EXPERIMENT NO. 1 (Group B)

Aim: Write a program using Arduino to control LED (One or more ON/OFF). Or Blinking

Outcome: Connectivity and configuration of Arduino circuit with basic peripherals like LEDS

- **Hardware Requirement:** Arduino, LED, 220 ohm resistor etc.
- **Software Requirement**: Arduino IDE
- **Theory:**

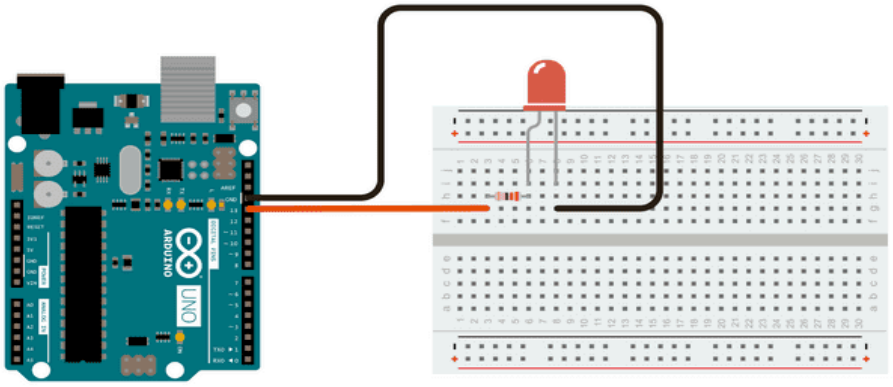
This example shows the simplest thing you can do with an Arduino to see physical output: it blinks the on-board LED.

This example uses the built-in LED that most Arduino boards have. This LED is connected to a digital pin and its number may vary from board type to board type. To make your life easier, we have a constant that is specified in every board descriptor file. This constant is *LED_BUILTIN* and allows you to control the built-in LED easily. Here is the correspondence between the constant and the digital pin.

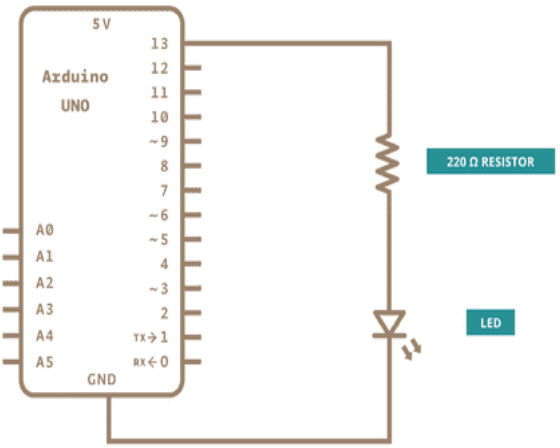
- D13 - 101
- D13 - Due
- D1 - Gemma
- D13 - Intel Edison
- D13 - Intel Galileo Gen2
- D13 - Leonardo and Micro
- D13 - LilyPad
- D13 - LilyPad USB
- D13 - MEGA2560
- D13 - Mini
- D6 - MKR1000
- D13 - Nano
- D13 - Pro
- D13 - Pro Mini
- D13 - UNO
- D13 - Yún
- D13 - Zero

If you want to lit an external LED with this sketch, you need to build this circuit, where you connect one end of the resistor to the digital pin correspondent to

the *LED_BUILTIN* constant. Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor. Connect the short leg of the LED (the negative leg, called the cathode) to the GND. In the diagram below we show an UNO board that has D13 as the LED_BUILTIN value.
The value of the resistor in series with the LED may be of a different value than 220 ohm; the LED will lit up also with values up to 1K ohm.



Schematic



Code

After you build the circuit plug your Arduino board into your computer, start the Arduino Software (IDE) and enter the code below. You may also load it from the menu File/Examples/01.Basics/Blink . The first thing you do is to initialize LED_BUILTIN pin as an output pin with the line

pinMode(LED_BUILTIN, OUTPUT);

In the main loop, you turn the LED on with the line:

digitalWrite(LED_BUILTIN, HIGH);

This supplies 5 volts to the LED anode. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

digitalWrite(LED_BUILTIN, LOW);

That takes the LED_BUILTIN pin back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the **delay()** commands tell the board to do nothing for 1000 milliseconds, or one second. When you use the **delay()** command, nothing else happens for that amount of time. Once you've understood the basic examples, check out the BlinkWithoutDelay example to learn how to create a delay while doing other things. Once you've understood this example, check out the DigitalReadSerial example to learn how read a switch connected to the board.

Conclusion: -

_____

_____

_____

_____

_____

Experiment no 4

```
/*
Blink
Turns an LED on for one second, then off for one second, repeatedly.
*/
// the setup function runs once when you press reset or power the board
void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(7, OUTPUT);
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(7, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(7, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
digitalWrite(8, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(8, LOW); // turn the LED off by making the voltage LOW
delay(1000); //
```

```
    digitalWrite(9, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(9, LOW); // turn the LED off by making the voltage LOW
    delay(1000); //
}
```