# Hello World!

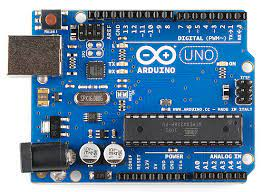
The famous **Hello World!** Is found with the Arduino as well. To get there we need to know four things:

1. What is the Arduino?
2. The Arduino Interface
3. The Arduino IDE
4. Turning an LED on and off - The **Hello World** of Arduino.

## What is the Arduino?

Arduino is an open-source company. It means that all their hardware and software designs are freely available to anyone who wants to copy or change it.

The Arduino microcontroller is an **open-source prototyping platform**. It means it is made so you can easily interface with all sorts of devices such as sensors, motors, lights etc to make something that may or may not be usable. Arduino and the open source community have created all sorts of functions to make it easier for us to use. You can Google about anything about the Arduino.



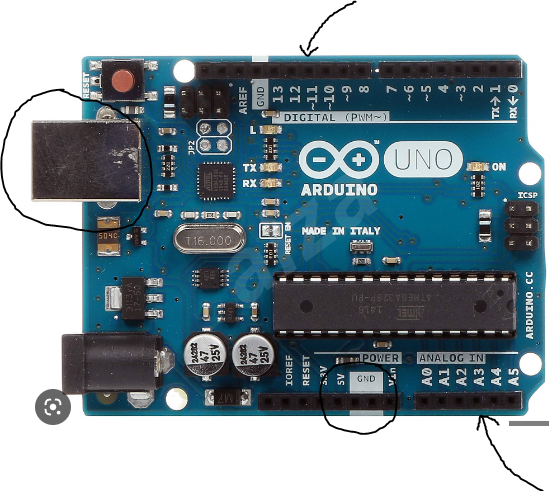
There are **different types of Arduinos** with different amounts of input (for temperature, humidity or other sensors) and output ports (for LEDs, motors etc) and specifications. We normally use the Uno as it is a good size to work with.

There are also lots of clone copies both legal and illegal. The Arduino original is made by the Arduino company and other companies can copy it as long as they follow the rules and/or get permission from Arduino. So the board you see in front of you may not look the same but can do the same thing. The ports should be similar.

But you don’t have to know any of this to get started.

## The Arduino Interface

The first step is to get a feel for the Arduino interface.



On the left top is the USB programming interface. This is a USB-B interface the same as all the old printers.If you connect that to the computer USB port you have 5V available bottom left.

The digital Input and output pins are visible on the top right. We will use one of those for this exercise. The analog inputs bottom right are normally used for analog sensors. We will talk more about these later.

## 

## The Arduino IDE

We have four options to use:

* [Tinkercad Circuits](https://www.tinkercad.com/learn/circuits) are really handy to use as it has a simulation environment.
* [Arduino IDE](https://www.arduino.cc/en/software)
* [Arduino Web Editor](https://docs.arduino.cc/arduino-cloud/getting-started/getting-started-web-editor)
* [Use Arduino with Chromebooks](https://support.arduino.cc/hc/en-us/articles/360016495639-Use-Arduino-with-Chromebook)

I find it easier to teach with Tinkercad Circuit before students move onto using the physical Arduino. This shows the default start position for a new program.There are four areas that are important.



1. Filename. Use descriptive filenames.
2. Tools - Under here you set the port you want to use and the Arduino type
3. The function **setup()** only run once
4. D. The area where your code starts

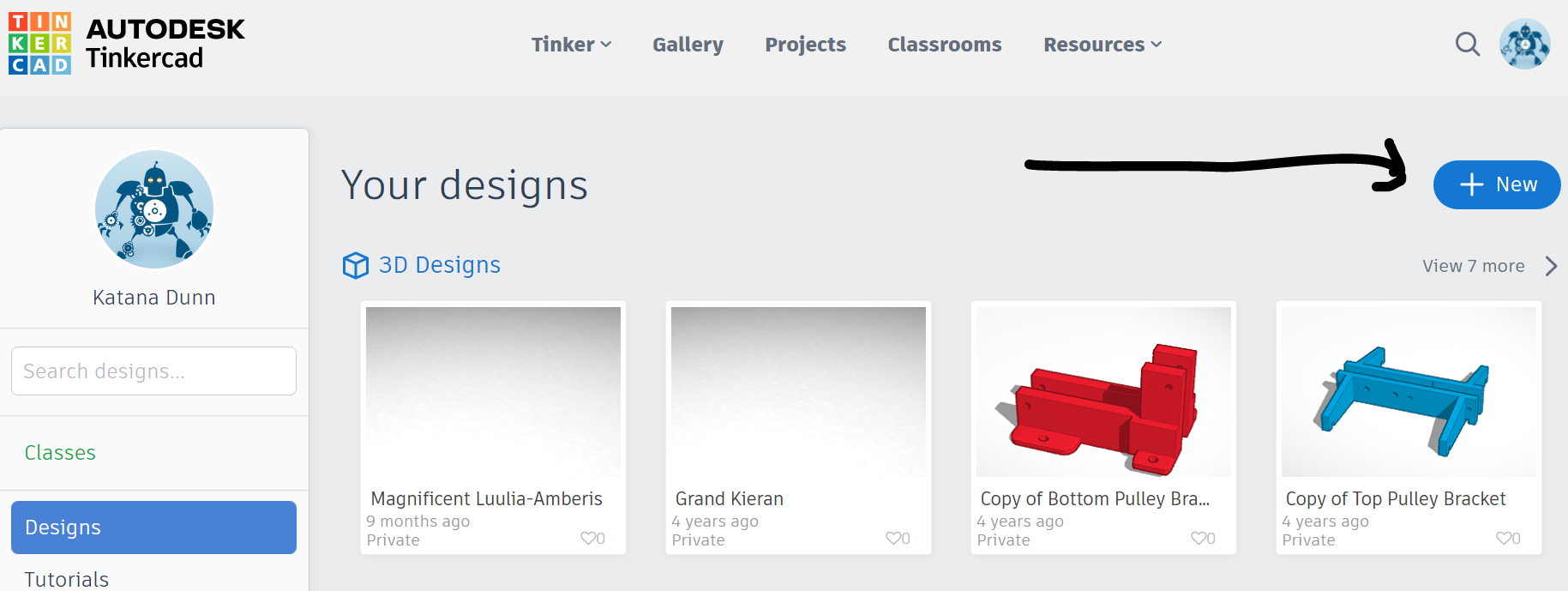
## Flashing LED

This is done in two parts: Setup the hardware and then write your code. We will first simulate it with Tinkercad Circuits and then using the Arduino IDE with a physical LED.

### Tinkercad Circuits

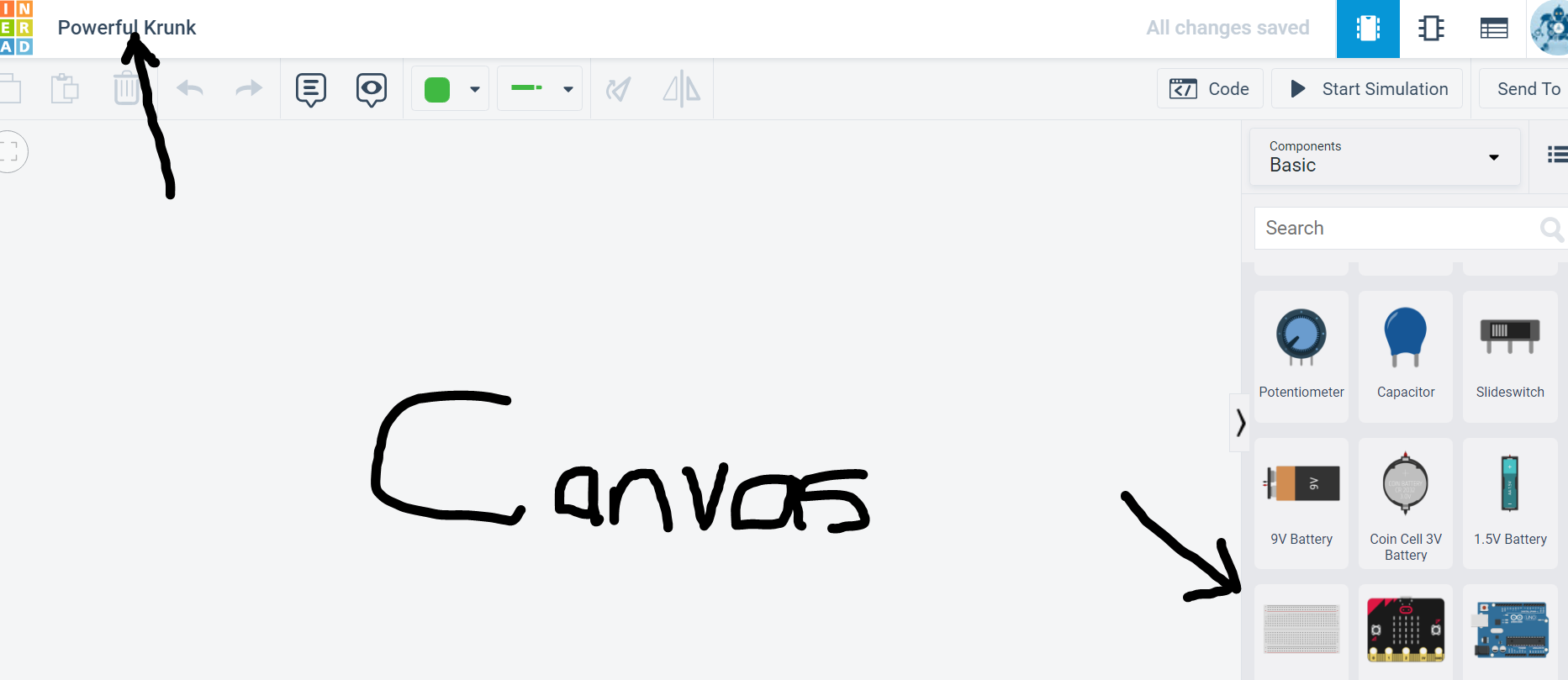
#### Hardware

First task can be done in Tinkercad Circuits. Find this page:

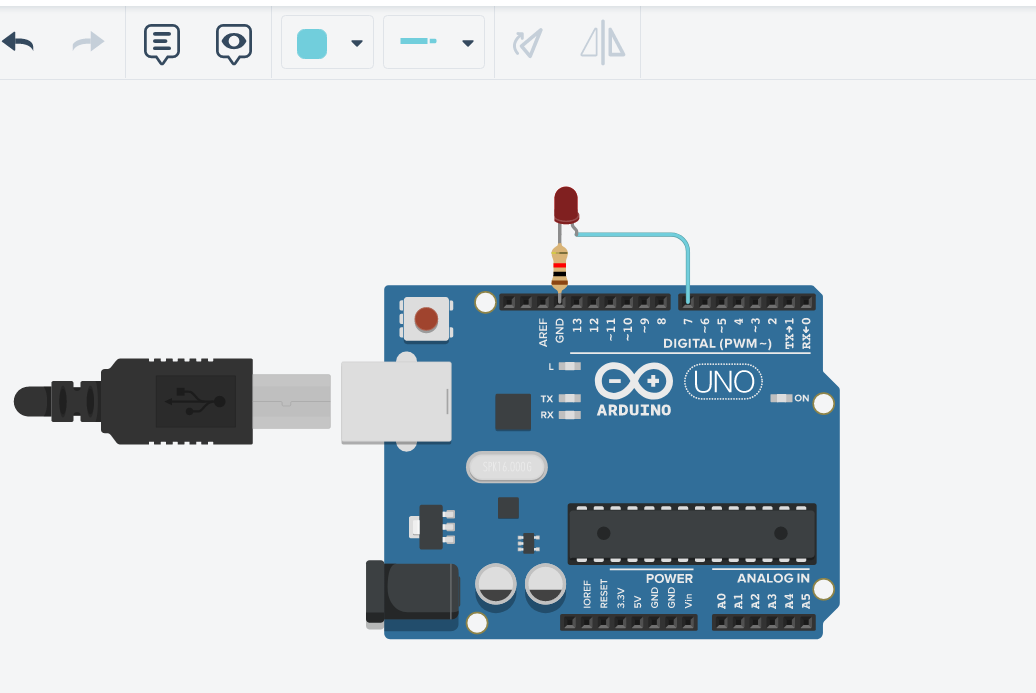


Click New -> Circuit.

Change filename to Flashing LED (as long at it is descriptive). Scroll on the right until you see the Arduino Uno. Click and drag onto canvas

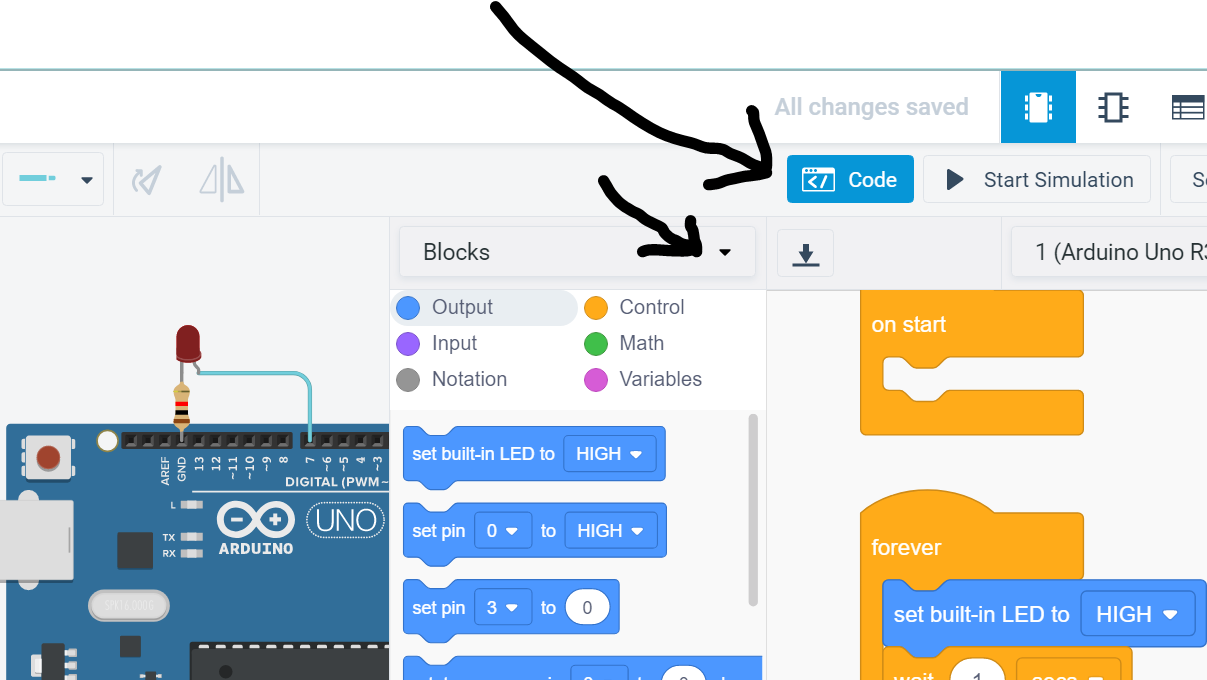


Also find the LED and a resistor (Top of components). Connect with wires as shown. Just click and drag to the next point. Change colour as required depending on convention. Black is ground and red is 5V.

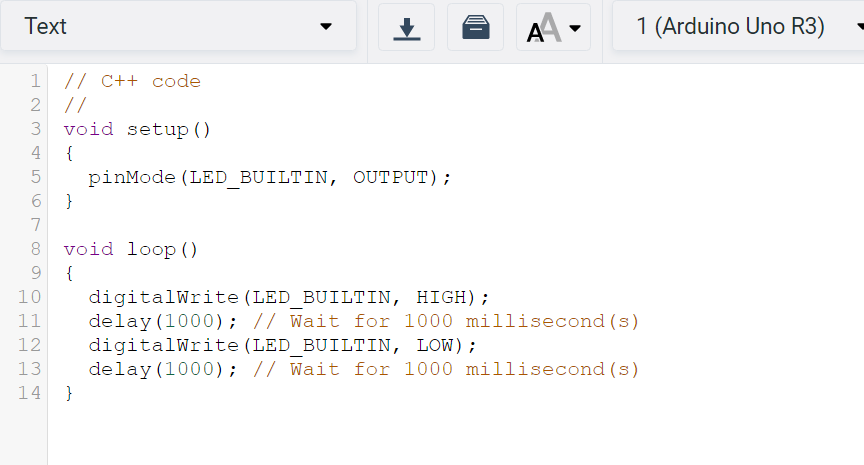


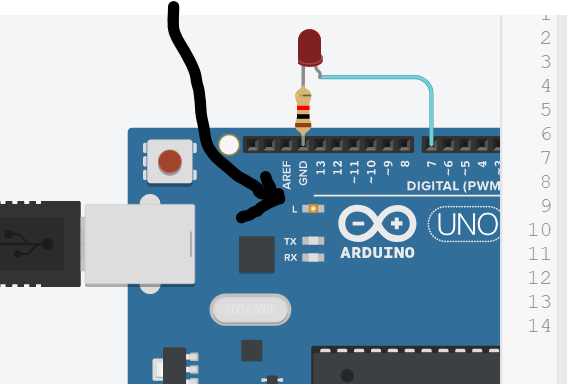
#### Software(Code)

Click on code. Then select Text instead of Block (You can use block for younger students as it means you do not have to worry about syntax as much).



This is the default code that comes up:



Press start simulation and observe LED on Pin 13 flashing. This is the built in LED. LED\_BUILTIN is a constant that is part of Arduino Software.

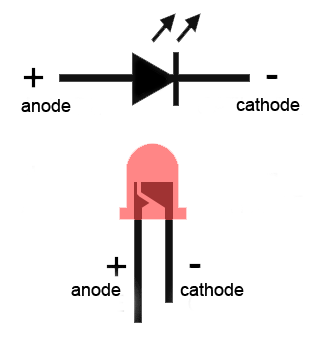
Warning: This code will work without the resistor but you will have a problem with the physical circuit.

Now we can modify the code to use our LED.

### Arduino IDE and Physical LED

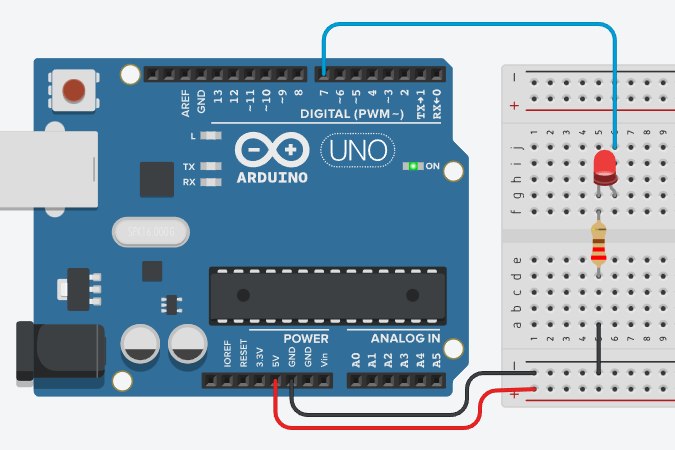
So this is where things become a bit more interesting as you should have an idea about voltage, current and resistance and how it relates to Ohm’s Law. Or we get the appropriate resistor size based on the current using the multimeter in Tinkercad Circuits(Or an actual multimeter).

This is important because the current through an LED should be limited to ensure it does not blow. This current can be controlled or set by a certain resistor value. The LED will also be dimmer or brighter depending on this value. Most LED’s have a maximum rating of about 30mA through it. A good value is about 10mA as the LED will be bright enough. The resistor value available is 220 Ohm.



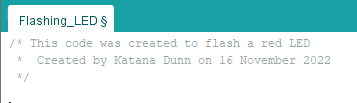
The other important thing is to remember the LED has polarity and it is important that the current flows in the right direction from positive to negative

To make it easier we put the LED and resistor on a breadboard and connect the holes to the Arduino as shown. Hover over the LED legs to see which side is positive



Open the Arduino IDE. File -> New. **Save** the file with the name “Flashing LED” or something else descriptive. The first step should be to write a docstring (Description at the top). It is a good habit to get into and a convention that must be followed later.

Using /\*to add docstring here \*/ can be used for multiple lines.



The next step is to add **variable declarations**. Single line comments only need “//”.



We will create a variable for the LED output port. Arduino has a range of data types. We will use integers here.

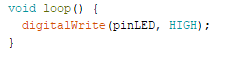




The first function we see is the setup function. It only runs once at the start. We will program the Arduino to set the LED port to an output. A reminder that C is case sensitive and must always have a semicolon at the end of an instruction line.

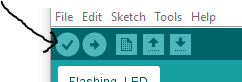
Void means it does not return a value and the empty brackets, (), means we do not pass any values to the function setup.

All we have to do now is set the LED port to high to turn the LED on.



Go to Tools-> Port and select the correct port. Go to Tools -> Board and make sure Arduino Uno is selected.

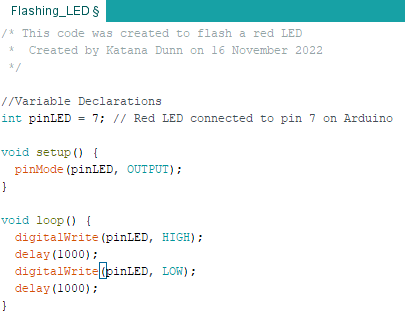
Hover over these buttons:



First is Verify. It basically checks if the code is correct and compiles the code.

Next we download the code to execute it on the Arduino. The LED should turn on.

To make it flash we add a 1 second delay to keep it on before turning it off for 1 second. This is the final code:



Hello World with Arduino!