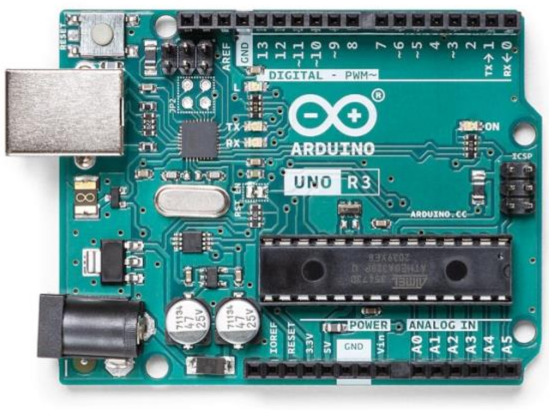
Arduino Micro-Controller



In the lab activities today you will be learning how to build circuits using ‘breadboards’ and electronic components which you will control using an Arduino (style) micro controller - a mini computer.

Instructions

Logging on to the system

- Working in pairs you will be given a user name and a password which you will use to log on to one of the bench top computers.
- When you have logged on to the system you will be asked to change your password - please enter your username (twice).

Connect Arduino

- Open the Windows tools menu - little ‘windows’ logo in bottom, left corner of screen
- Look for Arduino - select and open the Arduino IDE (integrated development environment)
- Connect your Arduino board to the computer using the USB cable provided
- In the ‘Tools’ menu look for the ‘Board’ option - make sure Arduino Uno is selected
- Select the ‘Port’ option directly below the ‘Board’ option
- Look at the bottom right corner of the for the ‘coms port’ number
- Make sure the same port is selected in the ‘Port’ option list

Activity 1

- When you have connected the board to the IDE-
- Select *File > Examples > 01.Basics > Blink* to open the built-in Blink example program.
- Click the ***Upload*** button to upload the program to the Arduino.
- Look at the Arduino. You should see the onboard LED blink on for 1 second and off for 1 second repeatedly.
- Class discussion - what is the code doing?
- Edit the program so the LED stays on for 2 seconds and off for 0.5 seconds.
  Note: the delay command uses milliseconds. There are 1,000 milliseconds in 1 second.
- Click the Upload button again to upload the new program to the Arduino.

Activity 2

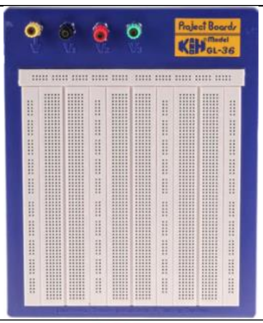
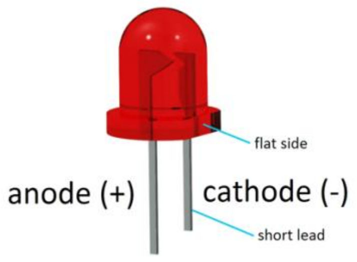
1. In this activity, you will build a circuit to connect an external LED to the Arduino. You will write a program to make the LED ‘blink’ on and off.
2. Add a second LED, write a program to make it blink at the same time as the first LED
3. Write another program to make the LEDs blink out of sync.

Learning Objectives

- Learn how to use a breadboard
- Build a circuit to connect an external LED to the Arduino
- Create a new Arduino program
- Control multiple LEDs with the Arduino

Materials

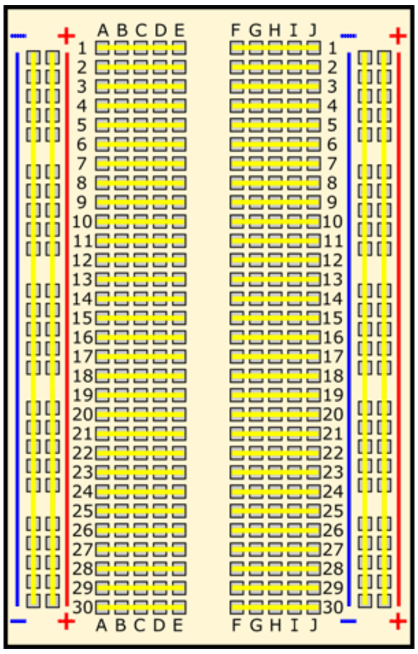
Each group will need:

Arduino UNO	
USB A-B cable	
Breadboard	
LEDs (2)	
150Ω or 220Ω resistor (2)	

10kΩ resistor (smaller than 220Ohm resistor)	
Jumper wires (assorted)	
Pushbutton	

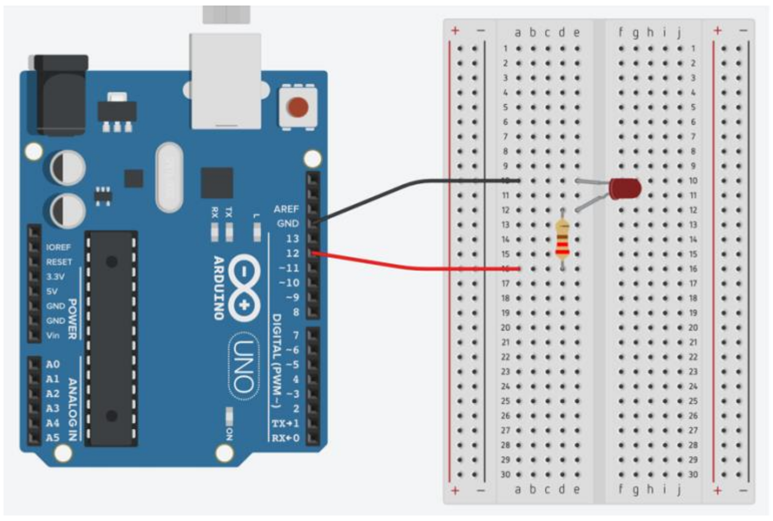
Activity 2

Breadboard - This image shows how holes in a breadboard are electrically connected to each other. Holes joined by a yellow line are electrically connected.



Activity 2.

Build the following circuit on the breadboard and connect it to the Arduino (gnd - ground, 12).



VERY IMPORTANT - INCLUDE A RESISTOR

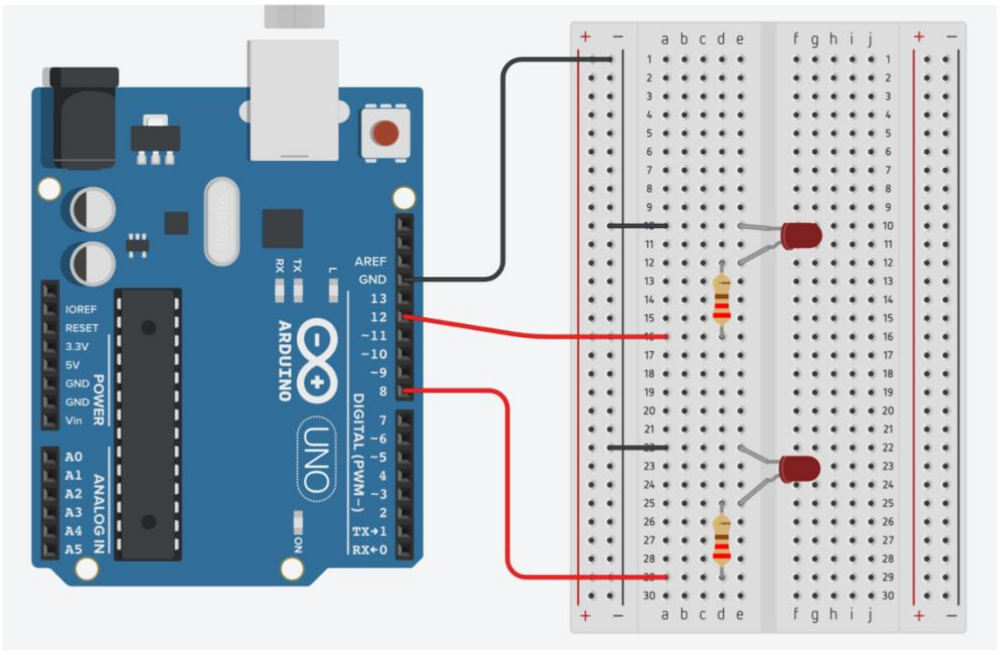
- Note that the legs of the LED (light) are of different length, the long leg is the Cathode or Negative connection.
- Open the Arduino IDE and create a new program (***File > New***). Save the program on your computer (***File > Save***).
- Type in code to blink the external LED

```
void setup()
        {
        // put your setup code here, to run once:
        pinMode(12,OUTPUT);  // set pin 12 as output
        }

void loop()
        {
        // put your main code here, to run repeatedly:
        digitalWrite(12,HIGH);  // turn LED on
        delay(1000);          // wait 1 second
        digitalWrite(12,LOW);   // turn LED off
        delay(1000);          // wait 1 second
        }
```

1. Upload the program to the Arduino. Make sure that your LED blinks on and off.

Example:



2. Connect another LED to Arduino pin 8 (don't forget the resistor). Edit your code to turn the second LED on and off at the same time as the first LED.

3. Edit your existing code to include the second LED.

4. Upload the new program to the Arduino. Make sure that both LEDs blink on and off at the same time.

5. Edit your code a third time to make the LEDs blink out of sync with each other.

Activity 3

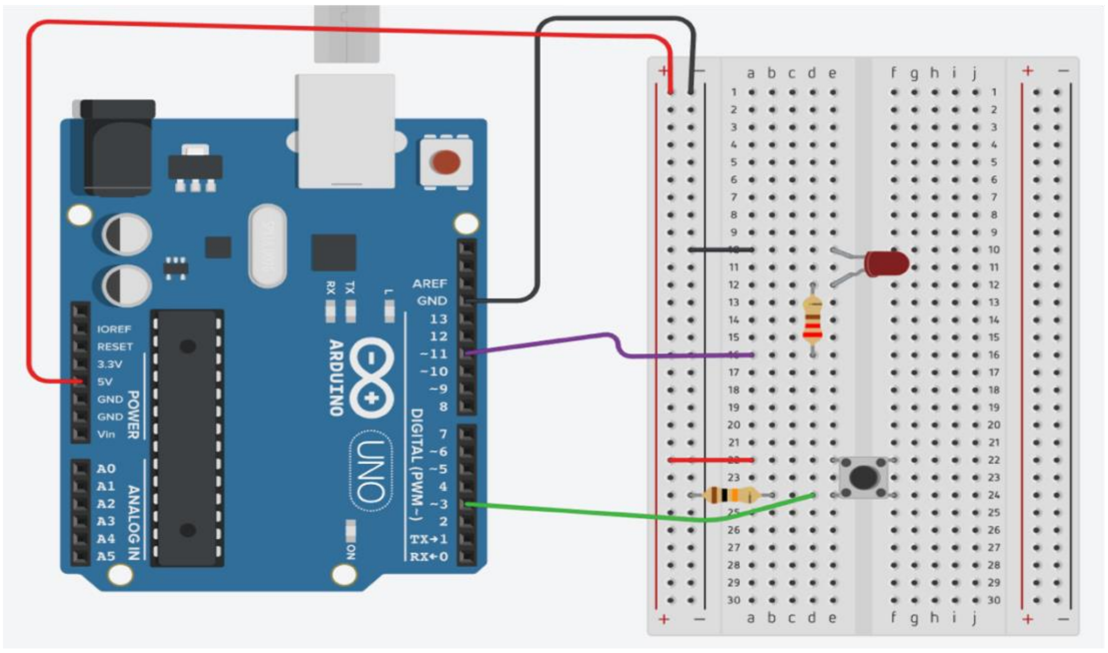
In this activity, you will learn to use a button as an input to the Arduino in order to control an LED. You will also learn about several new concept along the way, including the if/else statements, and (if we have time) variables in Arduino programs.

Materials

Each group of students will need:

- Arduino UNO
- USB A-B cable
- Computer with USB-A port* and Arduino IDE installed
- Breadboard
- LED
- 150Ω or 220Ω resistor
- Jumper wires (assorted)
- Pushbutton
- 10kΩ resistor (smaller than 220Ohm resistor)

1. Assemble this Circuit:



2. Create and save a new program in the Arduino IDE as follows:

```
void setup()
        {
        pinMode(12,OUTPUT); // set pin 12 as output
        pinMode(2,INPUT);   // set pin 2 as input
        }

void loop()
        {
        if(digitalRead(2) == HIGH)
                { // if the button is pressed
                digitalWrite(12,HIGH);    // turn the LED on
                }
        else
                { // else, if the button is not pressed
                digitalWrite(12,LOW);    // turn the LED off
                }
        }
```

3. Upload the code and make sure it works as expected (LED should be on when button is held down).
4. Modify the code so the LED is **on** when the button is **not** pressed, and turns **off** when the button **is** pressed.

Solutions - Code to Reverse LED Behaviour
There are multiple ways to change the code to make the LED be on by default and turn off when the button is pressed.

Method 1: swap "HIGH" and "LOW" in the digitalWrite commands:

```
void setup()
                {
                pinMode(12,OUTPUT); // set pin 12 as output
                pinMode(2,INPUT);   // set pin 2 as input
                }

void loop()
        {
        if(digitalRead(2) == HIGH)
                { // if the button is pressed
                digitalWrite(12,LOW);     // turn the LED off
                }
        else
                { // else, if the button is not pressed
                digitalWrite(12,HIGH);    // turn the LED on
                }
        }
```

5. Code with Variables
This program uses a variable for the LED pin. If you want to use a different pin, you only need to change the variable once at the beginning of the program.

```
int led_Pin = 12; // declare variable for LED pin
void setup()
        {
```

```
        pinMode(led_Pin,OUTPUT); // set pin 12 as output
        pinMode(2,INPUT);   // set pin 2 as input
        }

void loop()
        {
        if(digitalRead(2) == HIGH)
                { // if the button is pressed
                digitalWrite(led_Pin,HIGH);     // turn the LED on
                }
        else
                {  // else, if the button is not pressed
                digitalWrite(led_Pin,LOW);    // turn the LED off
                }
        }
```

6. Edit the code to include a variable for the button pin as well as the LED pin.

Activity 4

1. Can you build a circuit to run this code?

This example code is available under File > Examples > 03.Analog >Fading.

```
int ledPin = 9;    // LED connected to digital pin 9
void setup() {
// nothing happens in setup
}
void loop(){
        // fade in from min to max in increments of 5 points:
        for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {
        // sets the value (range from 0 to 255):
        analogWrite(ledPin, fadeValue);
        // wait for 30 milliseconds to see the dimming effect
        delay(30);
        }
        // fade out from max to min in increments of 5 points:
        for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {
        // sets the value (range from 0 to 255):
        analogWrite(ledPin, fadeValue);
        // wait for 30 milliseconds to see the dimming effect
        delay(30);
        }
        }
```