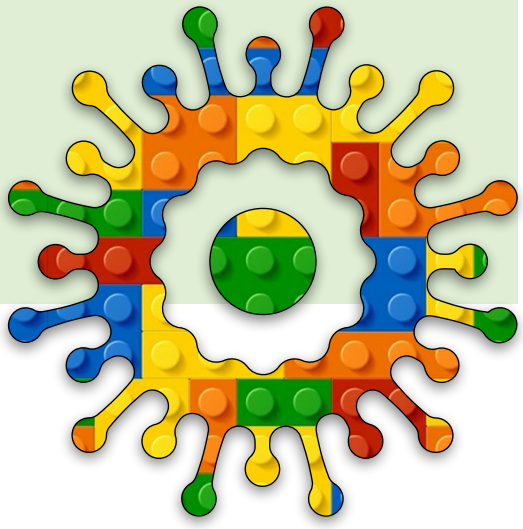
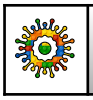
Introducing Robotics Module A Unit #1 hardware software





Content

Module A Unit #1 hardware and software

What is robotics?

This may seem a silly question because the answer is robots, right? Well, yes and no. Robotics is a very broad term applied to a wide range of scenarios. It can also be called physical computing.

It is anything where you programme a microprocessor (a chip); it could be something very simple and small like a micro-controller (likely) or a full-blown computer (unlikely).

What makes it robotics rather than just computing are the inputs and outputs. What data comes in and what it controls. In computing, you are controlling the screen; in robotics, you are controlling motors, LEDs, sound, and even a screen. The inputs are sensors, and these are many.

In many cases, robotics is about controlling motors. Hence we get the idea of something moving rather than something on a screen moving, for instance, an autonomous car, robotic arm, robot dog, or human-like robot.

But it can just be a source of data, physical data. These come from sensors such as: temperature, movement, light, sound. There is a sensor for everything. All these come under the general term robotics, and it is a very big and fascinating field to be involved in.

Robotics and artificial intelligence

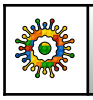
Robotics and artificial intelligence go together; we have all seen videos of robot dogs and walking, talking human-like robots. Although they seem to be almost interchangeable, they are two very different disciplines. It is also very, very difficult to combine the two; we are still a very long way from Terminator or WestWorld as portrayed in the media.

Yet AI is being used increasingly with driverless cars and drones. Many companies are spending eye-watering amounts of money to achieve the meshing of robotics and AI. In this module, we are going to start very simply combining the two and build up from there.

We will start by understanding robotics (or physical computing) using a simple board and some easily available components. Then we will use what we have learned about AI from ML5.js and apply it to the board. The board in question is a micro-controller; think of it as a very simple computer.

These boards can be very small indeed and are basically a single micro-chip with lots of connections. In our starting point, we are going to use something that doesn't look particularly tiny; it is called an Arduino Uno and it is the size of a credit card.

The Arduino Uno was designed for the single purpose of teaching people how to programme a micro-controller and is probably the most popular micro-controller ever made (and copied). It is relatively cheap, and there is a huge reservoir of information, ideas, support, as well as components that will help you on your journey if robotics interests you.



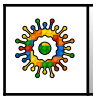
Embedded artificial intelligence

The Arduino Uno is great for what we are going to do in this module, but it does have a very small memory, and so you can only do so much. We don't put a neural network model into the micro-controller; we connect p5.js and ml5.js with the Arduino Uno through the serial port by USB.

What we will be trying to do later is use another (even smaller) micro-controller, which has a much larger memory, and install a simple neural network model inside the micro-chip. That is for much later and is called embedded AI.

For that, you may use an Arduino Nano 33 BLE Sense. Both have the capacity for a very small neural network model to be uploaded. To do that, they have special chips also on the board.

All that is for much later on and is much more challenging. Another alternative would be to use an SBC (single board computer), something like a Raspberry Pi. To programme that, you would use Python (and TensorFlow).



Introduction to Arduino Uno R3

This is the board we will be using for the first module. The Arduino Uno is a staple of the physical computing world and is where most people who get into robotics start.

The Uno is open source, which means that you can build your own and sell it. This has led to many companies marketing their own versions; this is perfectly fine and, on the whole, they are also cheaper versions.

If you can afford it, then I recommend buying an original from a reputable source such as Pi Hut. If you are on a bit of a budget, you can get copies from Amazon, Pi Hut, or eBay; however, make sure that you get a Uno that looks like the original (see Fig. 1 below).

When you start searching for an Uno, you will see a wide variety of versions (see Fig. 2). This also includes the newer R4 version (Wi-Fi and Minima). They are still an Arduino Uno, but I would steer clear of them for now as the original R3 is robust and adequate for what we want to do.

Fig. 3 and 4 show two examples of copies, and I found that they are half the price of an original Arduino Uno. I cannot guarantee the build quality, but I have no reason to think they are any less capable than the original. The choice is yours; I have used a mixture and never had a major issue.

Figure 1: Arduino Uno R3

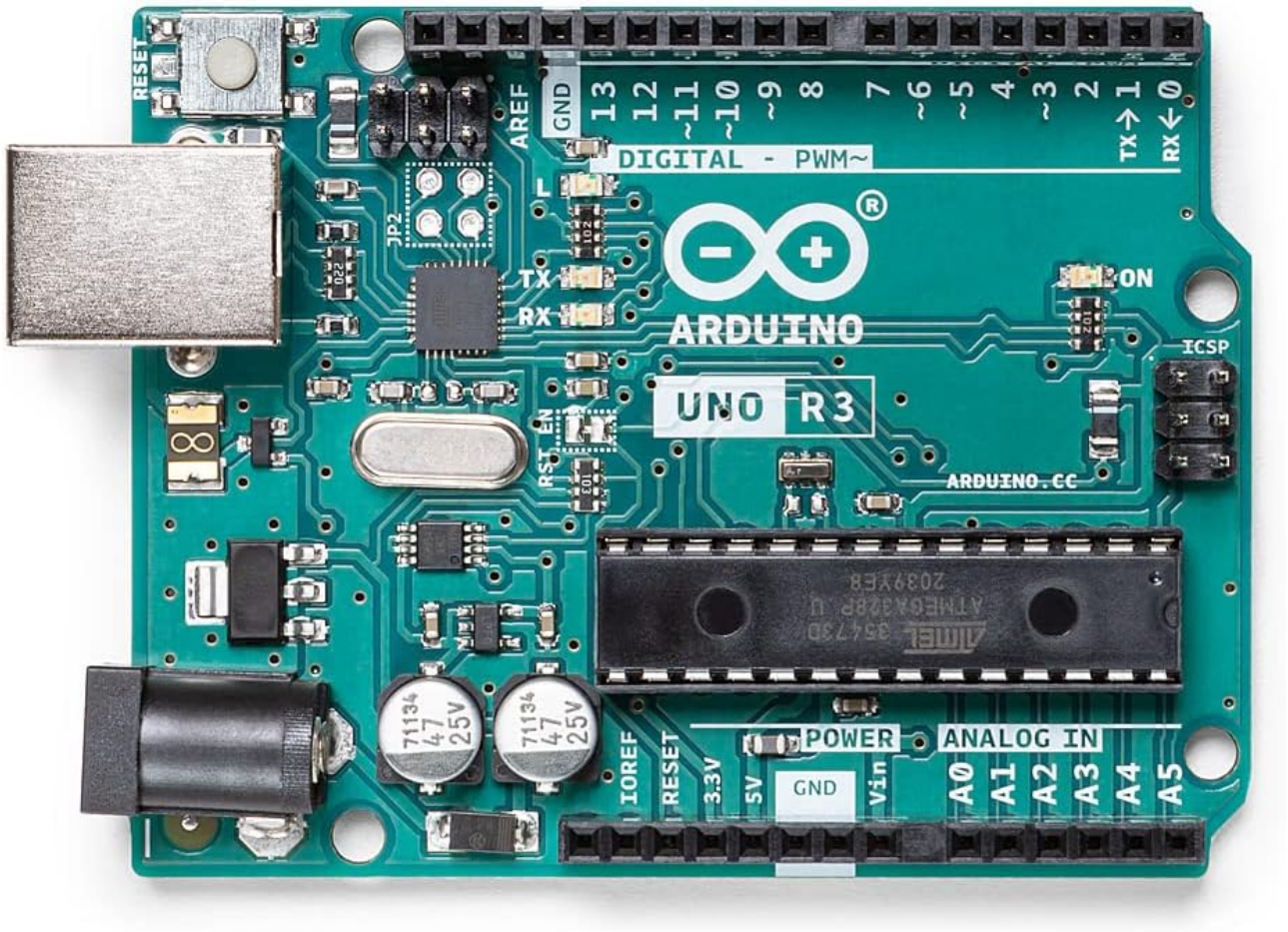


Figure 2: the Arduino classic family

Classic Family

In the Classic Family, you will find boards such as the legendary Arduino UNO and other classics such as the Leonardo & Micro. These boards are considered the backbone of the Arduino project, and have been a success for many years (and more to come).

Boards

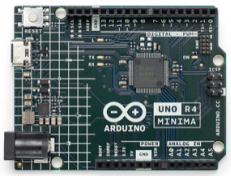
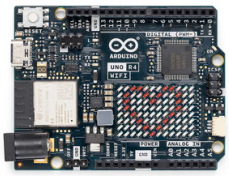
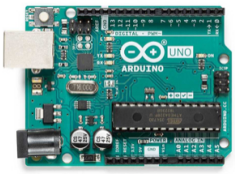
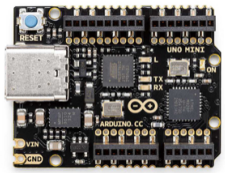
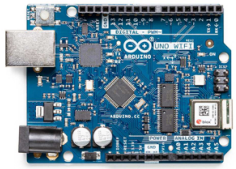
			
Arduino UNO R4 Minima	Arduino UNO R4 WiFi	Arduino UNO R3	Arduino Leonardo
			
Arduino UNO Mini Limited Edition	Arduino Micro	Arduino Zero	Arduino UNO WiFi Rev2

Figure 3: DFROBOT version (Pi Hut)

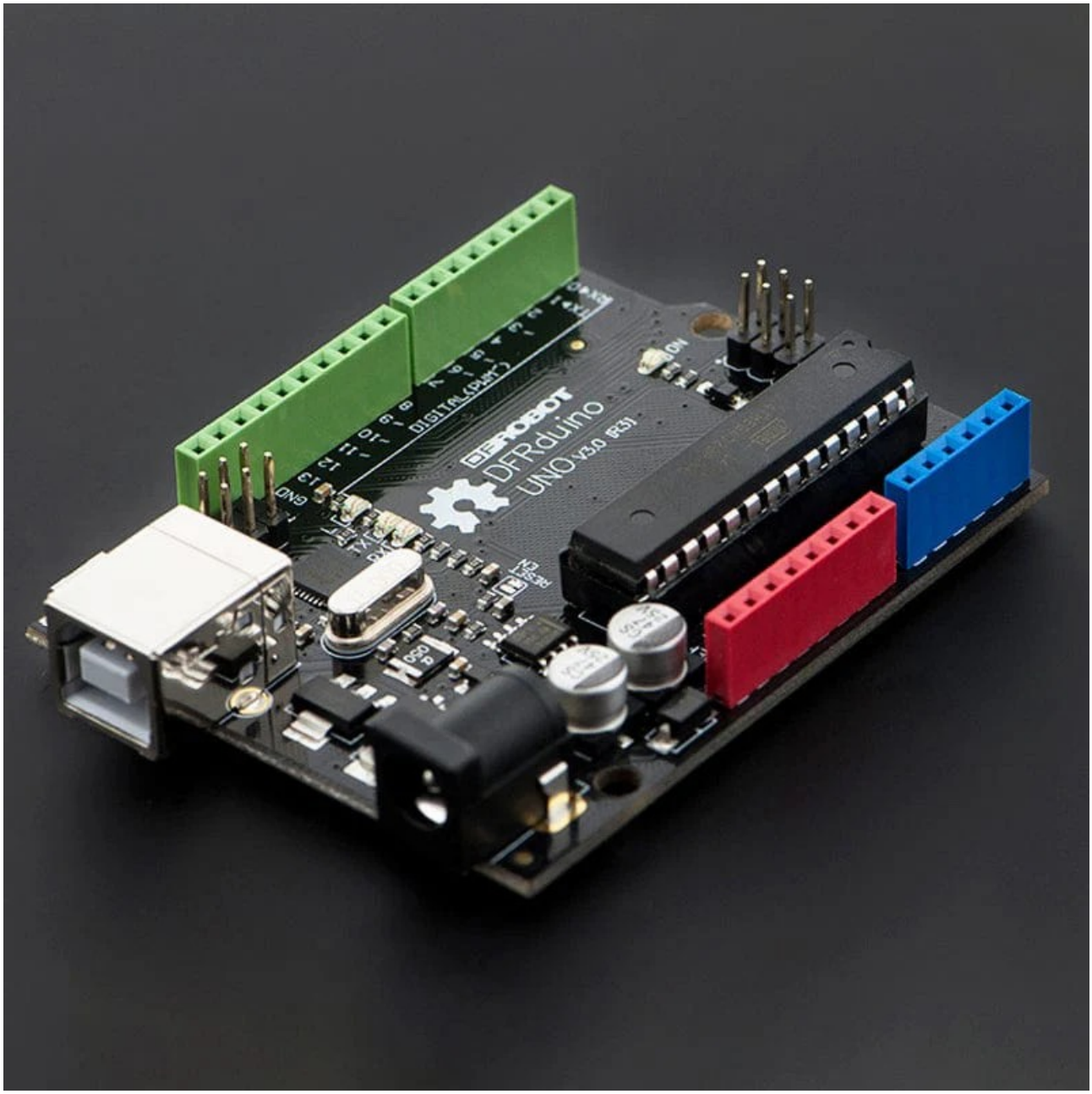
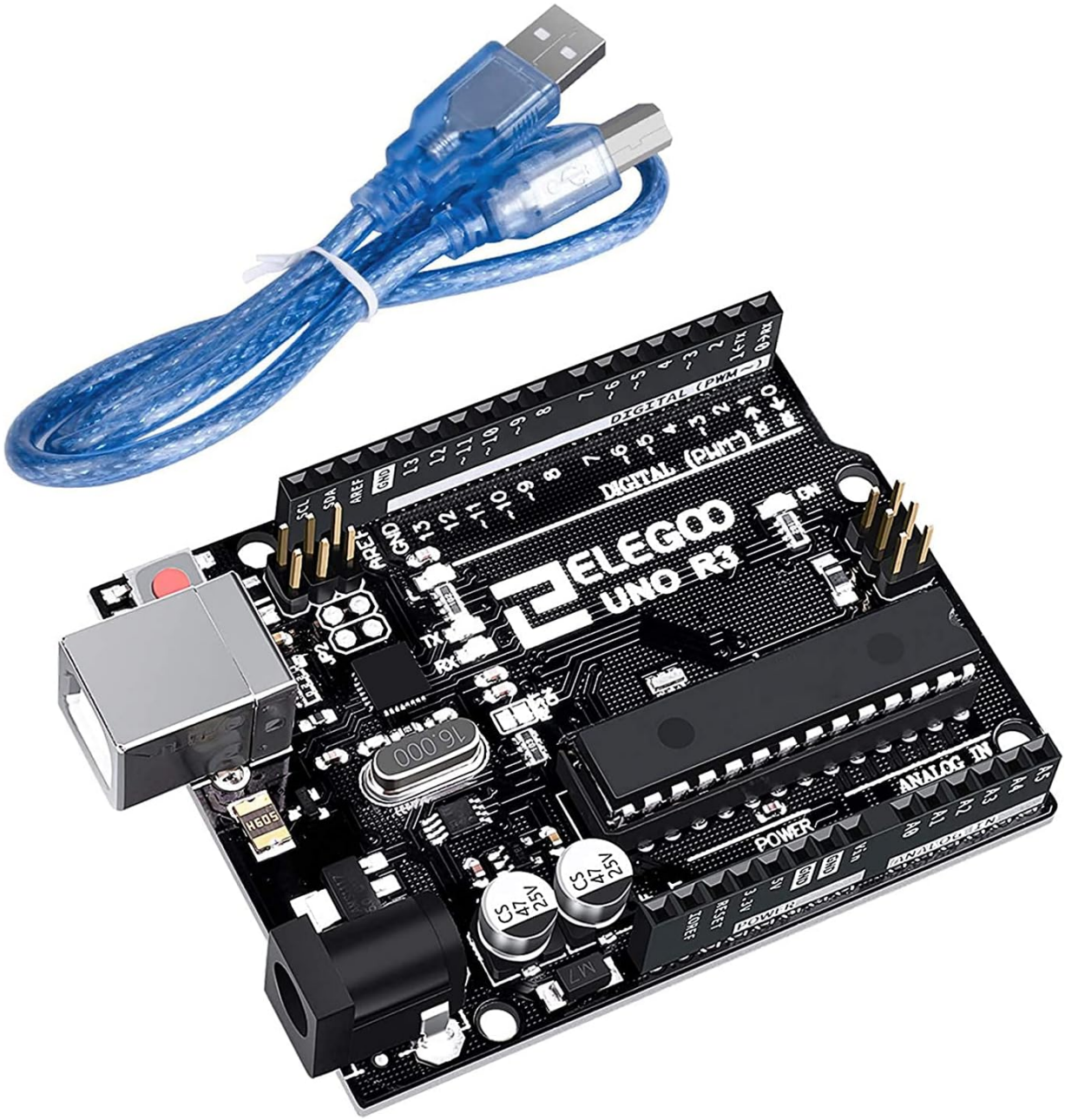
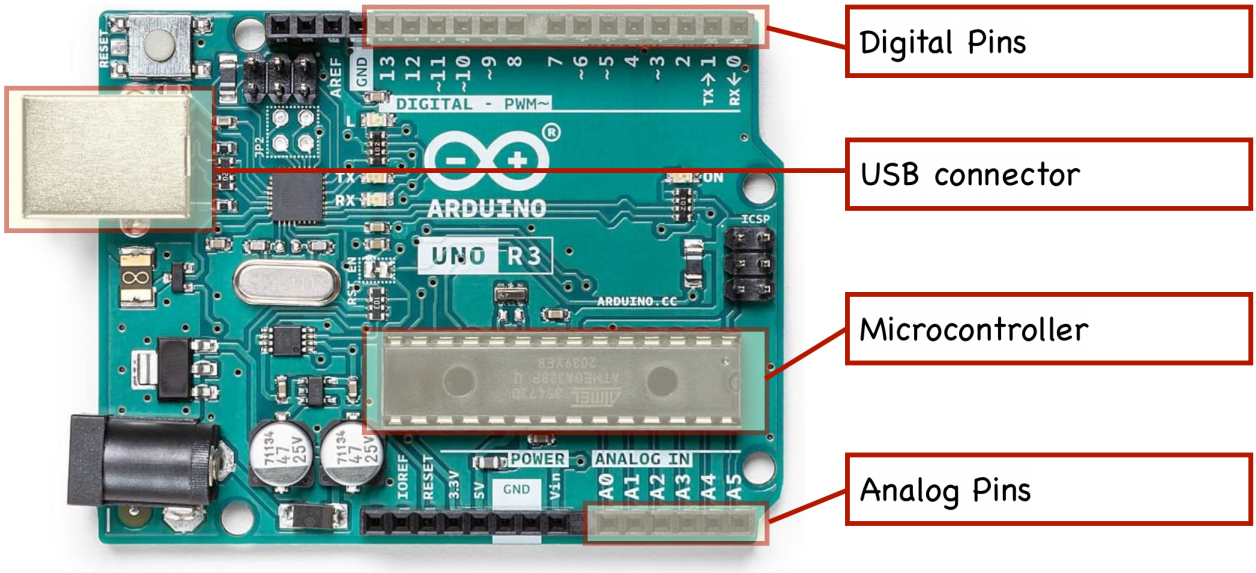


Figure 4: ELEGOO version (Amazon)



The main parts of the Arduino

Figure 5: main parts of the Arduino Uno



The above diagram shows you the general layout of the Arduino Uno and the key parts. You will be connecting your computer to the Arduino via the USB port. You will be using the digital pins and the analog pins for different components and purposes. The analog pins can also be used as digital pins if you ever need to. The microcontroller is the brain of the board. It has a very small amount of memory compared to your computer.

Digital Pins

There are 14 dedicated digital pins. In simple terms, they are either on or off, they are 5 volts (on or HIGH) or 0 volts (off or LOW). You can use them to switch an LED on or off.

USB connector

The USB cable plugs in here, and the other end connects to your computer.

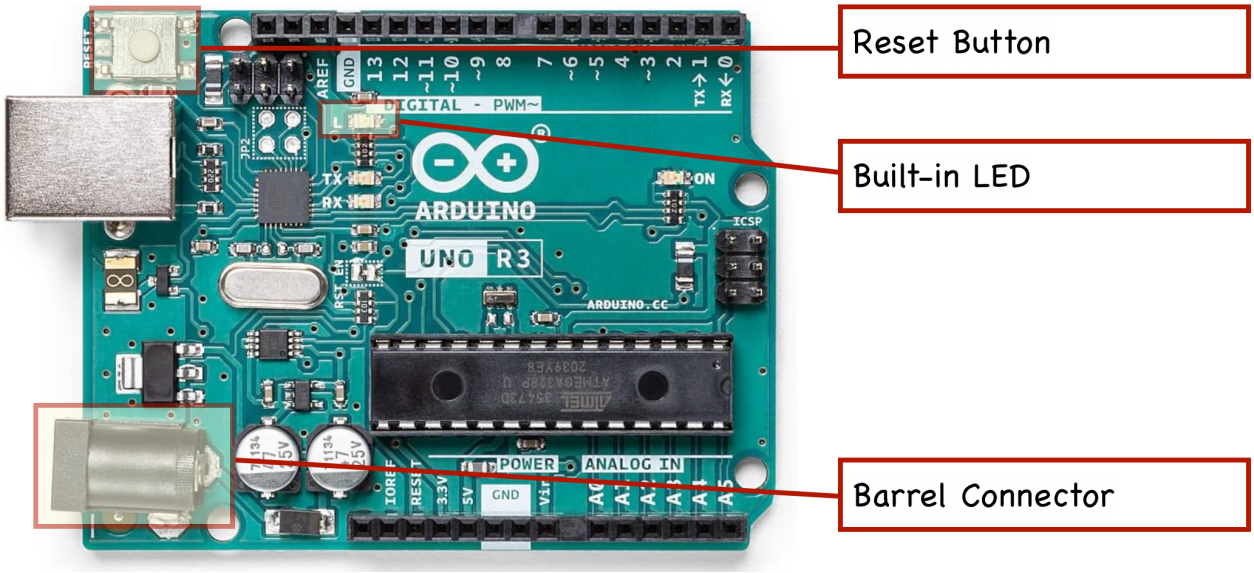
Microcontroller

This is the brain (microchip) of the operation. It operates on 5v and can do lots of useful things.

Analog Pins

These can receive an analog signal (like turning a dial or some other sensor). There are six of them.

Figure 6: more key parts of the Arduino Uno



Here are a couple more features of the board. If you want to power it away from your computer, you can connect a 9-volt battery to the jack socket. Also, if the Arduino has completed a particular task, you can reset it by pressing the RESET button. It doesn't erase the code. Once the code has been uploaded to the Arduino, it stays there until you overwrite it. It stays there even if you remove the power.

Reset button

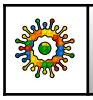
Useful sometimes when you want to start a sequence of events from the start.

Built-in LED (L)

This is connected to pin 13 and can be used as a stand-alone LED if you don't want to connect one to the pins.

Barrel Power Supply

You can use a 9v battery (one of those square ones you get in smoke alarms) to power your Arduino if not connected to the computer. I recommend investing in some rechargeable ones if you plan to use them a lot.



Prototyping

We are not making anything; we are prototyping. This means we are building circuits, using components (sensors or motors), and learning how to use them and code them for a particular purpose. There is no soldering involved because we will use a breadboard to connect the components together.

Eventually, you may well want to build something onto a circuit board; for that, you could solder the components onto a blank board called a PCB (printed circuit board). You may also want to mass-produce it as a gadget, and in which case, you would send your wiring diagram to a specialist company, and they will make it for you.

Although we may stray into making robots (hopefully), we need to start at the beginning and just explore different components and boards and learn to code them.

Software

To write your code and upload it to your Arduino board, you need a bit of software called an IDE, which stands for Integrated Development Environment. The beauty of Arduino is that they provide one for you. They actually provide two versions.

You can do this one of two ways: either download the application from the Arduino website and then run it on your computer, or you can use a web version.

Option 1: the download version.
Option 2: the cloud version.

An **IDE** stands for **Integrated Development Environment**. It's a software application that provides comprehensive facilities to computer programmers for software development. In simpler terms, it's a single program that combines several tools you need to write, compile, and upload code to your Arduino board.

Here's what the Arduino IDE typically includes:

- **Text Editor:** Where you write your code (called "sketches" in the Arduino world).
- **Compiler:** Translates your human-readable code into machine code that the Arduino's microcontroller can understand.
- **Uploader:** A tool that sends the compiled code from your computer to the Arduino board via a USB cable.
- **Serial Monitor:** A built-in terminal that allows you to send data from your computer to the Arduino and receive data back, which is very useful for debugging.
- **Library Manager and Board Manager:** Tools to easily install and manage additional code libraries and support for different Arduino boards.

While the official Arduino IDE is the most common, other IDEs like Visual Studio Code with specific extensions (e.g., PlatformIO or the official Arduino extension) can also be used for more advanced development.

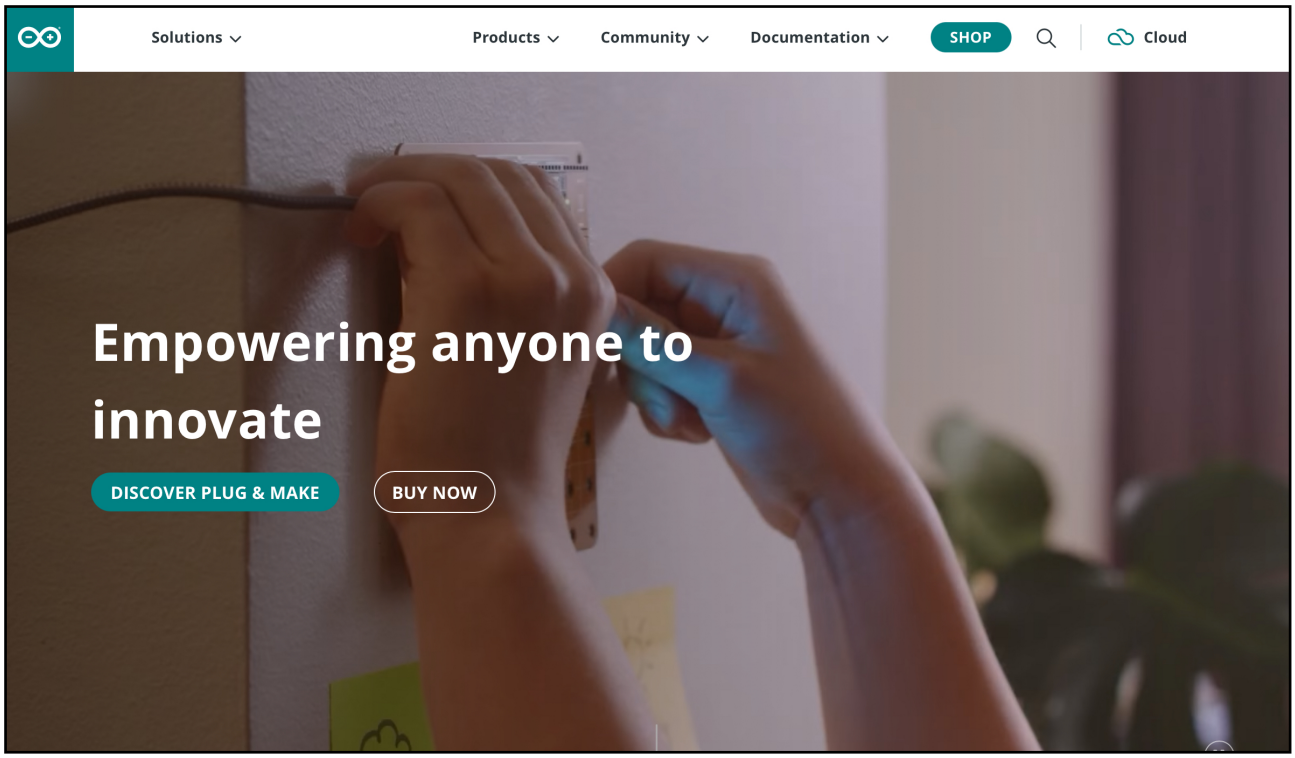
The Arduino Website

To get started, you will need to visit the Arduino website. The link is shown below:

https://www.arduino.cc

You will see the following page. If you click on products, you will get a page displaying all the Arduino boards and the software options. If you have a Chromebook or just want to use the web editor.

Figure 7: Arduino Web Page

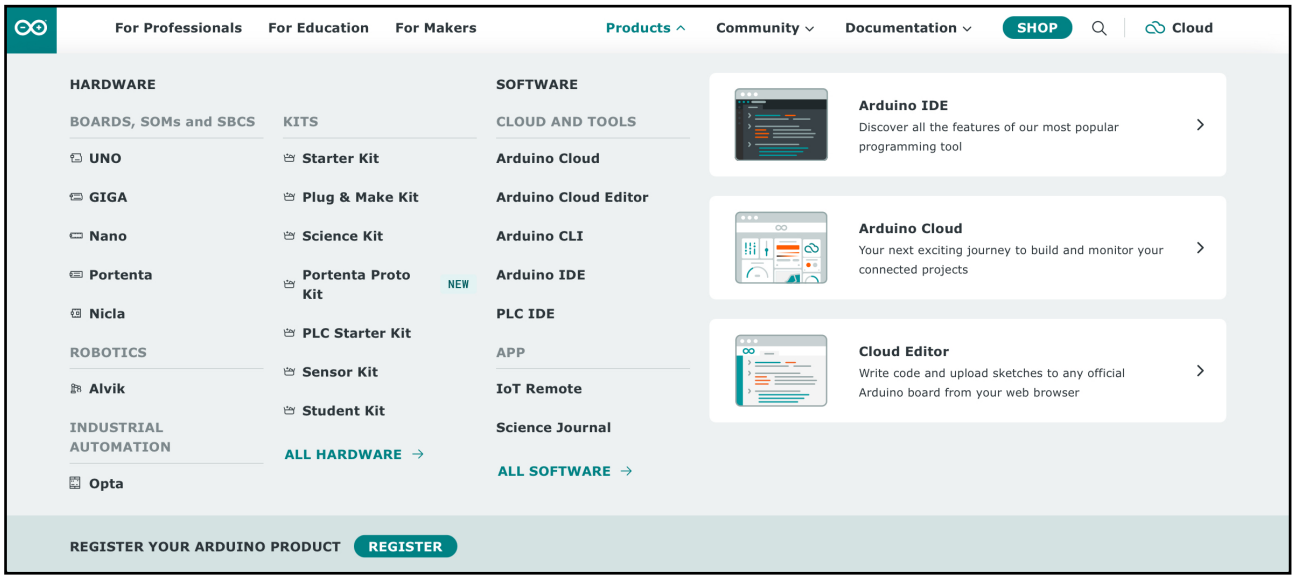


Cloud or download?

You have a number of options:

The Arduino IDE, which you will have to download to your computer, and the Cloud Editor, which you can use from your web browser (but you will need an account). I will cover, briefly, both, but I will be working with the downloaded Arduino IDE for most of this course/tutorial.

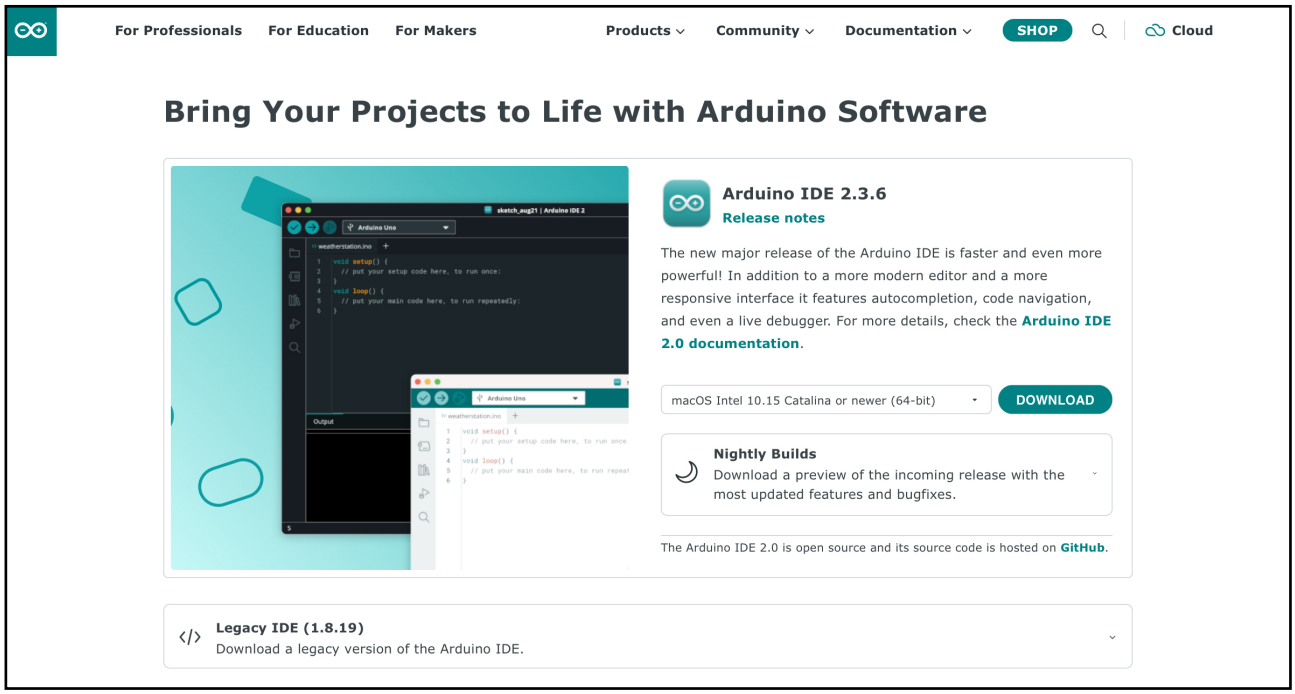
Figure 8: Product Page



The benefit of this is that you can connect your Arduino straight to the application once it has been downloaded. You don't need to log in or anything like that. The drawback is that you will need to update it when necessary and update the boards and the libraries. You may well have to add extra libraries manually. This is not particularly onerous, but you need to bear this in mind. The IDE is free to use, even if they ask for a donation but you have only so many uploads per day.

As you can see, the latest version is version 2, but you can also download the original version 1. The difference is mainly that you can connect your cloud-based sketches in version 2, bringing both together. You have to select the operating system your computer uses: Windows, iOS, or Linux.

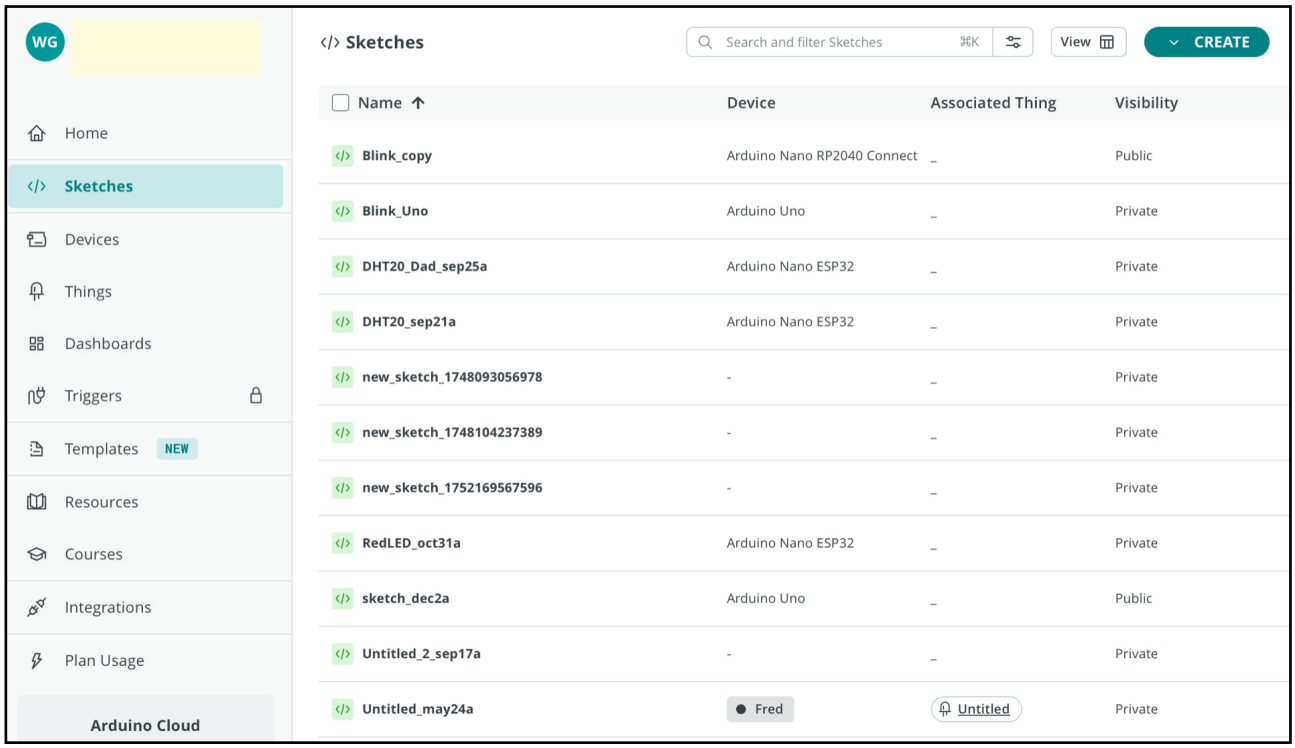
Figure 9: downloading the IDE

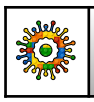


Arduino Cloud

Web-based IDE called Arduino Cloud, which is what you would use if you were exploring the Internet of Things (see tutorial 6). You will need to sign up, but it is free. You can pay for extra features (plans), but to get started, you can use the free version. The benefits of using the web-based version are that it is always updated and the libraries are all there. If you are using a Chromebook, then this makes sense. One drawback is that you have to download and install the agent so that it can communicate with the board.

Figure 10: cloud editor





What other components do you need?

This is a brief list as well as the Arduino Uno:

- Breadboard (get the half size)
- LED traffic light
- Buttons
- Variable resistor (pot)
- Jumper wires

You can get these from almost anywhere, Amazon, Pi Hut or eBay, etc.

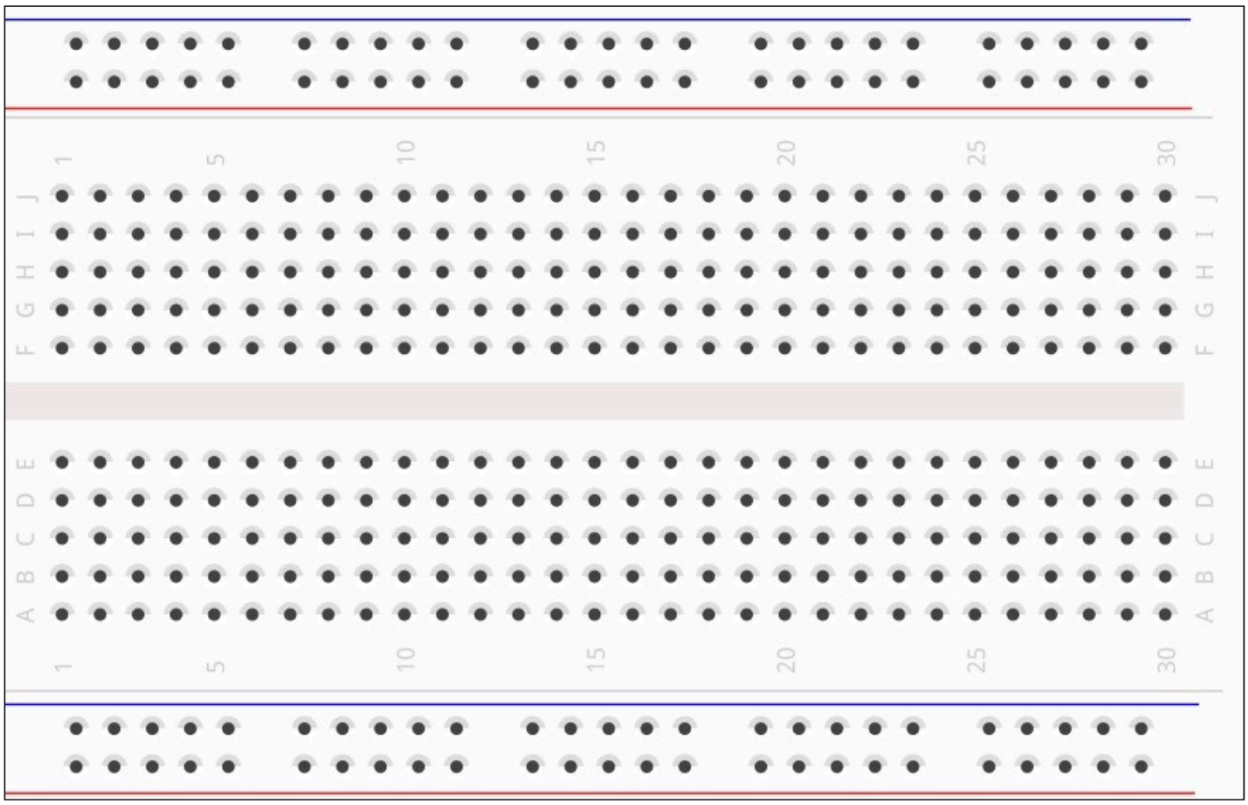
Breadboard

The breadboard is simply a great way to connect components without having to solder anything. There are strips of metal underneath that connect the holes together.

There are many sizes and varieties of breadboards. This one is a half-size breadboard; it has 400 pin points. They can be larger (full size) and smaller (mini). The half-size is perfect for our purposes and most projects. Whichever one you get, they all work on the same principle.

An additional item that is not essential but very useful is to get a base plate, which often is an acrylic base with holes and screws for the Arduino and for the breadboard (half size). It just makes things a bit tidier, especially if you have to move it. Again, this is not essential, but you will see it in the photos.

Figure 11: Half-Sized Breadboard



The ground (GND) rail is connected along the whole length of the breadboard. This means that if you have a number of components all wanting to connect to the ground (GND), you can plug them all into the same one.

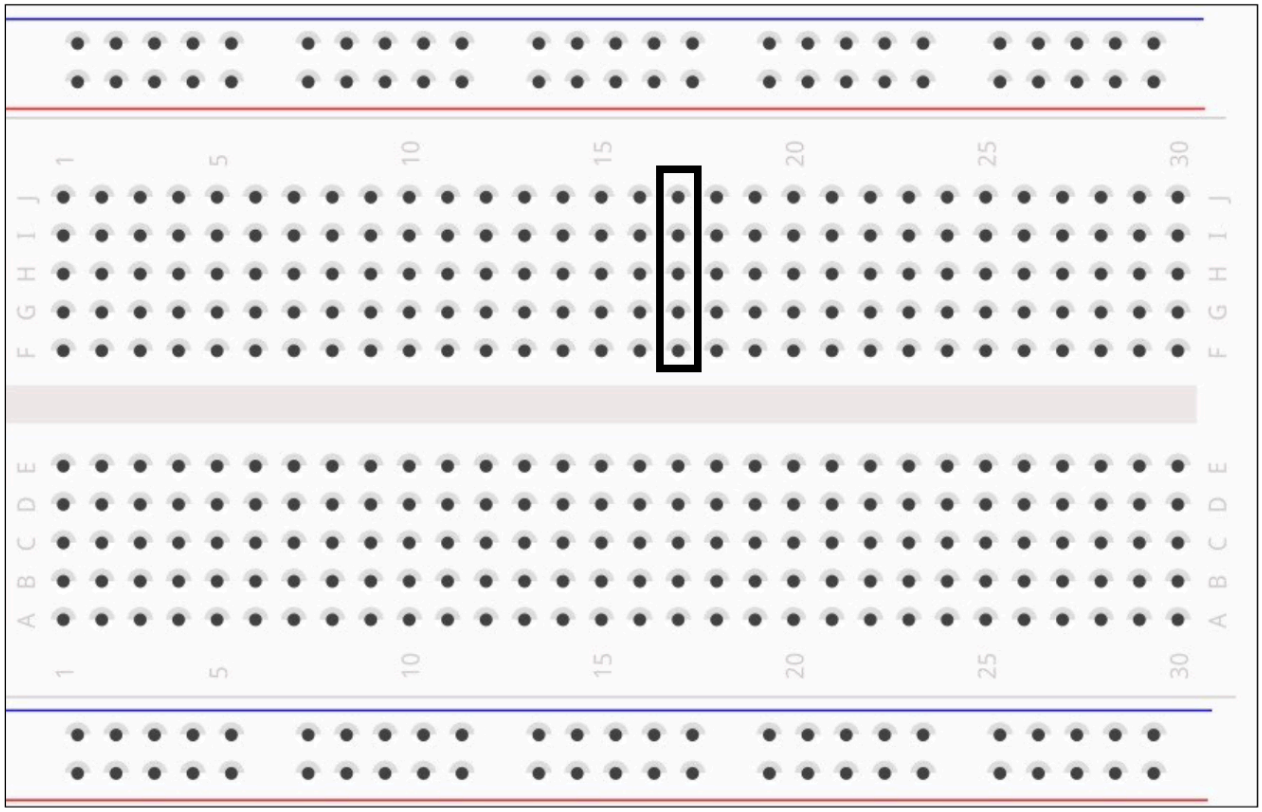
Figure 12: The negative (blue), earth or ground (GND) rail

The positive (+ve or VCC) rail is connected across the entire length of the breadboard, as with the ground (GND). If you have lots of components that want a 5V supply (VCC), then you can use this to supply all of them.

Figure 13: The (red) positive (VCC) rail

The holes are connected across in rows of five, so that anything in that row is connected electrically. Each row of five holes is separate from all the other sets of five holes.

Figure 14: Five connected pin holes



Three LED traffic light module

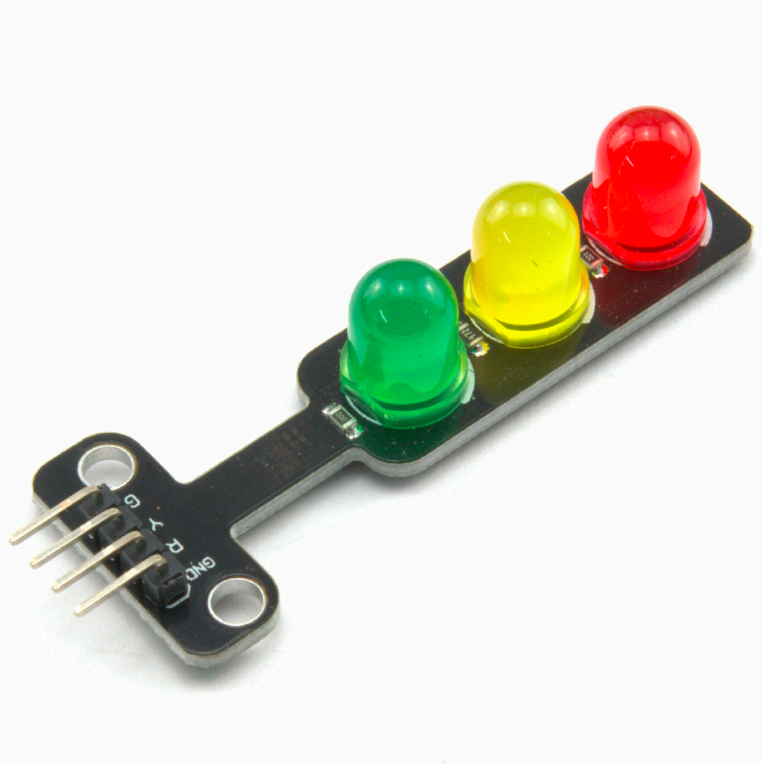
These have three LEDs (and their resistors) built into a module to make life much easier for you. There are four pins that can be pushed into the breadboard.

These are readily available modules and just make life so much easier. Here is a picture of one. It has four pins:

A) one for ground (GND),
B) one for the red LED (R),
C) one for green LED (G),
D) and one for the yellow LED (Y).

The beauty of using this is that you don't have to worry about using resistors. They are already built in and also you have three LEDs to play with and use. We will be using all three later on.

Figure 15: LED traffic light



USB

To make anything work, you will need to connect your Arduino to your computer. This doubles up as the means of transmitting the code and powering the Arduino.

For the Uno, you will need a USB cable that has a USB A to USB B connector. They look like the old printer-type cables. You plug one end into any USB port on your computer/laptop, and the other end into your Arduino Uno.

It has two purposes:

1. To power the Arduino (which runs on 5 volts) so you can use the cable with a normal phone charger or battery.
2. It sends (uploads) the code from the computer to the Arduino.

Figure 16: USB(A)/USB(B)

Button

The button is a tactile, momentary type button, so it only completes the circuit while it is pressed (or breaks the circuit). They come in a variety of sizes (and colours), and mainly they fit on the breadboard, although modules don't. One of the issues is that they will need a resistor in series so that the current isn't too high (this is also the case with LEDs). You can simply place a resistor in series or use a module that has a resistor already built in. A third option is to pull up an internal resistor.

We will be using the button in fig. 18 below and use the pull-up resistor option.

Figure 17: (button module)

Figure 18: (button)

Variable Resistor (Pot)

The variable resistor is also called a potential divider or pot for short. I will call it a pot for now. There are a variety of types of variable resistors and values. The most common is a 10KΩ, which stands for 10,000 Ohms (the measure of the resistance). They are variable by turning a knob from 0 to 10,000 ohms(Ω).

They have three pins: the two outside ones are for the ground (GND) and the 3.3-volt input (VCC) inputs. The middle one is the output voltage, which is based on where the knob is turned from 0 to 3.3 V. We connect the pot to the analog pins on the Nano ESP32.

I will be mainly using the one in fig. 21 shown below.

Figure 19: (pot)

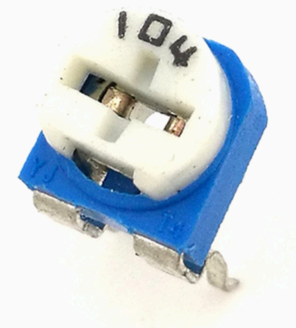


Figure 20: (pot)

Figure 21: (pot)

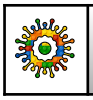
Jumper wires

These are wires widely used in electronics to connect components on the breadboard. They have pins sticking out or holes at the ends of the wires. Get ones that are as long as possible or, better still, get a mixture of lengths. There are three main types:

1. Male-to-Male connectors
2. Male-to-Female connectors
3. Female-to-Female connectors

What you will need are male-to-male, and they usually come in packs of ten, which is plenty. You might also want to get female-to-male and female-to-female while you are at it. They might come in handy for projects down the line.

Figure 22: (male/male jumper leads)



Final words

Although once you get going it is surprisingly easy, getting your head round this, if you are a newcomer, can be a bit daunting. There are plenty of videos on YouTube to help you, and I plan to have some short videos to demonstrate what I mean; keep an eye open for them.

Module A is just to get you started with the minimum of fuss and outlay. At a later date, I will introduce motors, displays, and sensors, but you will have enough to do with these components for the time being.

This will lead nicely onto the other tutorials:

🤖 The Internet of Things (for which you will need an Arduino Nano 33 IoT board)
🤖 The Artificially Intelligent Robotics (where we can connect with our p5.js sketches and ml5.js AI models).