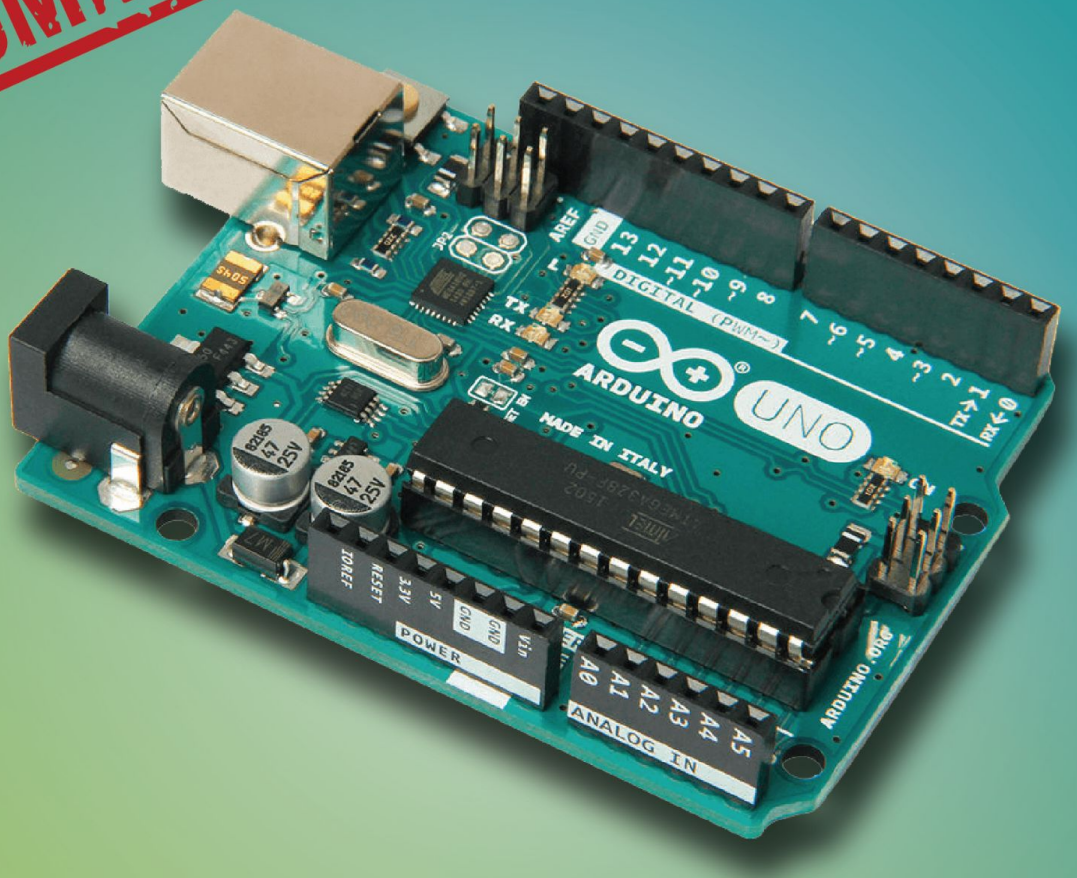
Arduino

Project's Book

ARDUINO

COMING SOON



Level 1

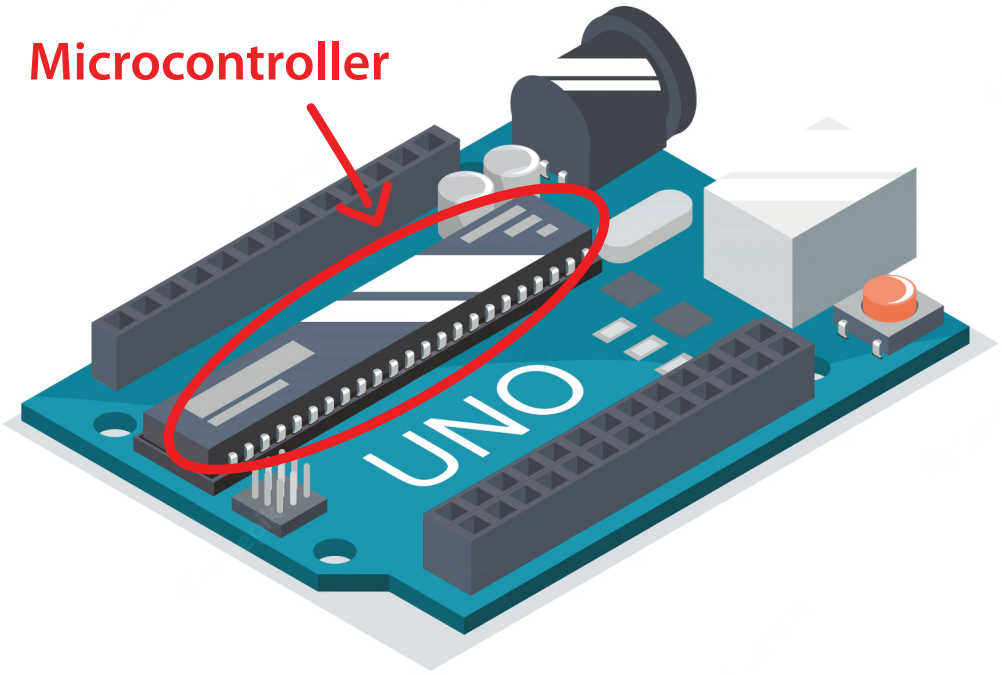
Rana Dajani

Introduction

Have you ever looked at a gadget and wondered how it works? Maybe it was a remote-controlled boat, a vending machine, or an electronic toy? Or have you wanted to create your own robot? Where and how do you start? The Arduino board can help you find some of the answers to the mysteries of electronics in a hands-on way.

Arduino system inspires you to create devices that can interact with the world around you. By using an almost unlimited range of input and output devices, sensors, indicators, displays, motors, and more, you can program the exact interactions required to create a functional device.

Over the years Arduino has been used as the “brain” in thousands of creative projects; by sending a set of instructions to the microcontroller on the board.



You are surrounded by dozens of microcontrollers every day: they are embedded in timers, thermostats, toys, remote controls, microwave ovens, and even some toothbrushes. They have been programmed to sense and control activity using sensors and actuators.

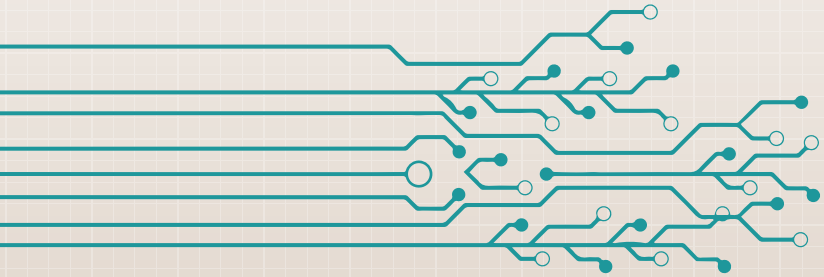
1. **Sensors listen to the physical world (input devices)**. They convert energy that you give off when you press buttons, or wave your arms, or shout, into electrical signals. There are many kinds of sensors (e.g. buttons and knobs are sensors that you touch with your fingers).
2. **Actuators take action in the physical world (output devices).** They convert electrical energy back into physical energy, like light and heat and movement.
3. **Microcontrollers listen to sensors and talk to actuators (brain).** They decide what to do based on a program that you write.

> Microcontrollers and the electronics you attach to them are just the skeleton that make your projects responsive, but only you can make them beautiful.

Table of contents

Comprehensive

Glossary



Lesson

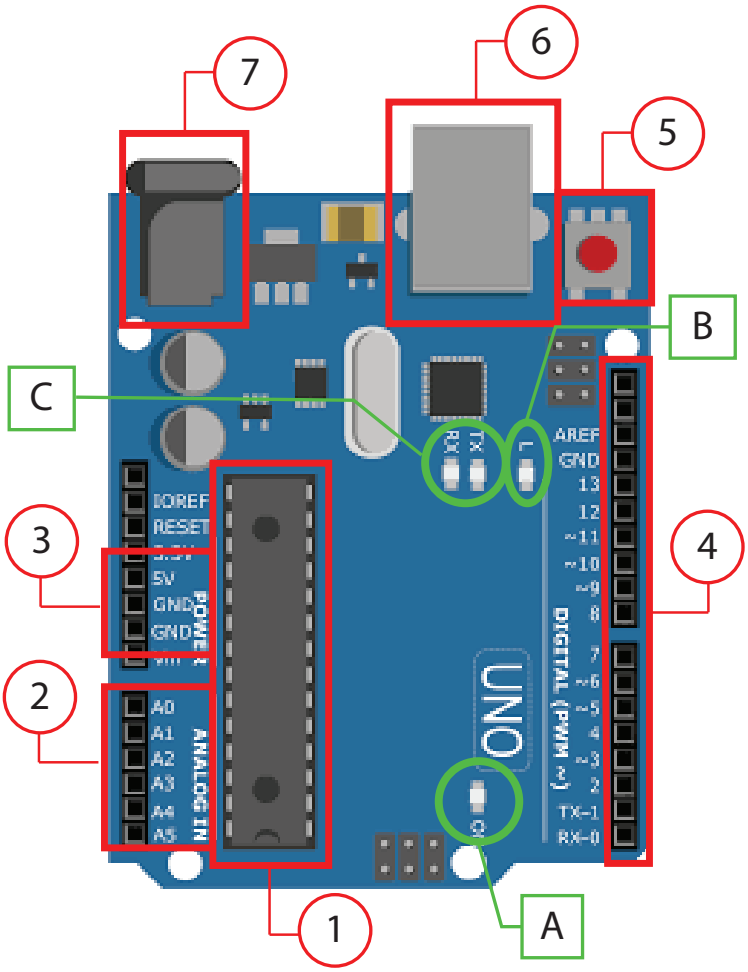
Getting Started

1

🛠 Get to know your tools

Introduction to the microcontroller board…

1. **ATmega microcontroller**
   The '**brain**' of your board
2. **Analog Input**
   Pins that use analogRead()
3. **GND and 5V Pins**
   Provide **power** (5V) and **ground/GND** (0V) to the circuit
4. **Digital Pins**
   Pins that use
   digitalWrite()
   digitalRead()
   analogWrite() –only with PWM (~) symbol pins
5. **Reset Button**
   Reset the microcontroller
6. **USB Port**
   Power the board, upload **sketches** (programs), communicate with your program
7. **Power Connector**
   Power the board without plugging USB, using an external power source (e.g. 9V battery)

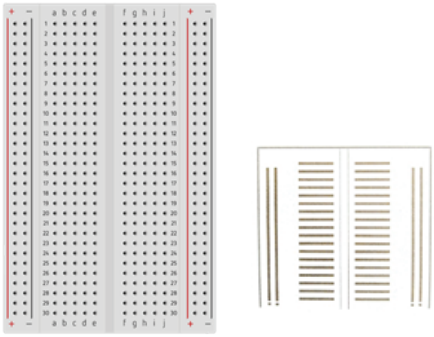


A. Power LED
Indicates the board is receiving power

B. Pin 13 LED
Built-in LED actuator

C. TX and RX LEDs
Indicate communication between board and computer. Flickers during sketch upload and communication

Basic Circuit Components



Breadboard

The breadboard is the primary place you will be building circuits. The horizontal and vertical rows (used for power and ground connections) of the breadboard carry electricity through thin metal connectors under the top layer of plastic. The middle row breaks the connection between the two sides of the board. The holes allow you to connect wires and components together without having to use soldering

Jump wires

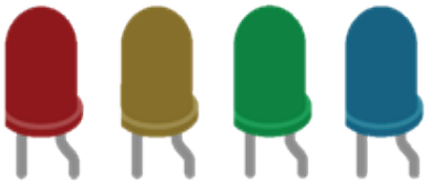
Used to connect components to each other on the breadboard

USB Cable

Allows you to connect your board to the computer for programming. It also provides power to the board

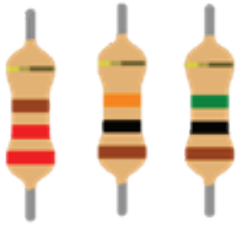
Battery Snap

Used to connect a 9V battery to power leads that can be easily plugged into the breadboard or the Arduino



LED

Light emitting diode that illuminates when electricity passes through in one direction

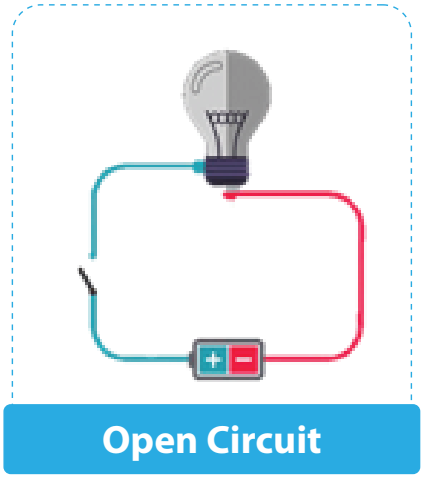


Resistors

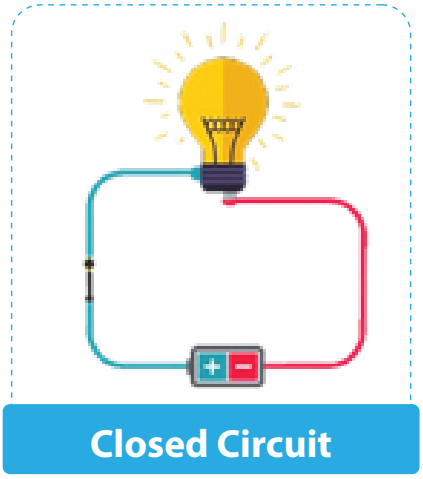
Resist the flow of electrical energy in a circuit, changing the voltage and current

Learning about circuits

A circuit is a complete path around which electricity can flow. It must include a source of electricity (power), such as a battery. Conductive materials, like metal wires, link the positive point of higher potential energy (often referred to as power or +ve) and negative ends of the battery (often referred to as GND or –ve), the point where potential energy is lowest in the circuit. In these type of circuits electricity only flows in one direction (direct current, or DC). In an open circuit, there is a break along the line, and the current (electric flow) stops.



Open Circuit

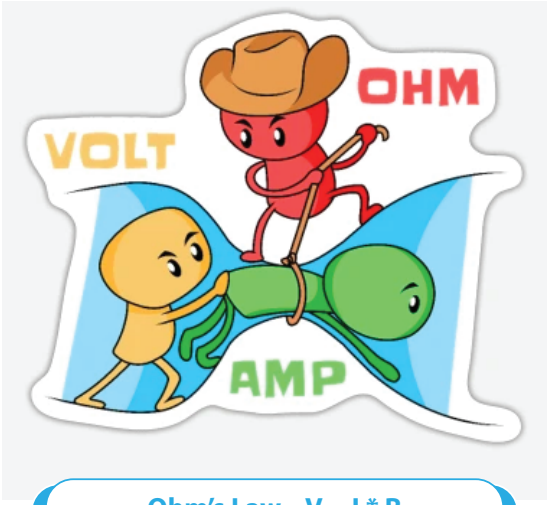


Closed Circuit

When electric current flows, it can be used by electrical appliances, such as light bulbs, that convert the electrical energy into other forms of energy. Things that convert other forms of energy into electrical energy are called **sensors**, and things that convert electrical energy into other forms of energy are called **actuators**.

In order to talk about electrical circuits, you need to be familiar with the terms voltage, current, resistance and about Ohm's Law, which defines the relationship between all three.

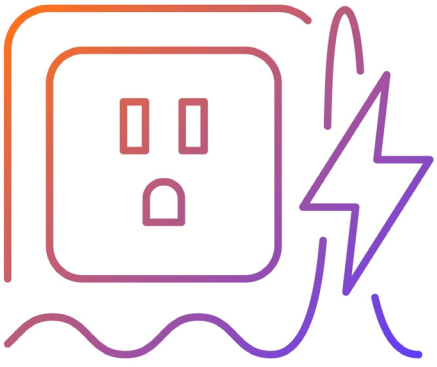
- **Voltage** (measured in volts – V) is the force that pushes electrons through a circuit to produce electricity.
- **Current** (measured in amperes – A) is the act of the electrons flowing through the circuit (rate of flow).
- **Resistance** (measured in ohms – Ω) is a force that counteracts the flow of current.



Ohm's Law V = I * R
Voltage = current * resistance

Note:

1. All the **electrical energy gets used up in a circuit** by the components in it. Each component converts some of the energy into another form of energy (light, heat, sound, etc.).
2. The **flow of current** at a specific point in a circuit will always be the same coming in and going out.
3. Electrical current will seek the path of least resistance to ground. Given two possible paths, more of the electrical current will go down the path with less resistance. If you have a connection that connects power and ground together with no resistance, you will cause a **short circuit**, and the current will try to follow that path. In a short circuit, the power source and wires convert the electrical energy into light and heat, usually as sparks or an explosion.



Software Setup

These software options will provide you a programming environment where you can write and experiment with your code.

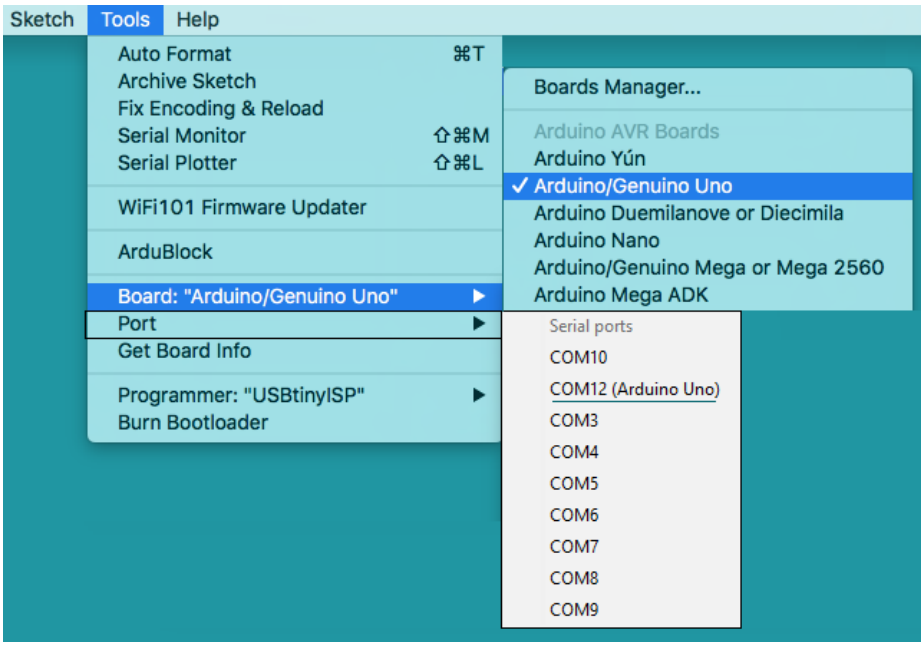
1. **Arduino IDE**

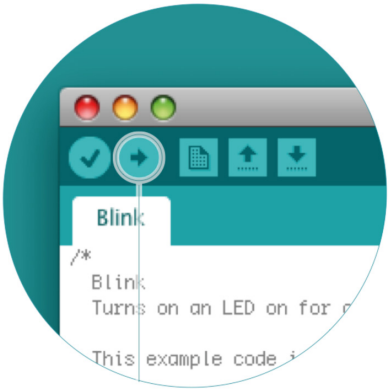
   In order to use the Arduino board, you need to program the microcontroller. You do this using the "Arduino IDE" (IDE = Integrated Development Environment) software, which allows you to write programs, which we call "sketches," and upload them. You can download the latest version of the IDE, and find the installation instructions, from: arduino.cc/download

After you have installed the Arduino IDE, it is time to set up your Arduino board.

You will tell Arduino IDE which board you are using. There are a lot of options! The one in your kit is an Arduino Uno.

Next choose the serial port your Arduino is connected to from the TOOLS > SERIAL PORT menu.





This is likely to be the COM port with the highest number. There is no harm in guessing wrong, and if it doesn't work, try the next one. To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect and select that serial port.

Finally, you will upload the Arduino sketch to the board!

Arduino Uno R3 Clone Driver Fix

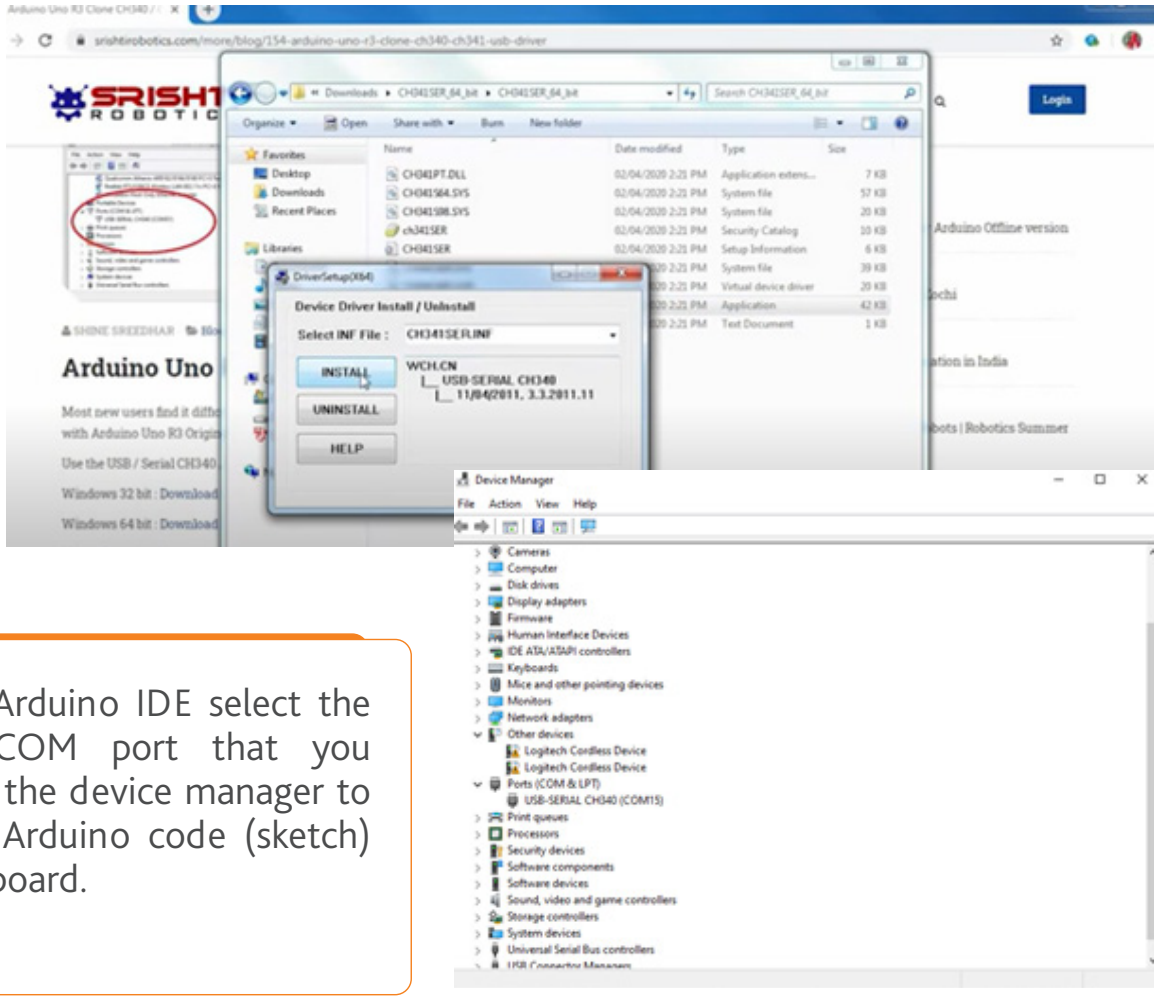
For the Arduino IDE software to recognize an Arduino clone microcontroller you need to download and install an Arduino Uno R3 clone USB Driver.

https://www.srishtirobotics.com/more/blog/154-arduino-uno-r3-clone-ch340-ch341-usb-driver

*Download the serial driver suitable to your device

Connect the Arduino board to your computer using the USB cable.

If installation is successful, the new USB driver will be visible in the Devices Manager of Windows.

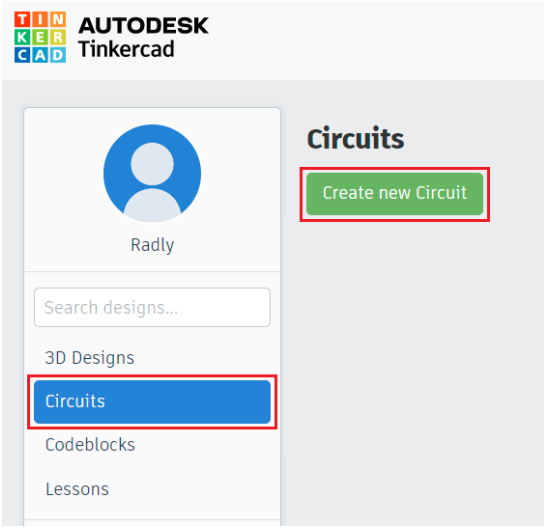


In the Arduino IDE select the same COM port that you have in the device manager to upload Arduino code (sketch) on the board.

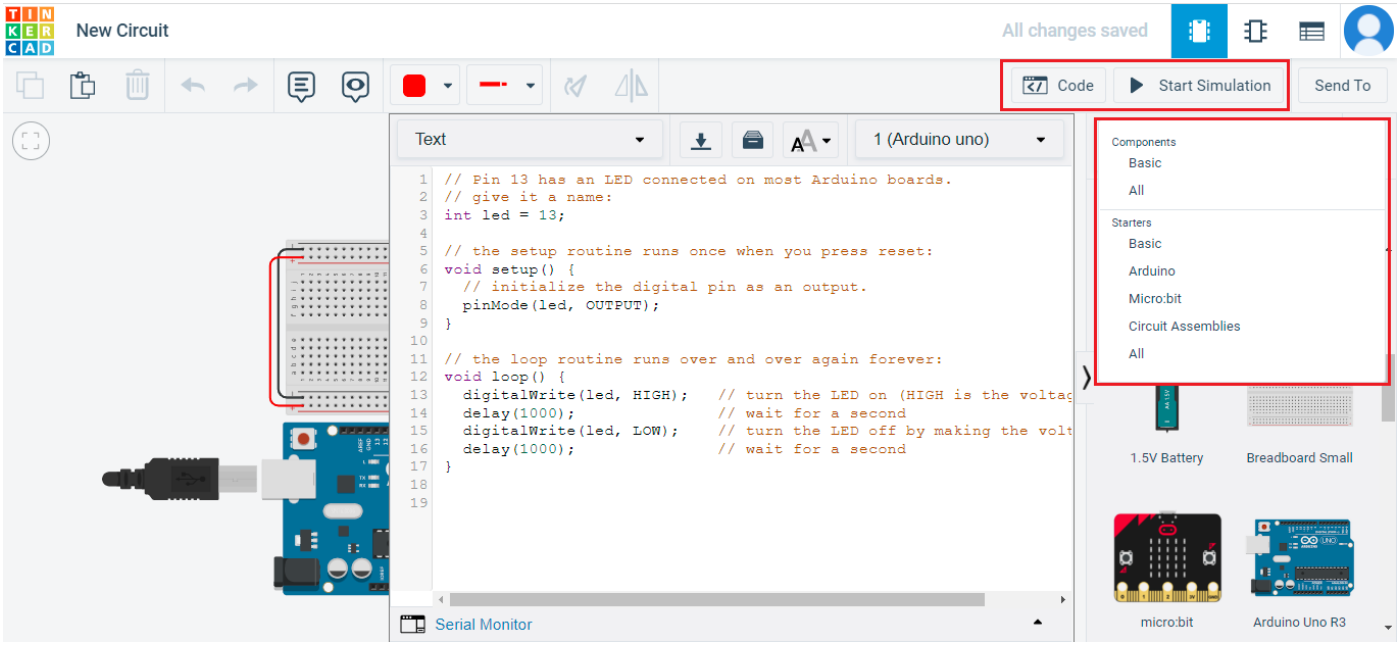
2. Tinker CAD (online open source)

Tinkercad Circuits allows students to use an interactive circuit editor, where they can explore, connect, and code virtual projects with a bottomless toolbox of simulated components.

You need to create an account and sign into Tinkercad, where you will find a dashboard of your recent designs. To see a view of your Circuit designs, simply click the Circuits link in the left menu and you can create something new by clicking the *Create New Circuit button.*



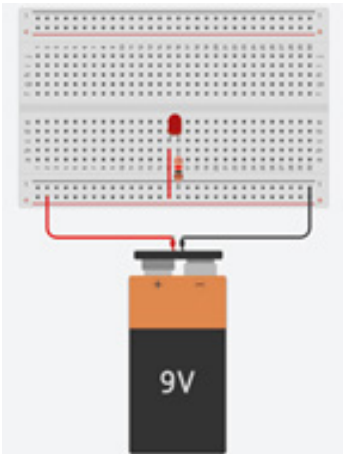
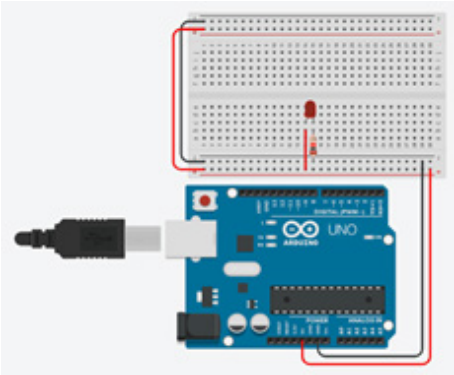
Tinkercad allows you to visualize what a program, you write in the "Code Editor", does using the simulated Arduino Board in the Workplane.



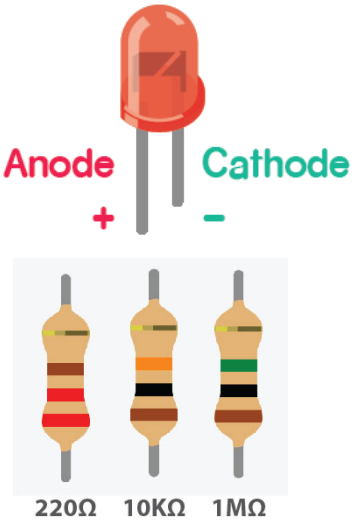
While you are writing an Arduino sketch, sometimes you make a mistake and the program will not be compiled (saved) and uploaded. This happens to professional programmers all the time!

The error message can tell you when a problem occurs and how to fix it.

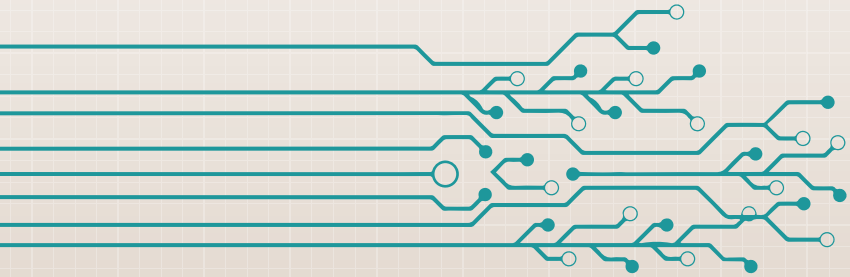
First circuit – Let there be light!

You can light up a LED in three ways:		
		
The LED can be connected straight to a 9V power source.	The LED can be connected to the breadboard (with its two legs in different rows of the board) which is powered by a 9V power source	The LED can be connected to the breadboard (with its two legs in different rows of the board) which is powered by the Arduino board through the USB power source

Notes:



- The longer leg of the LED, called the **anode**, is supposed to connect to the positive (+ve) power source and the shorter leg, the **cathode**, should connect to the ground negative (-ve) source.
- In all the 3 ways, a 220Ω **resistor** (the colored stripes identify the resistor's value) was added to the circuit to regulate the voltage flow through the LED, because a high voltage would cause the LED to pop!
- The resistor can go before or after the LED, and still protect it. The current that flows out of a battery is always equal to the current that flows back into the battery.
- The breadboard is powered through the horizontal rows "*voltage rails*", at the top and bottom of the breadboard, by connecting the red row to the positive (+ve) side of the power source (5V pin on the Arduino board) and the black row to the negative (-ve) side of the power source (GND pin on the Arduino board).
- When building your circuits for the future projects be sure to replicate the reference images of the lessons precisely.
- Be sure you are wiring the components to their correct polarity, which side is connected to the (+ve) end and which is connected to the (–ve).
- It is helpful to keep your wire colours consistent (red for power, black for ground) throughout your circuits.



Lesson

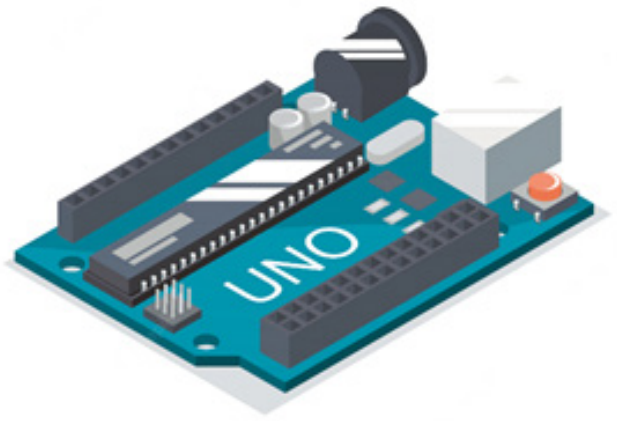
Blinking LED

2

Your first program

Now that you understand the basics of electronics, it is time to start controlling things with the Arduino, which means that you can now command the elements on when and how to work, how long and how often.

This is done through **programming** which is important for speeding up the input and output processes in a machine. Programing is also important to automate, collect, manage, calculate, and analyze processing of data and information accurately.



The circuit is setup differently in a way that the elements are connected to the Arduino pins directly. As you saw in the first lesson **(pg.00)** the Arduino board has **digital pins** (pins 2 – 13) on one side of the microcontroller, **analog input pins** (pins A0 – A5) on the other, as well as **power** (3.3V/5V pins) and **ground pins** (GND pins).

First you have to **configure** the Arduino's **digital pins** that you want to use with either with an INPUT or an OUTPUT component.

- pinMode(2, OUTPUT); or pinMode(3, INPUT).

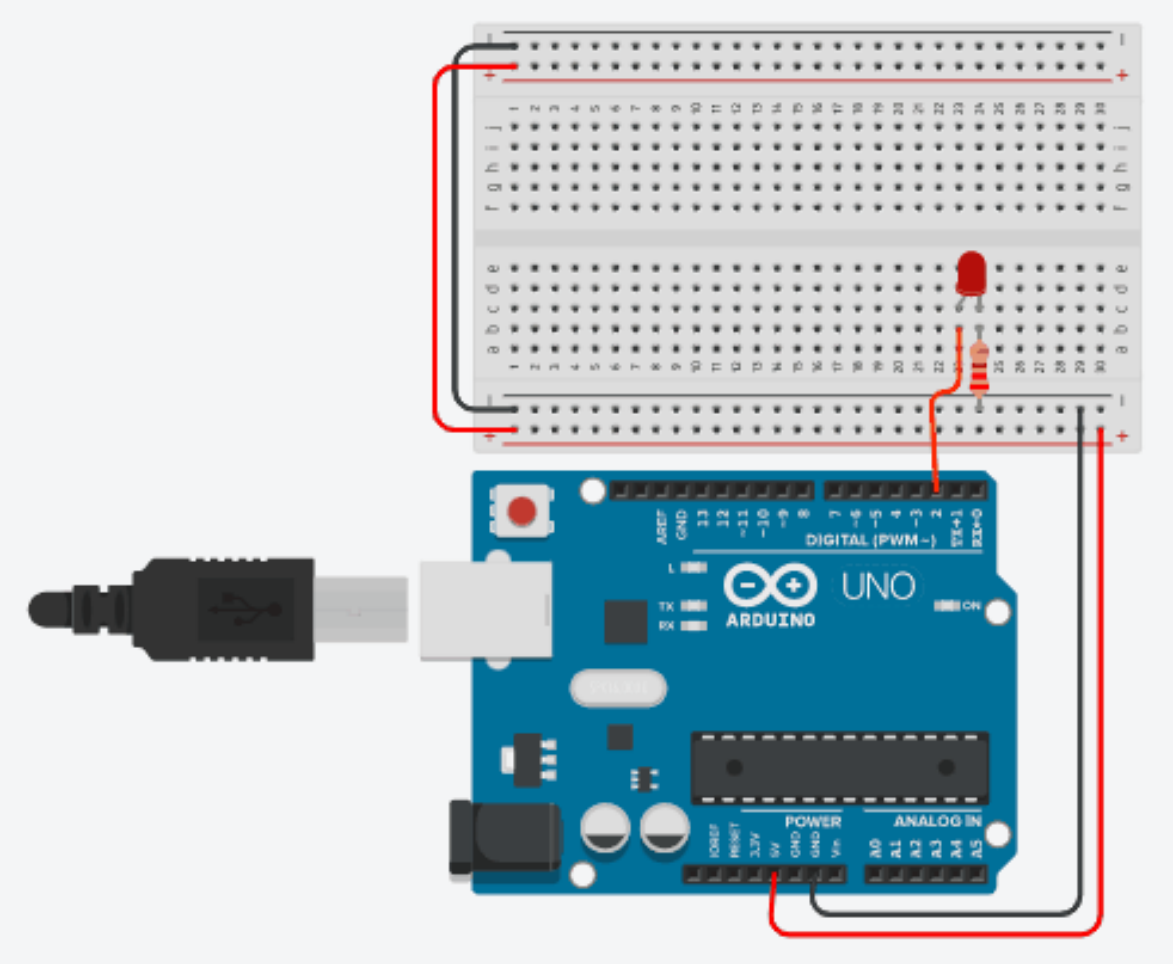
Next comes giving the **command** to that **component of the pin**:

- an OUTPUT component must be told if it should be *on/off*, given voltage(HIGH) or not(LOW), e.g. digitalWrite(2, HIGH);
- an INPUT component will collect readings based on its state (*on/off*),

e.g. digitalRead(3);

Build your circuit

Place a LED on the breadboard (with its two legs in different columns of the board), but instead of connecting the anode to the power, connect it to digital pin 2 to make it programmable by the board. Remember to also add a 220Ω resistor to protect the LED and connect it to the cathode and ground (GND).



Here, when the board is powered up by the USB, the LED will not automatically light up because it is no longer connected to the power source directly. Instead, being connected to digital pin 2, means it is waiting for an instruction on whether or not it should turn on.

To program this you need to understand the coding language that the microcontroller will understand.

The basic structure of the **Arduino C++ programming language** is made up of these two required parts, or functions, for the program to work.

```
void setup() {
statements;
}
```

setup() is the preparation. It is the first function to run in the program, it is run only once, and is used to set up pinMode() or initialize serial communication (you will learn this later).

```
void loop() {
 statements;
}
```

loop() is the execution. The function that follows next and includes the code to be executed continuously – reading inputs, triggering outputs, etc. This function is the core of all Arduino programs and does the bulk of the work.

*The loop() function will repeat the statements inside it over and over again and continue to run until you cut the Arduino's power supply.

Functions contain an enclosed block of statements in curly brackets {}. Each statement (line of code) ends with a semicolons (;). In this lesson you will use some functions including pinMode(), digitalWrite(), and delay() that are already part of the Arduino environment.

Build your program

To write your program you must know what you want to achieve from running this program.

The aim of this lesson is to program a LED to keep turning on for one second and turn off for another second (Blinking LED).

It is good practice to always write down your list of instructions as a pseudo code, which is a rough draft in your own words, before translating them into the programming language syntax.

The pseudo code would be:

Inside the setup function:

1. Configure the digital pin that the LED is connected to as an OUTPUT pin (since you want to light up the LED, you are using this LED as an OUTPUT component)

Inside the loop function:

2. Command the LED to turn on
3. Wait for 1 second
4. Command the LED to turn off
5. Wait for 1 second

* The previous steps (2-5) will repeat inside the loop

That translates to code like this:

```
void setup() {
    pinMode(2, OUTPUT);
}
void loop() {
    digitalWrite(2, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(2, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
}
```

Notes:

- The delay() function allows you to **pause the execution** of your Arduino program for a specified period. The function requires a whole number that specifies how many milliseconds the program should wait.
- // this is a comment – to include an explanation of your code, you can leave a comment. The computer will ignore anything that starts with //.
- You can write a multiline comment by enclosing the code lines within these comment markers (/* …. */)

  e.g. /* digitalWrite(2, HIGH);
  delay(1000);*/

* These lines will not be executed.

Take it further!

Challenge

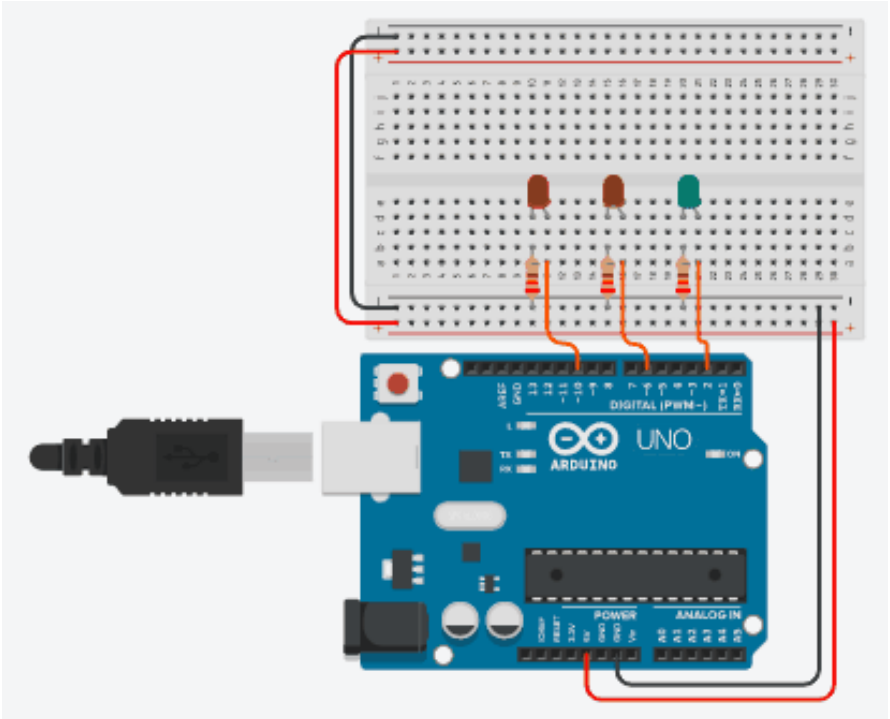
Traffic Light Simulator

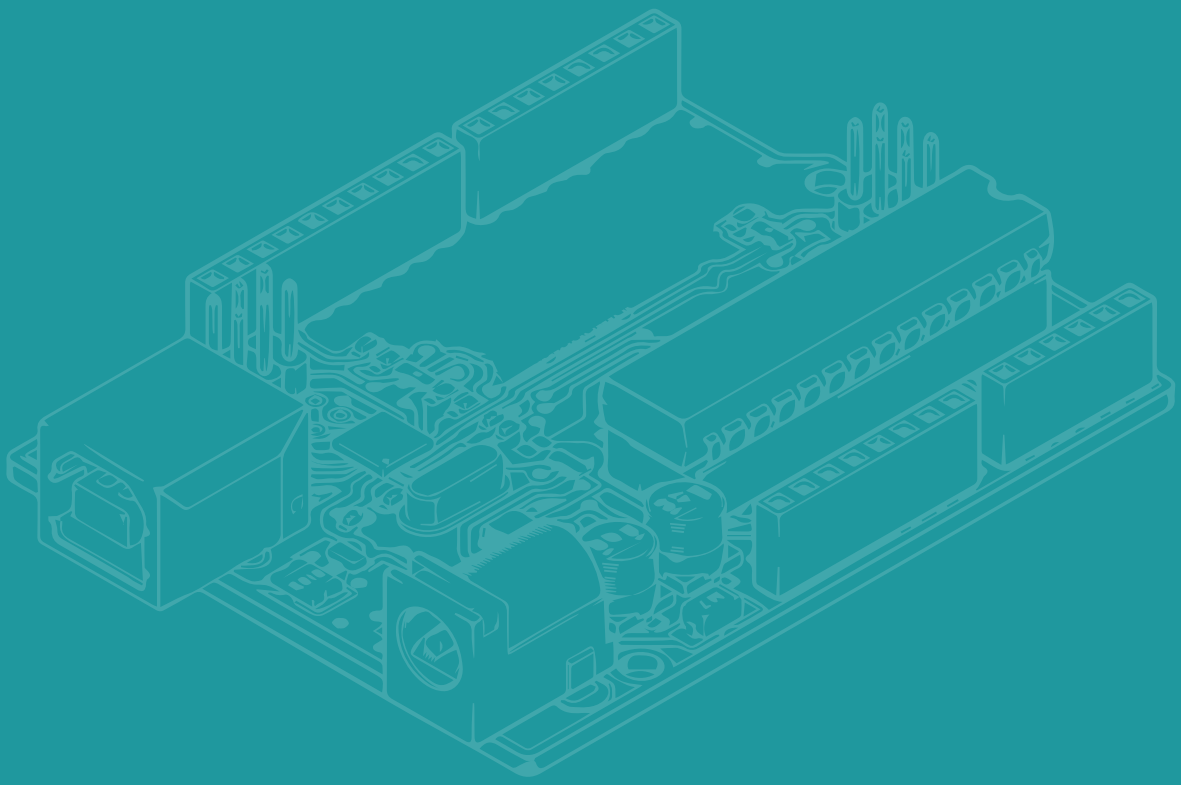
Traffic lights are great examples of simple robotic systems that have had profound effects on society.

Using code and a simple circuit, we will create a model of a three coloured traffic light. The circuit will consist of three lights: green, yellow and red, that are each connected to the Arduino board.

The traffic light will be green for 5 seconds, yellow for 1.5 seconds, and red for 3 seconds.

Hints:





ARDUINO