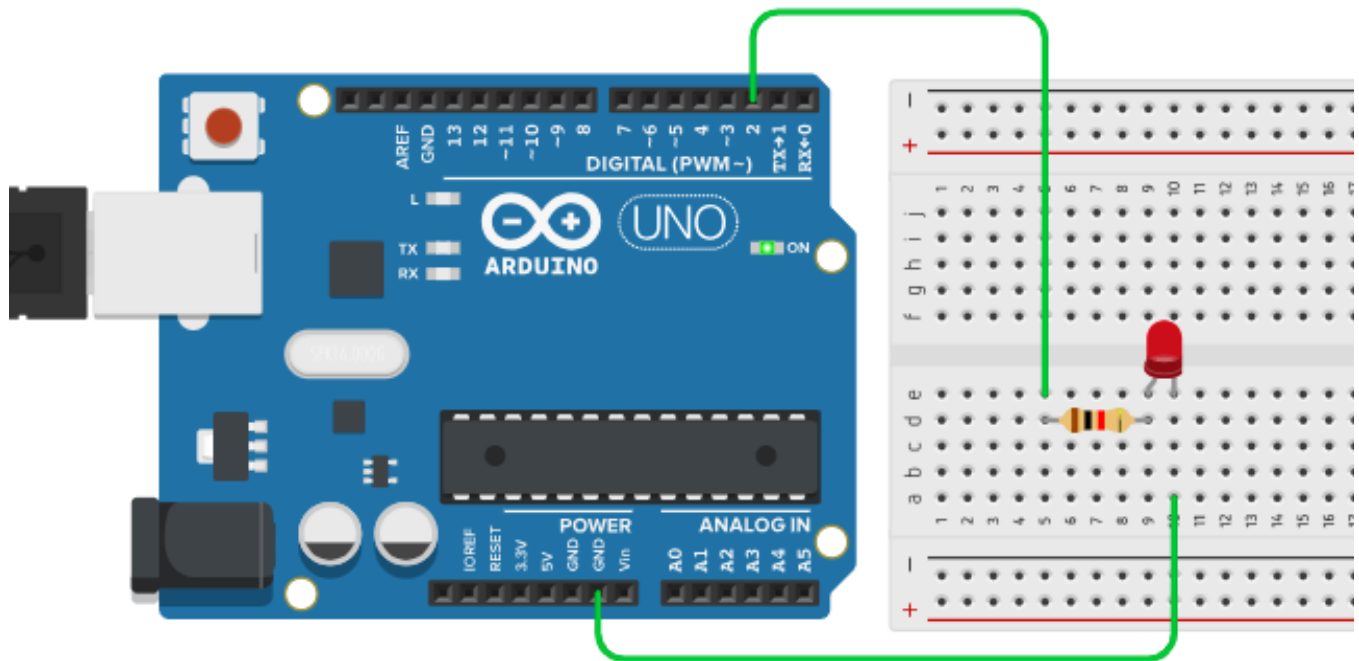


## Activity: Write an Arduino Program to Blink an LED

### Objective:



We have blinking lights all around us. Whether it's a police car, an ambulance, metal detectors at the railway stations, and airports, security systems and alarm systems in our homes and apartments, blinking LED lights are a part of our daily life.





High-rise buildings must have blinking LED lights fitted at different levels and at the top. These are called aviation obstruction lights. Speaking of aviation, all airplanes and helicopters have blinking LED lights. Most electronic toys these days have at least one blinking LED light in them. Thus, something as simple as a Blinking LED light is almost an inevitable part of our daily life whether we see them or not.


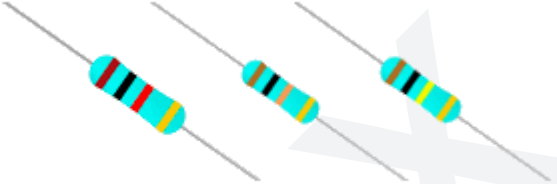


So, in today's activity we will write our very first Arduino program and that will be for blinking a small LED light. This activity consists of many sub activities. For the first part we will write a program for blinking the Arduino's on-board LED. In the second part, we will set up a simple hardware on the breadboard for blinking an external LED and make necessary changes in the code to make this work.

All Arduino boards have an on-board LED light connected to pin D13. An external LED, on the other hand, can be connected to any digital pin of the arduino.

Blinking an LED is a combination of 2 events viz. Turning the LED On and Turning it Off. So technically, we will learn now to turn an LED On and Off using Arduino.

## Materials Required:

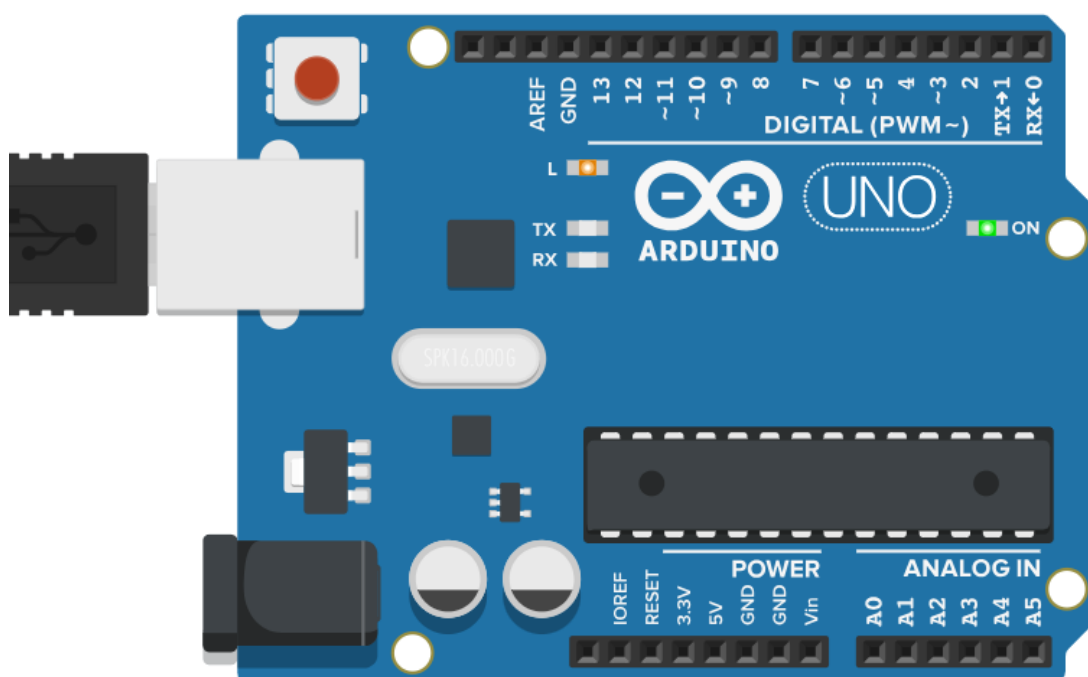
S.no.	Name	Qty	Image
1	Arduino	1	 A blue Arduino Uno R3 microcontroller board with a USB Type-B port, a DC power jack, and a 5-pin header.
2	Breadboard	1	 A standard white breadboard with a central strip of 40 pins and two side strips of 25 pins each.
3	M - M jumper wires	5	 Five multi-colored jumper wires with male-to-male connectors on both ends.
4	M - F jumper wires	5	 Five multi-colored jumper wires with a male connector on one end and a female connector on the other.

5	USB cable for Arduino	1	
6	Resistors: 1k (Brown Black Red) 10k (Brown Black Orange) 100k (Brown Black Yellow)	1 each	
7	5mm LED	1	
8	Lamp blox	1	

## Connection Diagram:

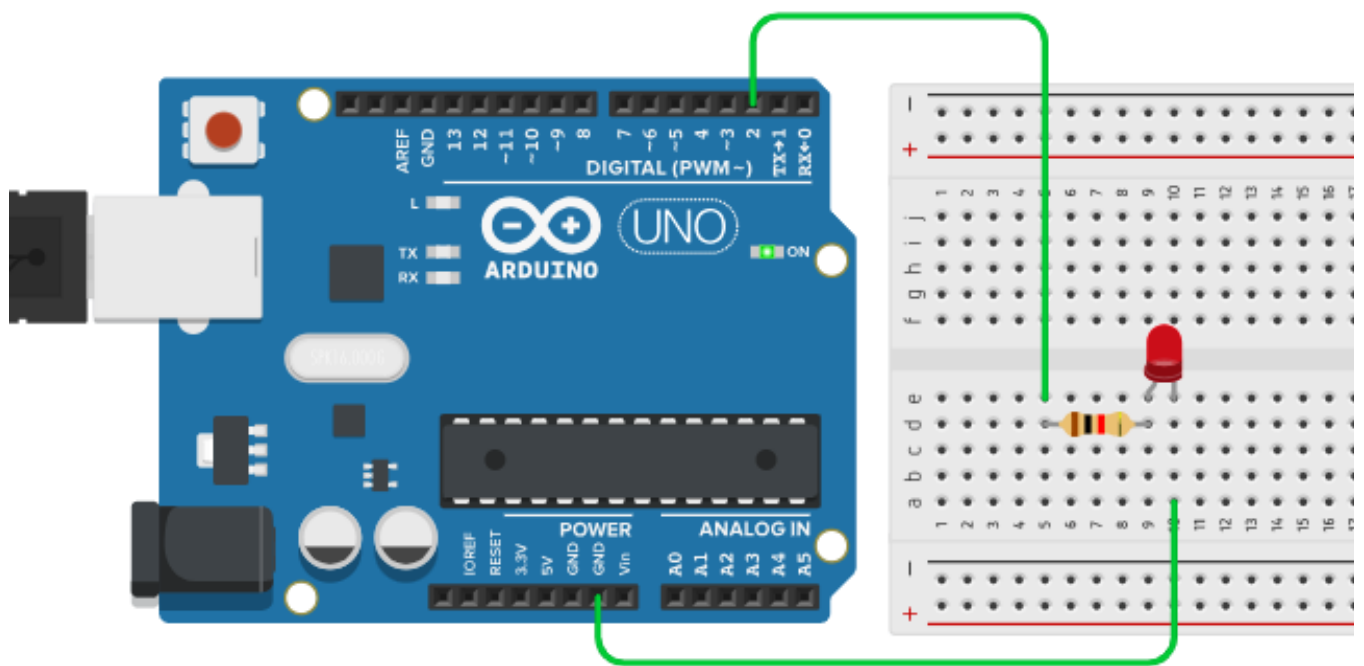
### Part 1:

In the first part, there is no separate connection diagram. It is just the Arduino board fitted with the USB cable. Since this part uses the Arduino's on-board LED, there is no need for any external component.



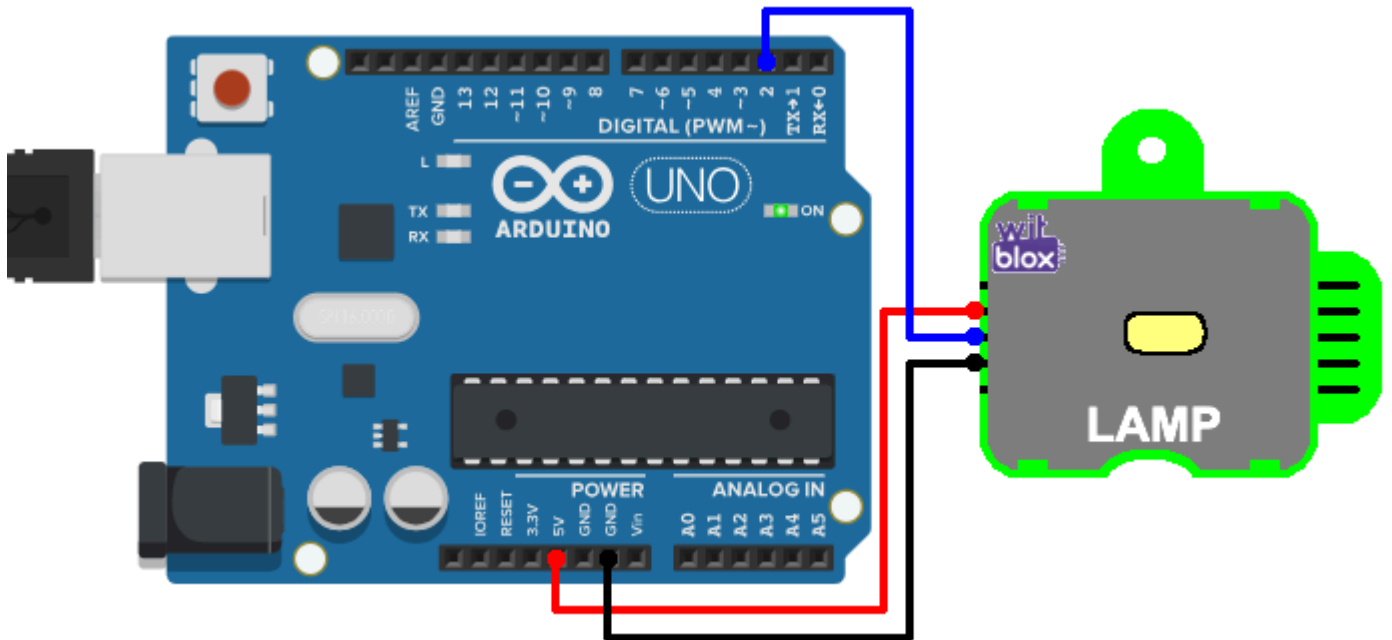
## Part 2:

In the second part, we use an LED (any color) and a 1k resistor (Brown Black Red) as external components. These components plug into the breadboard and are connected to the Arduino board via jumper wires as shown in the image below.



### Part 3:

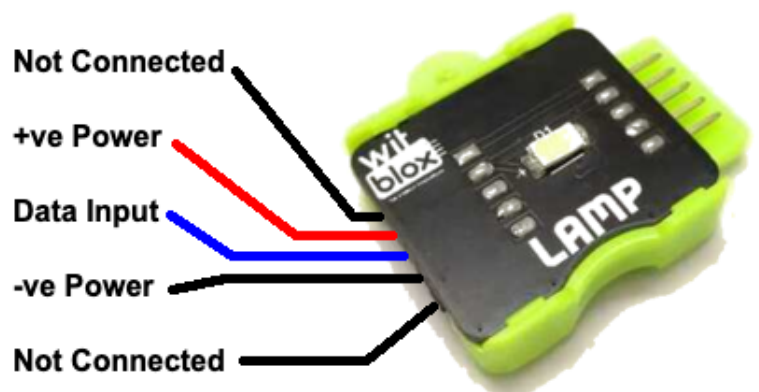
In the third part, we use the WitBlox Lamp blox as our external LED and connect it to the Arduino using the breadboard and jumper wires as shown in the image below.



Observe the difference in connections when connecting a single LED and connecting the Lamp blox.

When a simple LED is connected only 2 wires are needed. (refer Part 2 diagram)

1. For connecting the negative of LED to Gnd of Arduino.
2. For connecting the D2 pin of Arduino to one end of the 1k resistor.



When the Lamp blox is connected, 3 wires are needed.

1. For connecting the 5v pin of Lamp blox to the 5v pin of Arduino.
2. For connecting the Gnd pin of Lamp blox to the Gnd pin of Arduino.
3. For connecting the Data input pin of Lamp blox to the D2 pin of Arduino.

## Arduino Code:

When it comes to the Arduino programming, the code for all the three parts are exactly the except for the change in the pin number. All this will be clear when you see the code below.

### Code for Part 1:

```
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT); // LED_BUILTIN is pin D13
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH); // pin HIGH means ON
  delay(1000); // stay ON for 1 sec
  digitalWrite(LED_BUILTIN, LOW); // pin LOW means OFF
  delay(1000); // stay OFF for 1 sec
}
```

### Explanation:

The code is divided into two sections.

1. "setup" and
2. "loop"

The "setup" section is used for events or actions that are done only once, while the "loop" section is used for events and actions that are done repetitively forever (till the power is ON).

Declaring which pin will work as output is needed just once and is therefore written inside the "setup" section. The turning ON and OFF of the LED, on the other hand, needs to be done repetitively and so it is written inside the "loop" section.

Both "setup" and "loop" are called Functions in the programming environment. So, one can say that any Arduino program has at least 2 functions which are "setup" and "loop". Any program written for any version of Arduino will mandatorily have these 2 functions. It may have more functions with different names but these two: "setup" and "loop" are always there. Every line that ends with a semicolon (;) is a statement.

`digitalWrite(LED_BUILTIN, HIGH);`” this statement makes the D13 pin on the Arduino HIGH and this turns the on-board LED ON.

`digitalWrite(LED_BUILTIN, LOW);`” this statement makes the D13 pin on the Arduino LOW and this turns the on-board LED OFF.

`delay(1000);`” this statement makes the Arduino halt its execution for the time mentioned inside the bracket. The time here is denoted in milliseconds. So 1 second is written as 1000 milliseconds.

So, the overall observation by the user is:

1. The LED goes ON.
2. The LED stays ON for 1 second.
3. The LED goes OFF.
4. The LED stays OFF for 1 second.

Thus producing the blinking effect.

### Code for Part 2:

```
void setup()
{
  pinMode(2, OUTPUT);    // declare pin D2 as an output pin
}

void loop()
{
  digitalWrite(2, HIGH); // pin HIGH means ON
  delay(2000);           // stay ON for 1 sec
  digitalWrite(2, LOW);  // pin LOW means OFF
  delay(2000);           // stay OFF for 1 sec
}
```



## Explanation:

The only difference between the code of part 1 and part 2 is the pin number. Here, in part 2 we are using Arduino 's digital pin 2 (D2) for controlling the LED. To make a visible difference in output behavior, we have changed the delay time from 1 second to 2 second (1000ms to 2000ms).

## Code for Part 3:

```
void setup()
{
  pinMode(2, OUTPUT); // declare pin D2 as an output pin
}

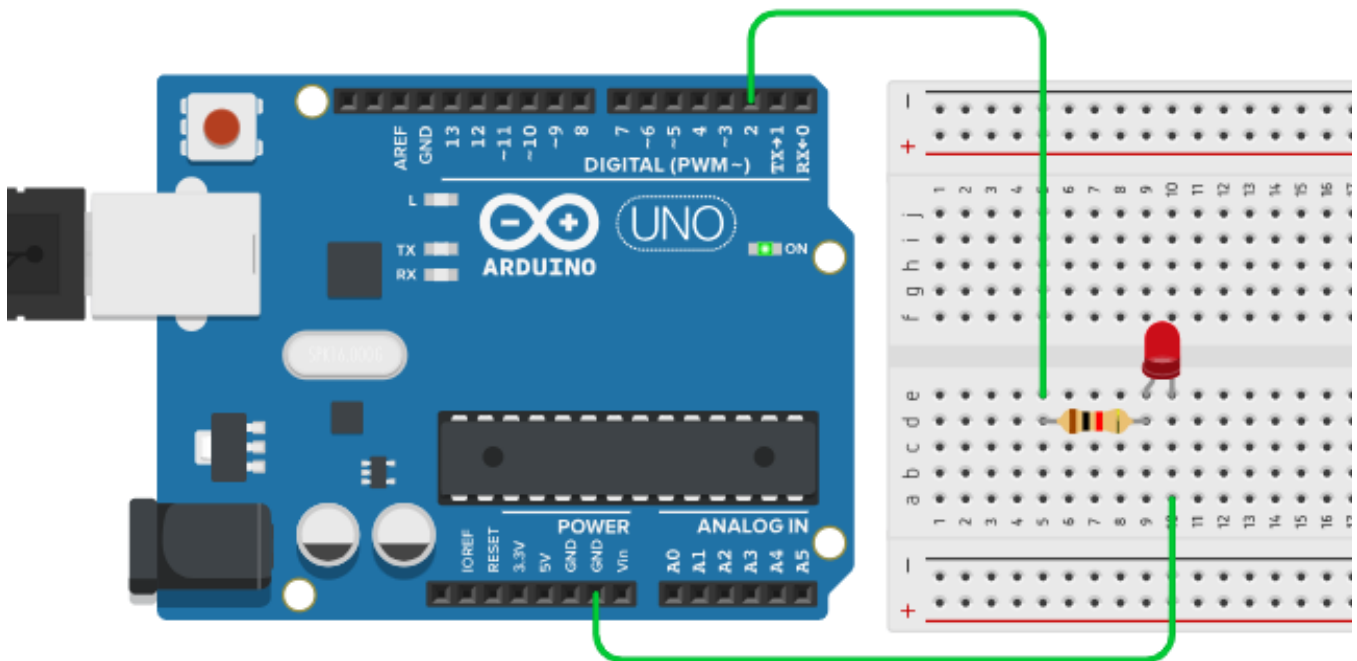
void loop()
{
  digitalWrite(2, HIGH); // pin HIGH means ON
  delay(2000); // stay ON for 1 sec
  digitalWrite(2, LOW); // pin LOW means OFF
  delay(2000); // stay OFF for 1 sec
}
```

## Explanation:

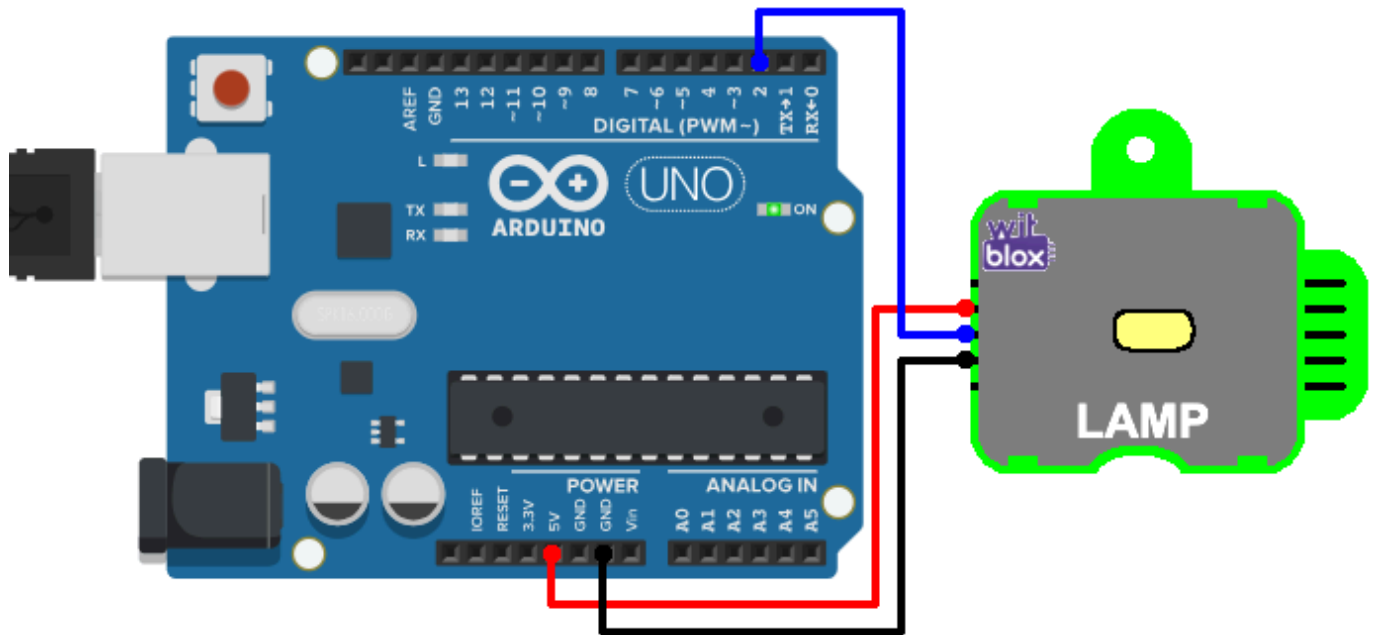
The code for Part 2 and 3 are identical. Even the pin numbers are the same. The difference is in the way the Lamp blox is connected. Since the Lamp blox has its own circuit which needs power to operate, the power pins of 5v and Gnd are connected from the Arduino to the Lamp blox. The data pin is connected to D2.

## Outcome and Observations:

1. For all the 3 versions of the activity, the outcome is exactly the same. The connected LED (on-board or external) goes ON and stays ON for a defined time and then goes OFF and stays OFF for a defined time and then the whole process repeats itself.
2. If you want to connect an LED of any other color just go ahead and swap the Red LED with the Green one or the Yellow one. You will see no difference except for the change in color.
3. You can experiment with different values of resistors provided in the kit. You will observe that:
  - a. 1k will make the blinks brightest.
  - b. 10k will make the blinks dimmer.
  - c. 100k will make the blinks almost invisible.



4. You can experiment with connecting the LED to different digital pins on the Arduino. All you need to do is make changes to the pin number. So, simply replace 2 with 5 in the code if you want to control the LED with pin D5 of the Arduino. here is an example for pin D5



```
void setup()
{
  pinMode(5, OUTPUT);    // declare pin D5 as an output pin
}

void loop()
{
  digitalWrite(5, HIGH); // pin HIGH means ON
  delay(2000);           // stay ON for 1 sec
  digitalWrite(5, LOW);  // pin LOW means OFF
  delay(2000);           // stay OFF for 1 sec
}
```

5. Bigger the number you put inside the delay bracket, the longer the blinks will be.
6. If you change both the delays with the same number, you get evenly spaced blinks. If, however, you use different numbers you get delays according to the numbers used after turning the LED ON and OFF i.e. uneven blinks.
7. To make blinking patterns you can use the statements multiple times and add a long delay at the end. Figure this part out on your own. Happy brainstorming.