMode(4, OUTPUT);         // Setting the pin 4 as output
     pinMode(5, OUTPUT);         // Setting the pin 5 as output
  }
void loop()
  {
    digitalWrite(2, HIGH);          // Turn on LED 1
```

```
  delay(100);                    //  Wait for 100 milliseconds
  digitalWrite(2, LOW);          //  Turn off LED 1
  digitalWrite(3, HIGH);         //  Turn on LED 2
  delay(100);                    //  Wait for 100 milliseconds
  digitalWrite(3, LOW);          //  Turn off LED 2
  digitalWrite(4, HIGH);         //  Turn on LED 3
  delay(100);                    //  Wait for 100 milliseconds
  digitalWrite(4, LOW);          //  Turn off LED 3
  digitalWrite(5, HIGH);         //  Turn on LED 4
  delay(100);                    //  Wait for 100 milliseconds
  digitalWrite(5, LOW);          //  Turn off LED 4
}
```

In above code in void setup() function, the digital I/O pins are set to outputs, because we want them to send current to the LEDs. We specify when to turn on each LED using the digitalWrite() function in the void loop() while the delay() function is used to define time.

- ***The Schematic & Breadboard Circuit:***



- ***WARNING:*** Do not connect leds without the use of resistors as the voltage provide by Arduino digital pin is 5V which is maximum than the optimal operating voltage of leds.

• Project 3 : Repeating With Functions and For Loops

When writing the code of previous project, you may notice that we repeat some line of codes again and again. We could simply copy and paste these lines of code to duplicate it in a sketch, but that's inefficient and a waste of our Arduino's program memory. Instead, we can use for loops. The benefit of using a for loop is that we can determine how many times the code inside the loop will repeat. To see how a for loop works, lets create another project having the same function as the previous one but using for loops:

• *What we do:*

13. Turn on LED 1
14. Wait for 100 milliseconds
15. Turn off LED 1
16. Turn on LED 2
17. Wait for 100 milliseconds
18. Turn off LED 2
19. Turn on LED 3
20. Wait for 100 milliseconds
21. Turn off LED 3
22. Turn on LED 4
23. Wait for 100 milliseconds
24. Repeat infinitely

• *What we need:*

6. Four LED of any color
7. Four 560ohm Resistors
8. Breadboard
9. Arduino and USB Cable
10. Connecting Wires

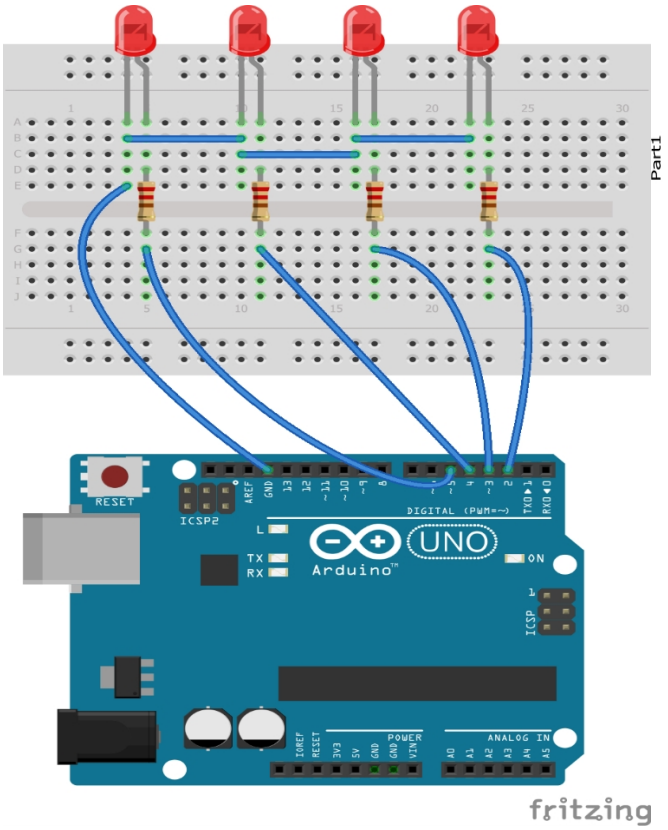
• *The Code:*

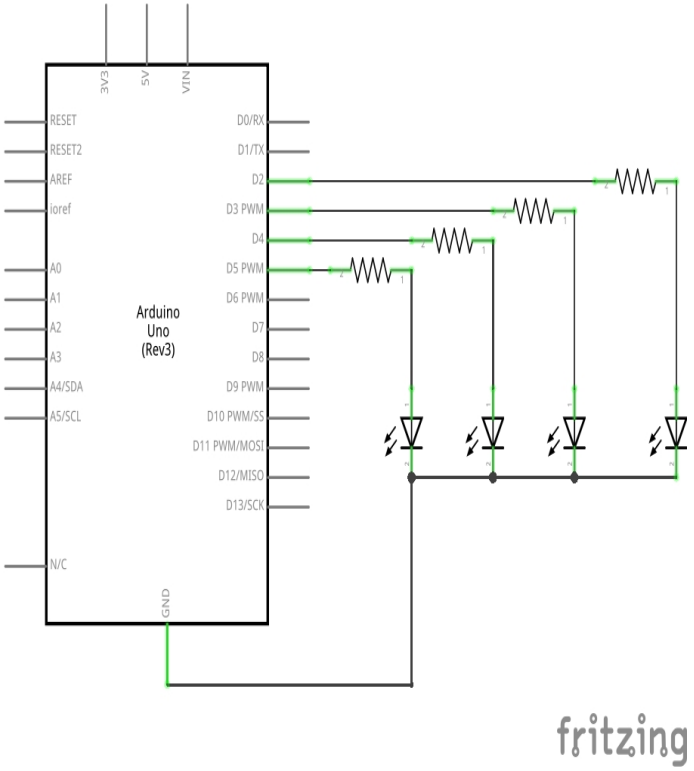
```
//Project: Create a blinking LED Wave
int d = 100;                        // Creating a integer variable d with value equal to 100
void setup()
  {
     pinMode(2, OUTPUT);          // Setting the pin 2 as output
     pinMode(3, OUTPUT);          // Setting the pin 3 as output
     pinMode(4, OUTPUT);          // Setting the pin 4 as output
```

```
            pinMode(5, OUTPUT);         // Setting the pin 5 as output
        }
void loop()
  {
     for (int i =2; i<6; i++)
       {
         digitalWrite(a, HIGH);
         delay(d);
         digitalWrite(a, LOW);
         delay(d);
       }
  }
```

The for loop will repeat the code within the curly brackets beneath it as long as some condition is true. Here, we have used a new integer variable, i, which starts with the value 2. Every time the code is executed, the i++ will add 1 to the value of i. The loop will continue in this fashion while the value of i is less than 6 (the condition). Once it is equal to or greater than 6, the Arduino moves on and continues with whatever code comes after the for loop.

- ***The Schematic & Breadboard Circuit:***

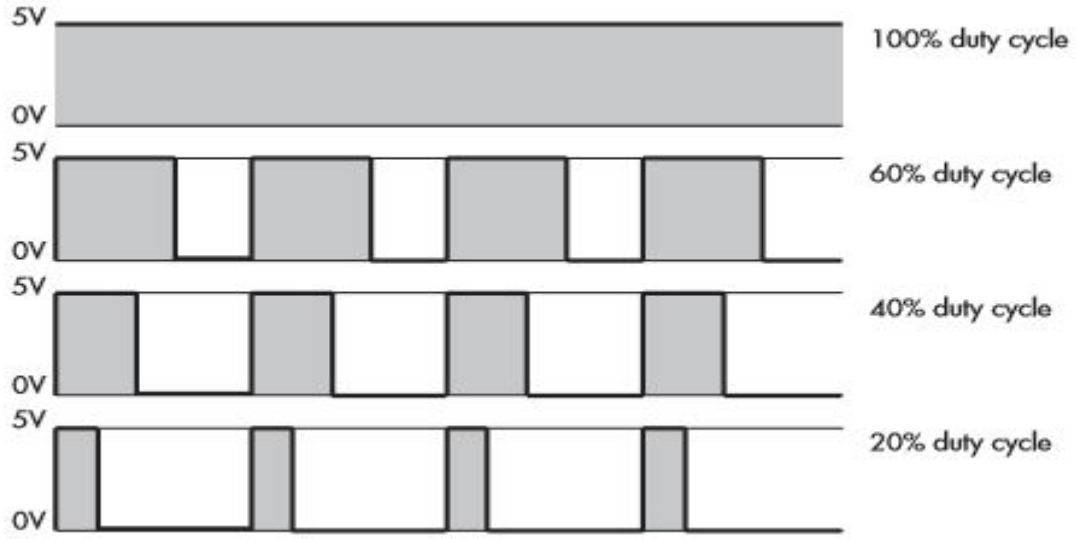




- ***NOTE:*** Do not connect leds without the use of resistors as the voltage provide by Arduino digital pin is 5V which is maximum than the optimal operating voltage of leds.

• Project 4 : Demonstrating PWM

PWM can be used to create the illusion of an LED being on at different levels of brightness by turning the LED on and off rapidly, at around 500 cycles per second. The brightness we perceive is determined by the amount of time the digital output pin is on versus the amount of time it is off—that is, every time the LED is lit or unlit. Because our eyes can't see flickers faster than 50 cycles per second, the LED appears to have a constant brightness. The greater the duty cycle (the longer the pin is on compared to off in each cycle), the greater the perceived brightness of the LED connected to the digital output pin. Below figure shows various PWM duty cycles. The filled-in gray areas represent the amount of time that the light is on. As you can see, the amount of time per cycle that the light is on increases with the duty cycle.



To create a PWM signal, we use the function analogWrite(x, y), where x is the digital pin and y is a value for the duty cycle, between 0 and 255, where 0 indicates a 0 percent duty cycle and 255 indicates 100 percent duty cycle.

• *What we do:*

Rather than just turning LEDs on and off rapidly using digitalWrite(), we can define the level of brightness of an LED by adjusting the amount of time between each LED's on and off states using pulse-width modulation

• *What we need:*

1. Arduino UNO & USB Cable
2. Connecting Wires
3. Led

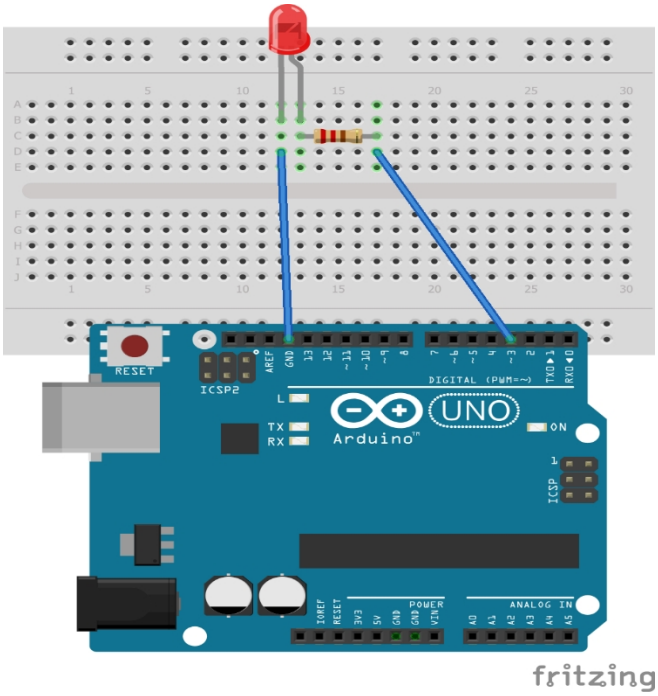
4. 560 ohm resistor

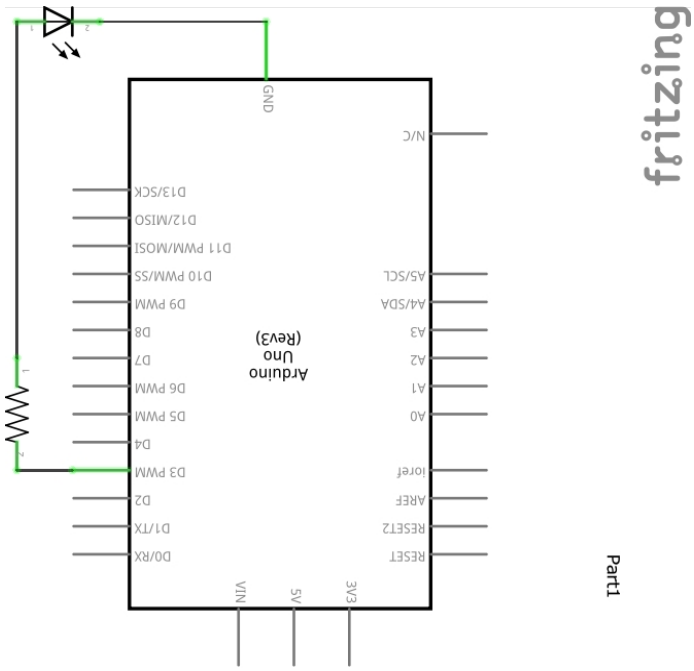
- ***The Code:***

```
// Project: Demonstrating PWM
int d = 5;
void setup()
  {
    pinMode(3, OUTPUT);    // LED control pin is 3, a PWM capable pin
  }
void loop()
 {
   for ( int a = 0 ; a < 256 ; a++ )
     {
       analogWrite(3, a);
       delay(d);
     }
   for ( int a = 255 ; a >= 0 ; a-- )
    {
       analogWrite(3, a);
       delay(d);
    }
  delay(200);
}
```

The LED on digital pin 3 will exhibit a “breathing effect” as the duty cycle increases and decreases. In other words, the LED will turn on, increasing in brightness until fully lit, and then reverse. Experiment with the sketch and circuit. For example, make five LEDs breathe at once, or have them do so sequentially (you can use the previous project for this)

- ***The Schematic & Breadboard Circuit:***





- ***Remember:*** Only digital pins 3, 5, 6, 9, 10, and 11 on a regular Arduino board can be used for PWM. They are marked on the Arduino board with a tilde (~),

• Project 5 : Digital Input With Switch

In previous projects, we used digital I/O pins as outputs to turn LEDs on and off. We can use these same pins to accept input from users—such as detecting whether a push button has been pressed by a user. Like digital outputs, digital inputs have two states: high and low. The simplest form of digital input is a push button. We can insert these directly into our solderless breadboard. A push button allows a voltage or current to pass when the button is pressed, and digital input pins are used to detect the presence of the voltage and to determine whether a button is pressed.

• *What we do:*

We will create a button that turns on an LED for half a second when pressed.

1. Test to see if the button has been pressed.
2. If the button has been pressed, then turn on the LED for half a second, and then turn it off.
3. If the button has not been pressed, then do nothing.
4. Repeat indefinitely

• *What we need:*

5. Breadboard
6. Arduino and USB Cable
7. Connecting Wires
8. One push button
9. One LED
10. One 560ohm resistor
11. One 10 kohm resistor
12. One 100 nF capacitor

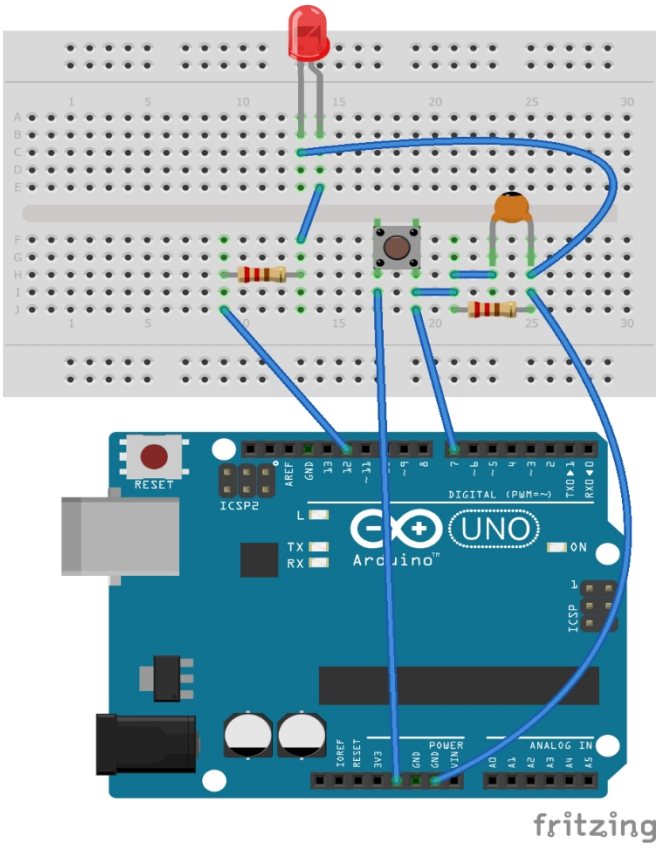
• *The Code:*

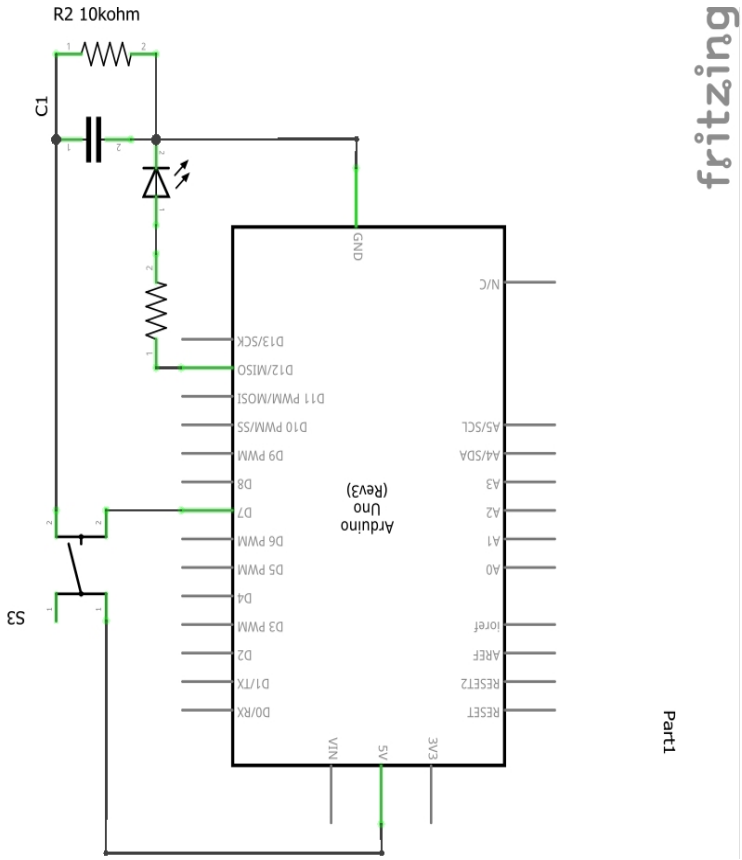
```
// Project 4 - Demonstrating a Digital Input
#define LED 12
#define BUTTON 7
void setup()
 {
   pinMode(LED, OUTPUT);          // output for the LED
   pinMode(BUTTON, INPUT);     // input for the button
 }
```

```
void loop()
 {
   if ( digitalRead(BUTTON) == HIGH )
    {
      digitalWrite(LED, HIGH);       // turn on the LED
      delay(500);                    // wait for 0.5 seconds
      digitalWrite(LED, LOW);       // turn off the LED
    }
 }
```

- ***The Schematic & Breadboard Circuit:***

Notice how the 10 kW resistor is connected between GND and digital pin seven. We call this a pull-down resistor, because it pulls the voltage at the digital pin almost to zero. Furthermore, by adding a 100 nF capacitor across the 10 kW resistor, we create a simple **debounce** circuit to help filter out the switch bounce. When the button is pressed, the digital pin goes immediately to high. But when the button is released, digital pin seven is pulled down to GND via the 10 kW resistor, and the 100 nF capacitor creates a small delay. This effectively covers up the bouncing pulses by slowing down the voltage falling to GND, thereby eliminating most of the false readings due to floating voltage and erratic button behavior.





- ***Remember:*** Read the switch Bouncing article to understand schematic and the project completely.