Lab 1
Playing with LEDs

LED: Light Emitting Diode

Objective

- Switching on an LED
- Blink an LED
- Switching on an LED with a switch
- Switching on a set of LEDs

The components

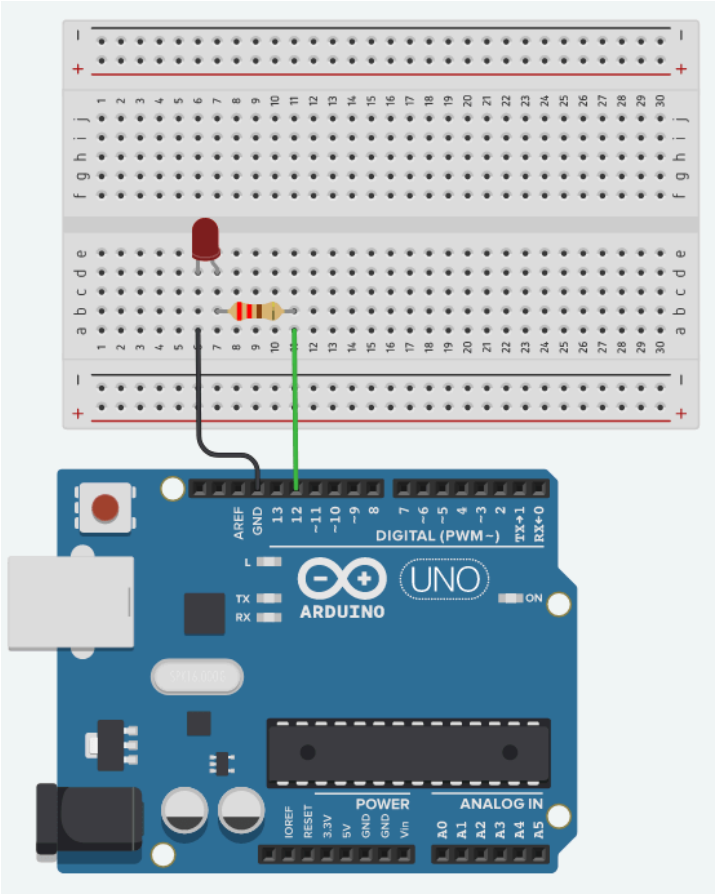
- Arduino
- Set of LEDs
- Resistances
- Switch
- Jumper wires (Male - Male)
- Breadboard

Implementation 1

In this lab, you will learn how to use an LED with Arduino. First, you will setup your circuit with an Arduino board and an LED, and then discover different ways to control the LED.

Create the Arduino LED circuit

We must create this circuit:



How to build the circuit

1. First, make sure that the Arduino is powered off (no USB cable plugged to anything).
2. Check the LED, you will see that one of the legs is shorter than the other one.
3. Plug the shorter leg of the LED into a hole in the breadboard. Connect that leg to a GND pin of the Arduino, using a black cable if possible (convention for GND).
4. Plug the longer leg of the LED to a different hole, on a different and independent line of the breadboard.
5. Add a resistor between this longer leg and a digital pin of the Arduino, using an additional colored wire (no red, no black) for convenience.

Here we choose digital pin 12 on the Arduino Uno.

Note: you could have chosen any of the digital pins ranging from 0-13, and also any of the analog pins, which you can use as digital pins. So, for the Arduino Uno, you get 20 possibilities.

Arduino C++ source code

Test using DigitalWrite with PIN 2

Let's start with the most simple thing. Let's say we just want to power on the LED when the Arduino program starts. This code will help us understand the setup and how digital pins work.

```
1.  #define LED_PIN 12
2.  void setup()
3.  {
4.    pinMode(LED_PIN, OUTPUT);
5.    digitalWrite(LED_PIN, HIGH);
6.  }
7.  void loop() {}
```

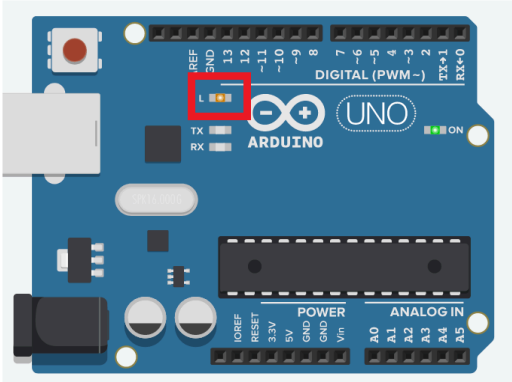
Test using only the built-in Arduino LED

If you don't want to build the circuit, or want to test with something even simpler, you can use the built-in LED on the Arduino, which is soldered on digital pin 13.

You can use LED_BUILTIN which is already predefined for this LED.

```
1.  void setup()
2.  {
3.    pinMode(LED_BUILTIN, OUTPUT);
4.    digitalWrite(LED_BUILTIN, HIGH);
5.  }
6.  void loop() {}
```

And you'll see the built-in LED powered on. Note: the location of the LED can vary depending on the type of your Arduino board.



Implementation 2

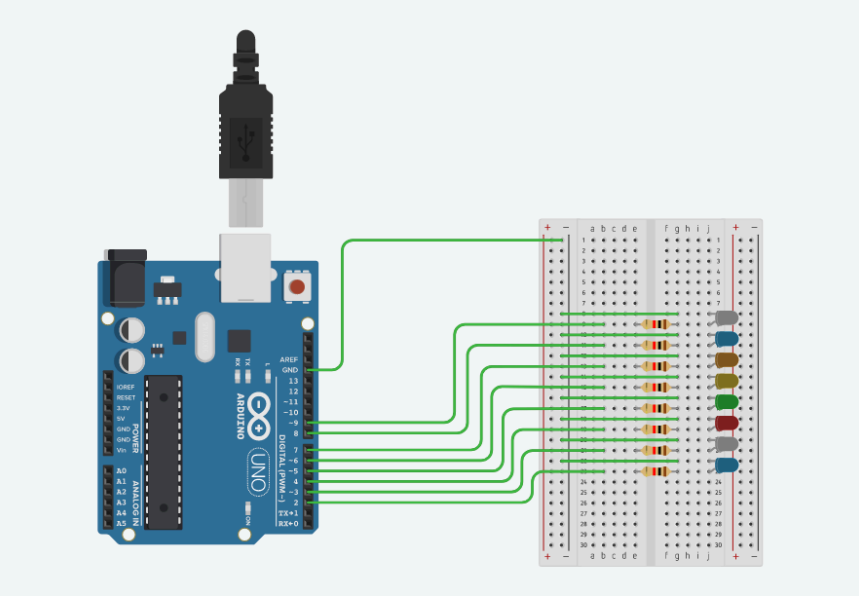
Let's move on to the stage where the LED is flashing. We're going to keep the same circuit but replace the source code with the following:

```
1. int ledpin = 12;
2. void setup() {
3.   pinMode(ledpin, OUTPUT);
4. }
5. void loop() {
6.   digitalWrite(ledpin, LOW );
7.   delay(1000);
8.   digitalWrite(ledpin, HIGH);
9.   delay(1000);
10.}
```

Implementation 3

Now we're going to connect several LEDs (8 or less) in a row and make them flash one after the other. Follow the circuit below.

After that, we'll insert the code below. (be careful to change the value of `lastPin`, which must be related to the maximum number of LEDs used.

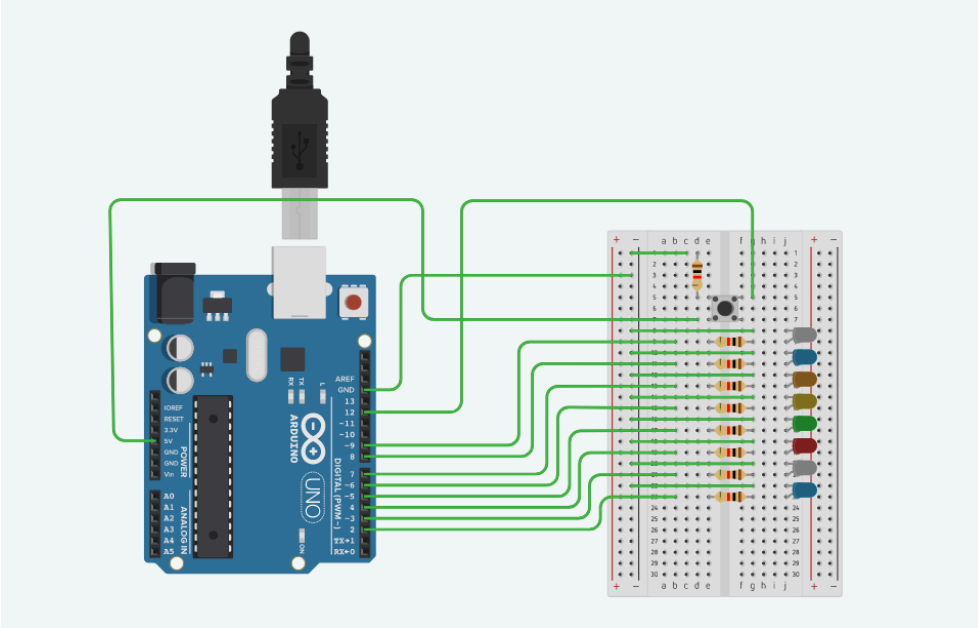


```
1.  const int firstPin = 2;
2.  const int lastPin = 9;
3.  void setup()
4.  {
5.    for(int thisPin = firstPin;thisPin <= lastPin; thisPin++)
6.    {
7.       pinMode(thisPin,OUTPUT);
8.    }
9.  }
10.  void loop()
11.  {
12.    for(int thisPin = firstPin;thisPin <= lastPin;thisPin++)
13.    {
14.       digitalWrite(thisPin,HIGH);
15.       delay(100);
16.    }
17.    for(int thisPin = lastPin;thisPin >= firstPin; thisPin--)
18.    {
19.      digitalWrite(thisPin,LOW);
20.      delay(100);
21.    }
22.    for(int thisPin = lastPin;thisPin >= firstPin; thisPin--)
23.    {
24.      digitalWrite(thisPin,HIGH);
25.      delay(100);
26.    }
```

```
27.     for(int thisPin = firstPin;thisPin <= lastPin; thisPin++)
28.     {
29.        digitalWrite(thisPin,LOW);
30.        delay(100);
31.     }
32.   }
```

Implementation 4

Update the previous circuit by adding a switch button as shown in the figure below:



We will then need to add the following source code:

```
1.    const int buttonPin = 12;
2.    const int firstPin = 2;
3.    const int lastPin = 9;
4.    void setup()
5.    {
6.      for(int thisPin = firstPin;thisPin <= lastPin; thisPin++)
7.      {
8.        pinMode(thisPin,OUTPUT);
9.      }
10.      pinMode(buttonPin,INPUT);
11.    }
12.    void loop()
```

```
13.   {
14.     if(digitalRead(buttonPin) == HIGH) {
15.       for(int thisPin = firstPin;thisPin <= lastPin;thisPin++)
16.       {
17.          digitalWrite(thisPin,HIGH);
18.          delay(100);
19.       }
20.       for(int thisPin = lastPin;thisPin >= firstPin; thisPin--)
21.       {
22.          digitalWrite(thisPin,LOW);
23.          delay(100);
24.       }
25.       for(int thisPin = lastPin;thisPin >= firstPin; thisPin--)
26.       {
27.          digitalWrite(thisPin,HIGH);
28.          delay(100);
29.       }
30.       for(int thisPin = firstPin;thisPin <= lastPin; thisPin++)
31.       {
32.          digitalWrite(thisPin,LOW);
33.          delay(100);
34.       }
35.     }
36.     else {
37.       for(int thisPin = lastPin;thisPin >= firstPin; thisPin--)
38.       {
39.          digitalWrite(thisPin,LOW);
40.       }
41.     }
42.   }
```

Note (What is a button?):

Button mechanics

As you probably already know, a button is simply a wire that may or may not be connected depending on its position. In practice, there are several of them, differing in size, electrical characteristics, possible mechanical positions, and so on.

The normally open (NO) push button

In this part of the lab, we're going to use this type of push button. They have two positions:

- Released: the current does not flow, the circuit is disconnected; the circuit is said to be "open".
- Pressed: the current flows, the circuit is said to be closed.

The normally closed (NC) push button

This type of button is the opposite of the previous type, i.e. when the button is released, it lets current through. And vice versa:

- Released: the current flows, the circuit is connected; the circuit is said to be "closed".
- Pressed: the current does not flow, the circuit is said to be open.

Switches

Unlike a push button, a switch acts like a toggle. One press closes the circuit and a second press is needed to open it again. It therefore has stable states (open or closed). A switch is said to be bistable. You come across them every day when you turn on the light.

Button electronics

Symbol

The push button and the switch do not have the same symbol on the electronic scheme. The first is represented by a bar which must make contact to close the circuit or break contact to open the circuit. The second is represented by a wire that opens a circuit and can move to close it. Here are their symbols, and it's important to remember them:

NO push button **NC push button** **Switches**

They often have four legs. If this is the case, the pins are connected in pairs. This means that they work in pairs. So you have to be careful when you connect it, otherwise you'll get the same behaviour as a wire (if you connect two linked pins).