YTP - Module YTP01

Arduino & Micro-Controllers

Teachers Handbook
LESSON 2
What is an Arduino?

Goals & Success Factors

Lesson Goal

- To understand the question “what a micro-controller?”
- Learn how to setup an Arduino
- Understand Analogue and Digital
- Learn how to create the most basic Arduino project

Lesson Success Factor(s)

- Setup an Arduino successfully
- Complete the Arduino Blink project

Runsheet (Guideline)

00:00 Introduction
00:05 Goals & Success Factors
00:10 What is an Arduino?
00:20 Analogue vs Digital
00:25 Arduino PINs
00:30 Build Blink Circuit
00:45 Code & Test Blink Circuit
01:00 Finish

Extension(s)

- Change the resistor
- Add another LED

Students Pre-work

Watch the following video:
Introduction to Arduino: youtu.be/09zfRaLEasY

Teachers Pre-work

Watch the following video:
Introduction to Arduino: youtu.be/09zfRaLEasY

Student Homework

Watch the following videos:
Recap of Arduino hardware and software: bit.ly/ytp01-led-blink

Knowing your way around the Arduino Editor: bit.ly/ytp01-the-arduino-editor

Handouts (optional)

Blink diagram
Crossword

Please download from:
https://github.com/nexgencodecamp/ytp/tree/master/modules/arduino/02-intro-to-arduino/handouts

Materials Needed

Components for Blink Sketch (enough for 1 between 2):

- Arduino UNOs
- LEDs (5mm)
- Jumper wires (2 per group)
- 1 x resistor (100 – 200 Ohm is fine)
- Breadboards

How you run this is entirely up to you. It really depends on class numbers, experience in the room, time limit and so on. You could for example:

1. Write/laminate some instructions for each student
2. Pair up students
3. Arrange the components into boxes and hand them out with instructions
4. Alternatively – do the exercise together from the projector and build it live
5. Use a set of photos/slides – one for each step and progress these as a class together
6. Show them how to assemble it then ask them to try it without instruction/photos

There are many ways to run this exercise !!

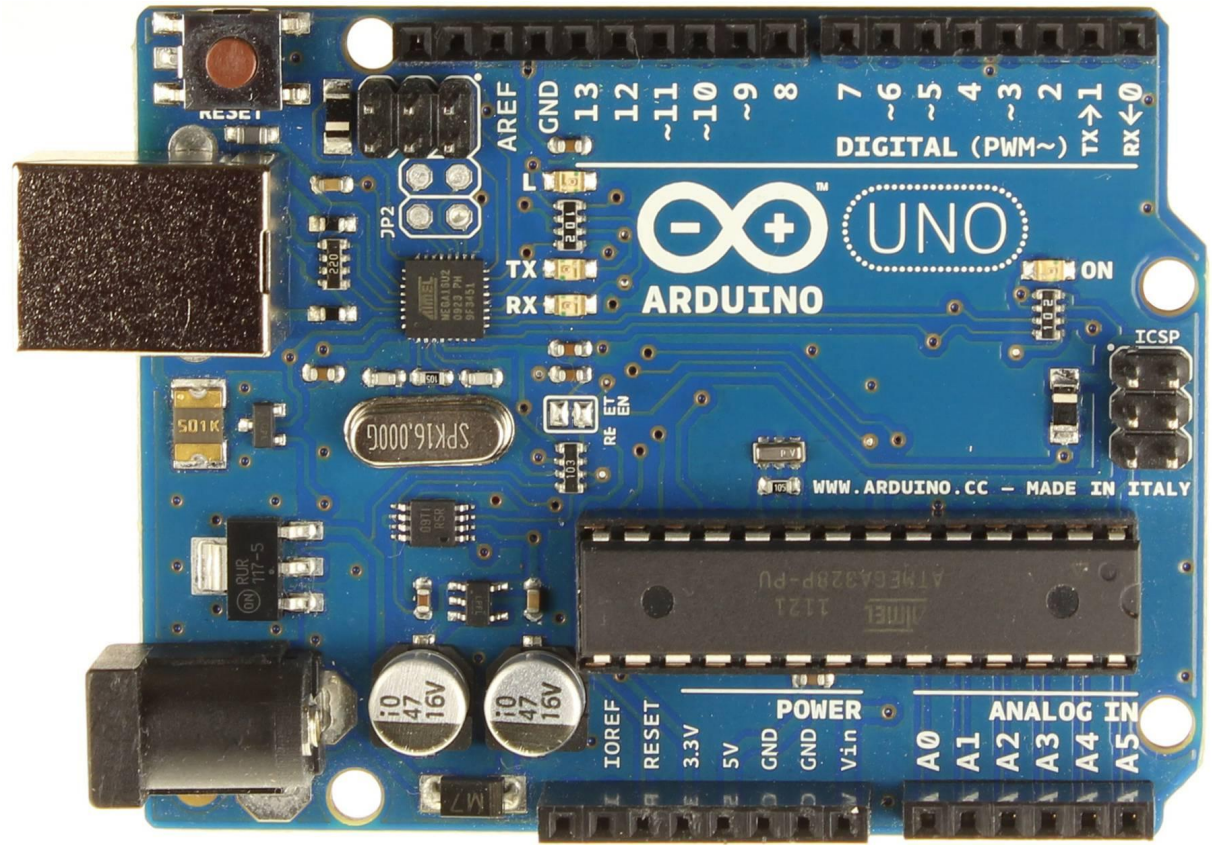
Handbook

Definitions

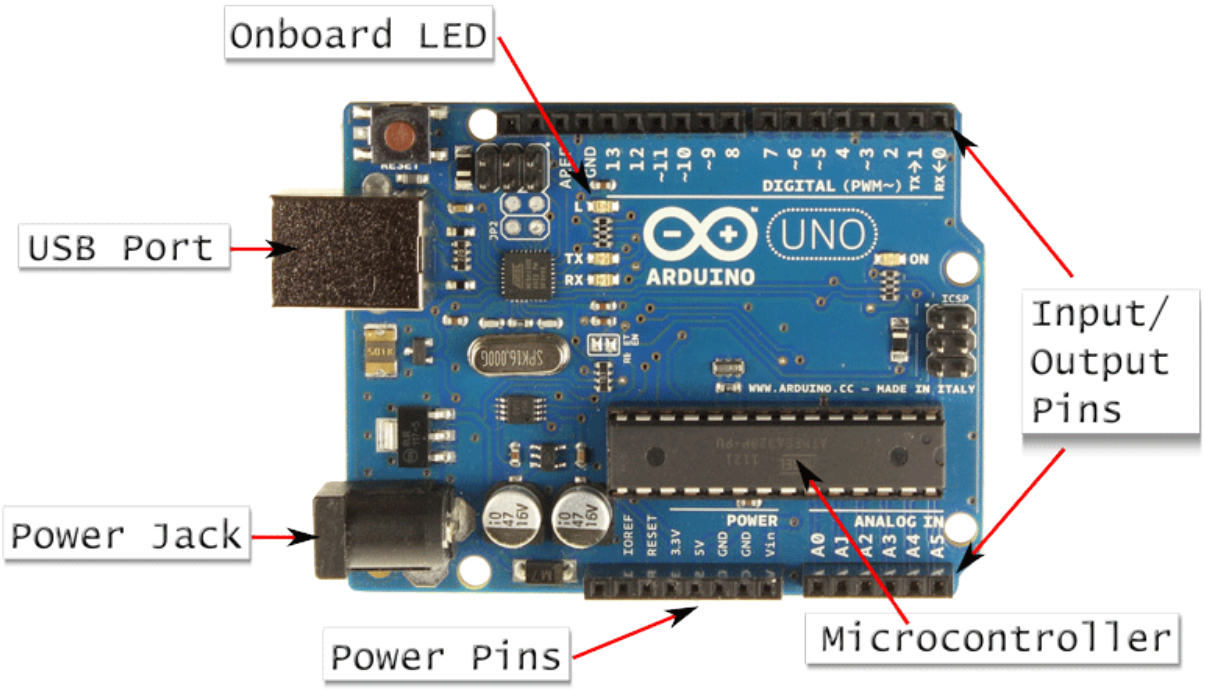
Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.
Source: http://www.arduino.cc/

The Arduino may prove to be one of the most important inventions of the 21st Century. The first micro-controller to bring physical computing to the masses, it has opened up a world of possibilities to all of us. Not just for engineers or coders, the Arduino was always intended for people such as artists, designers, hobbyists and creatives (see above quote)
Source: Pete Januarius - Nexgen Codecamp

What does it look like?



The Arduino UNO



Microcontroller

The brains of the Arduino. This is the chip everything is built on.

USB Port

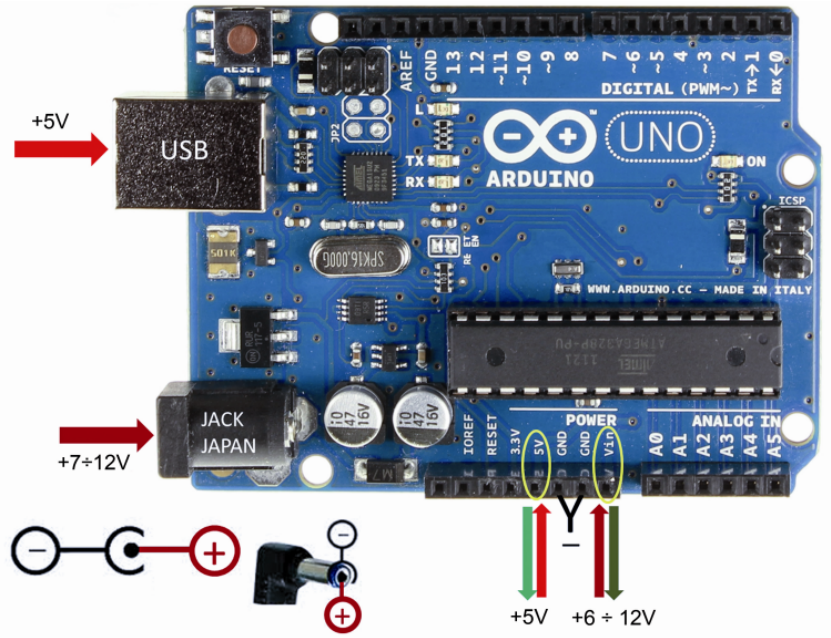
This is where you connect your USB cable

Onboard LED

A helpful Light Emitting Diode that you can use to test your Arduino with the classic Blink sketch (blinking an LED on and off).

Power Jack/Power Pins

Supply power to the Arduino. Look at the following diagram. This gives you a clear idea of what you should use in terms of voltage for each option:



Note that the 5v pin (circled in yellow) is an unregulated supply so you should NOT put more than 5v through this. In actual fact, we recommend that you never power the Arduino from this pin. Instead, if you are using batteries, connect the positive to the VIN pin which is a regulated supply between 6v and 12v and connect the negative to GND.

Do not power from both the VIN and the (barrel) jack at the same time as dangerous conflicts may occur.

Using the USB port will use the power from your computer to power the Arduino. This is fine for development and testing but obviously not when you want to deploy a project.

If you use the 7-12v barrel jack then you MUST use an external source (normally a power supply but can be batteries. You can actually use between 6-20v to power the Arduino but the recommendation from the manufacturer is 7-12v.

Input/Output Pins

Input and Output pins are the mechanism by which we read data in to the Arduino and push data out from the Arduino. Input data comes ‘in’ from a component such as a button or a temperature sensor. In the case of a button, the input data is simply an ‘on’ or ‘off’ value. In the case of the sensor it is a temperature reading (in actual fact probably a voltage reading). The input value is then at your disposal and available to your Arduino program code. Output data is data that originates from your code and is then ‘sent’ to an output pin which could be connected to a display, motor or some other device expecting to be ‘sent’ some values from the Arduino in order that it may be controlled.

How do we use an Arduino?

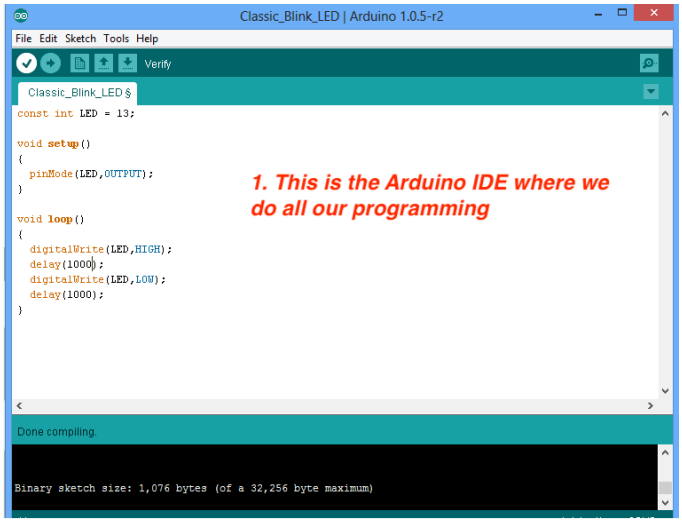
So you might be thinking “What do I actually do with an Arduino and how do I use it?”. Both great questions. There are a few key points to note about Arduinos and micro-controllers in general (Note that an Arduino is just one example of a micro-controller):

1. An Arduino (or a micro-controller for that matter) can be thought of as a simple computer, sort of like an insect brain as opposed to a complex/powerful computer which could be likened to a human brain.
2. It does not do multitasking like an operating system such as Windows. It can only run one program at any one time. If you want to run a new program on an Arduino you must overwrite the previous one. This is what it is built for. Arduinos do one thing and they do it very well.
3. Programs are *uploaded* using the Arduino IDE software (https://www.arduino.cc/en/Main/Software?) which itself is installed on your Win/Mac/Linux machine.
4. You create or write programs for the Arduino in a language called ‘C’ using the Arduino IDE software.
5. The real purpose of an Arduino is to enable us to control or sense the physical environment around us. The Arduino makes it incredibly easy. Here are a few examples of the kinds of things we might do with an Arduino:
    a. Turn an LED (light) on and off
    b. Read the current temperature and humidity
    c. Make a security alarm based on the ability to sense moving objects
    d. Make a simple musical instrument
    e. Make a line following robot
    f. Make a remote control using IR (Infra red)
    g. Sense the composition of gases
    h. Turn a motor
    i. Control one or more servos (motors)
    j. Make a simple or complex robot
    k. Make a traffic light
    l. Make an autonomous robot
    m. Control an LCD display
    n. Test the moisture level in the soil
    o. Make a morse code signaller
    p. Communicate via bluetooth
    q. Test the air pressure
    r. Make your home automated
    s. Light up LEDs as Christmas lights via Twitter (I’ve done this)
    t. Build a game controller
    u. Build a remote controlled car
    v. Build a light up doggy collar for those night walkies
    w. Build a food dispenser for your pets

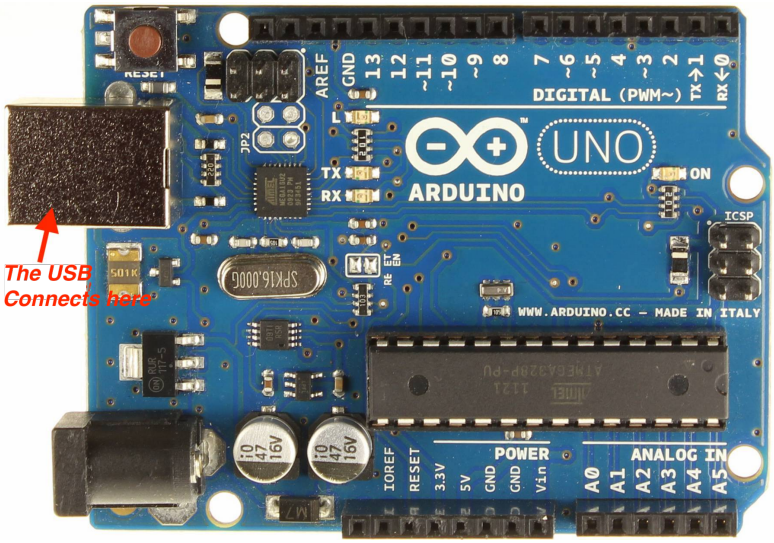
x. Make light up shoes that sense the steps you take
y. Make an LED light up T-shirt based on temperature
z. Make a LCD alarm clock

We are convinced that given a bit of time and some thought, anyone can not only come up with their own project, but they could build it pretty easily too!

Just to help you piece together the picture a bit more, here is the general setup that you would employ if you were using an Arduino - in other words - if you were writing some program for your Arduino that you wanted to upload:



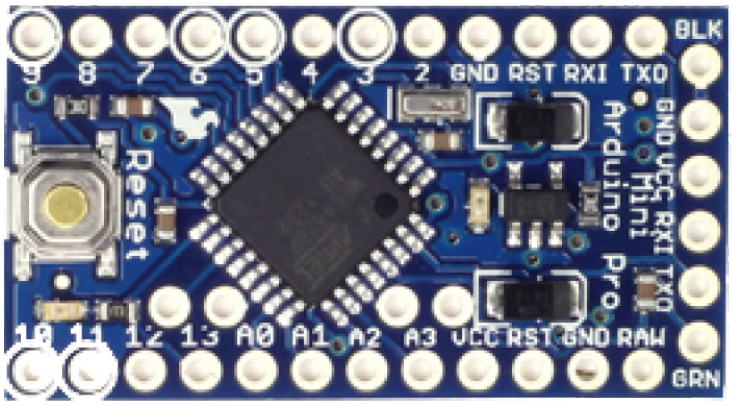
2. This is the USB cable that connects our computer to our Arduino. If we didn't have this, we would have no way of uploading our programs from the IDE to the Arduino.


The USB
Connects here
DIGITAL (PWM~)
ARDUINO
UNO
ON
ICSP
WWW.ARDUINO.CC – MADE IN ITALY
POWER
ANALOG IN

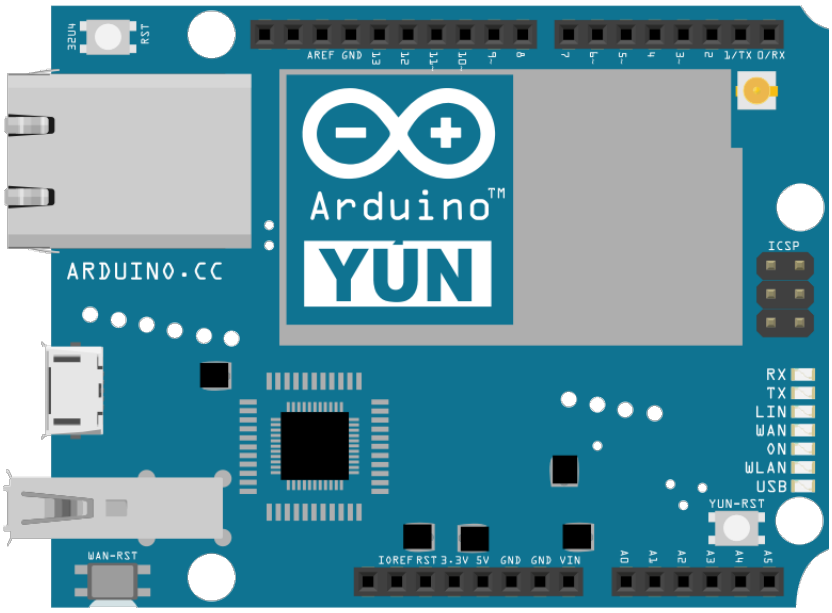
Arduino Form Factors

Arduino is based on an open source design which means that anyone can take the technical specification and create a product of their own design. Hence there are many different types of Arduino out there - we call these different form-factors. The one we will be using is the Arduino UNO. Generally speaking, this is the easiest one to use when you are starting out with micro-controllers. The pin headers make it very easy to connect circuits together.

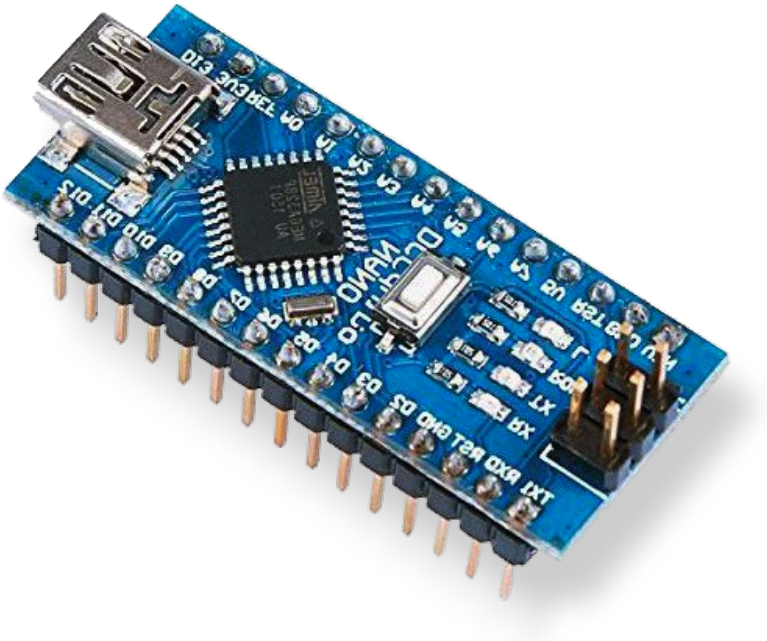
Arduino Pro-Mini



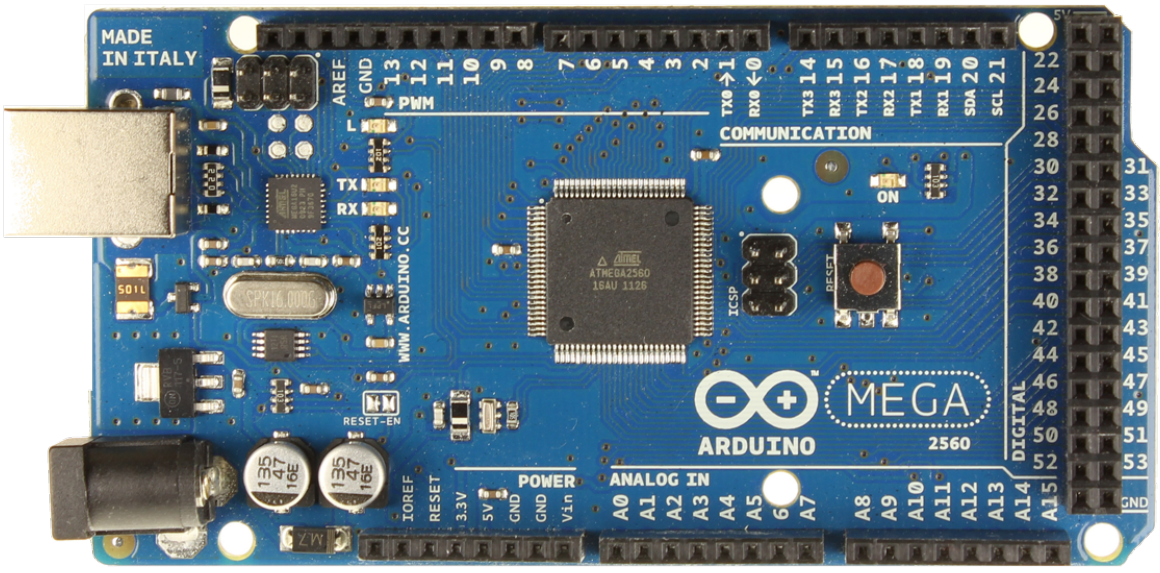
Arduino YUN



Arduino Nano



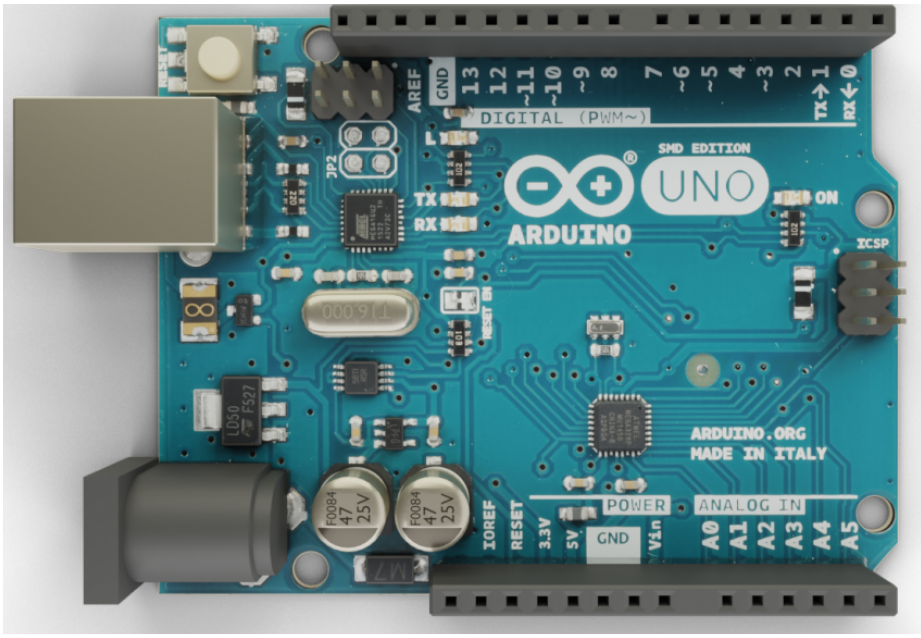
Arduino Mega



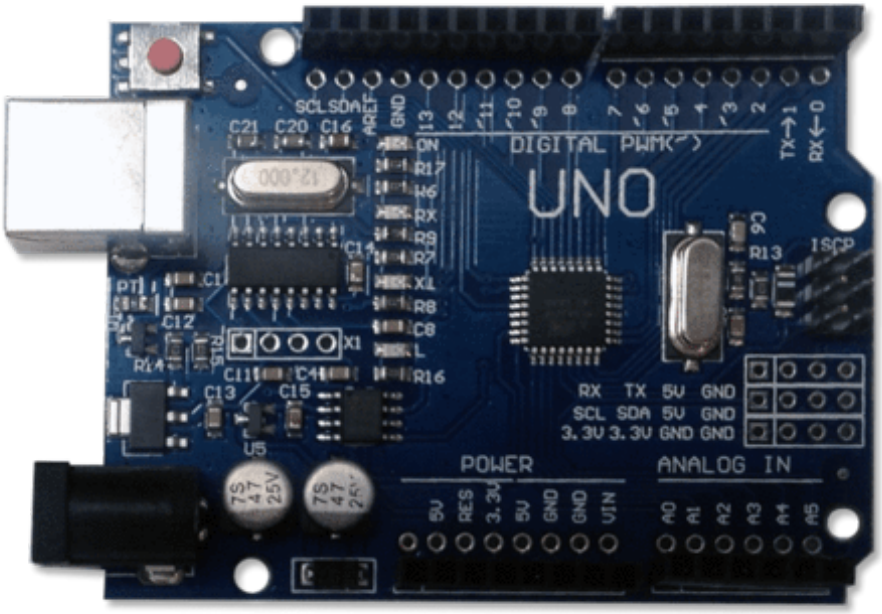
Arduino Clones

There are also many clones out there on the market. In other words you could purchase a genuine Arduino UNO or you could purchase a cheap Chinese clone. These are absolutely fine although they use different software drivers based on the CH340 chip which must be taken into account if you buy these. Drivers for clones are hard to find (see resources).

Genuine Arduino UNO



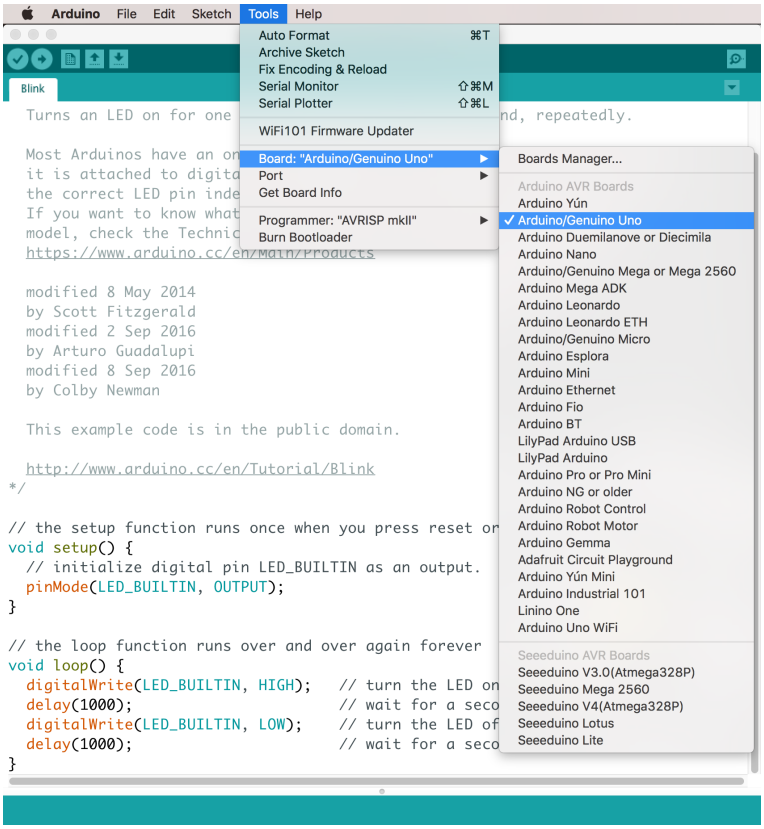
Arduino Clone



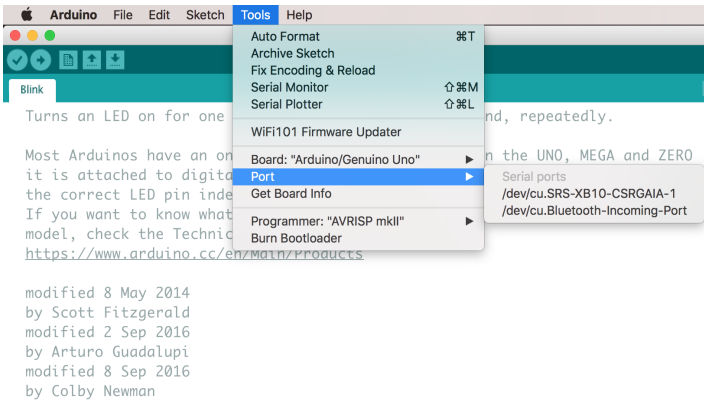
Step by Step Arduino Setup and use

What follows is a simple step by step guide of what you will do 90% of the time when using your Arduino to code and upload a program.

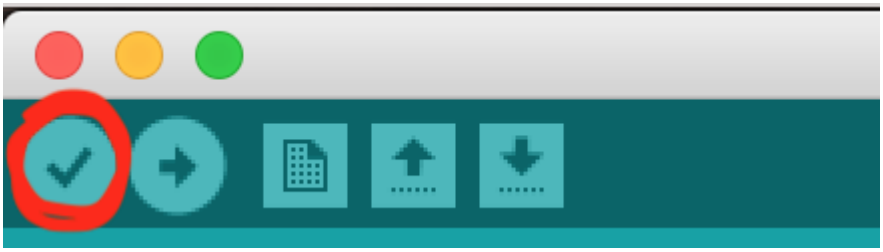
1. Boot up your PC/Mac
2. Open the Arduino IDE application
3. Plug the USB cable into the Arduino and your PC
4. Select the correct board



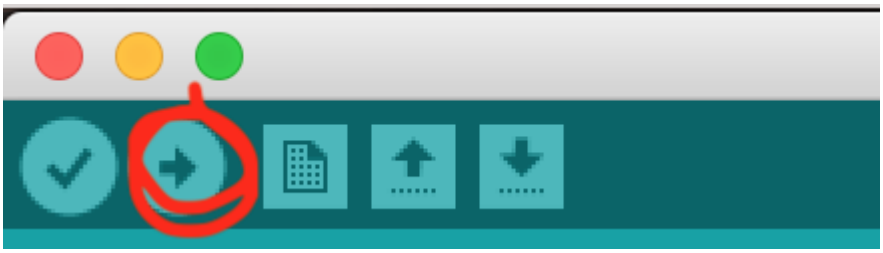
5. Select the correct port



6. Your Arduino is ready!
7. Next steps….
8. Write your code in the Arduino IDE application (see next section)
9. Verify/Compile the code



10. Upload the code



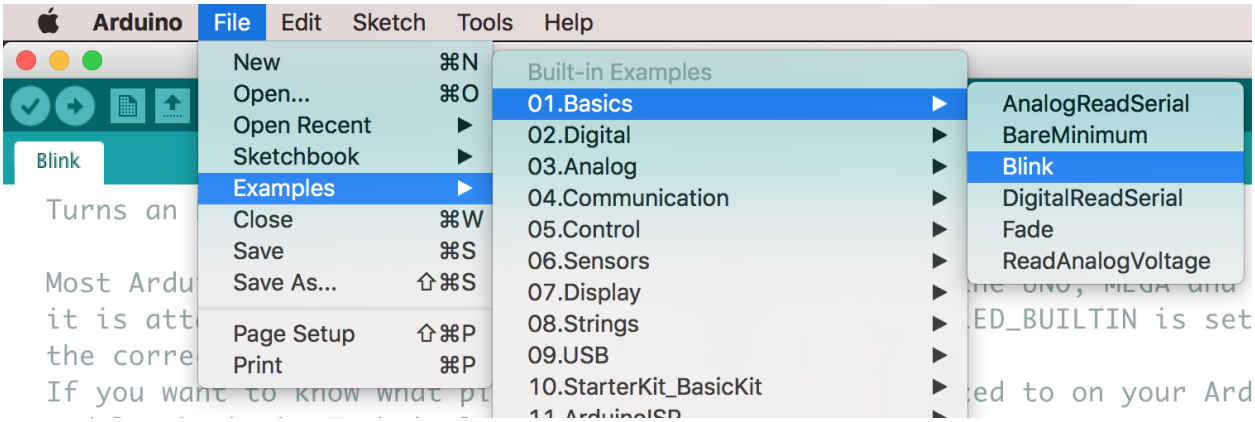
A Very Quick and Easy Example

The canonical example of every single programming language in the World bar none (as far as I know) is ‘Hello World’. I remember doing this aged 13 on my VIC-20 in BASIC in the early 1980s. Basically (excuse the pun) you print out the words ‘Hello World’ to the screen. In the Arduino version, we blink an LED on and off. So firstly you will need the following parts:

1. An Arduino
2. A USB cable
3. Arduino IDE installed on your laptop/computer be it Mac or Windows

“Where is the LED?” I hear you say! Well - we will be using the onboard LED which is internally connected to pin 13. We will configure pin 13 to be an output pin because we will be sending some output data to it ie. an on or off value (one or zero) every 1 second.

Open up the Blink sketch. You will find it in: File -> Examples -> 01.Basics -> Blink



You will see the following code:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Let's take this one step at a time…

setup()

setup() is one of two *functions* that belong to every Arduino program. Firstly a function is essentially an action that a program can take. In the case of 'setup', it is actioning the configuration of any pins, declaring variables (placeholders or buckets for data values) and anything else that it might need to do before the main program runs. It is kind of like an initialisation routine that any electronic device goes through when you first switch it on.

loop()

The second function is loop(). This is very straightforward. It essentially runs the code inside the function and then repeats ad infinitum… until you power your Arduino off. The code inside the loop() function above turns the LED on, waits 1 second, turns the LED off, waits 1 second and is then finished. The loop then repeats….the LED turns on, waits, turns off, waits…..and so on.

Setting the pin mode

Remember that we said that pins can be used for input or output? The pinMode instruction determines which it is. So for example in the code above we are setting the inbuilt LED to be an output pin so that we can 'send' data to it, the data being a high or a low value which translates to a high voltage (turning the LED light on) or a low voltage of zero (turning the LED off):

```
pinMode(LED_BUILTIN, OUTPUT);
```

This could also have been written:

```
pinMode(13, OUTPUT);
```

As LED_BUILTIN is basically a constant value equal to 13.

In this particular example we are not receiving input from any device, button or sensor so we do not need to configure any pins for input.

Writing to a pin

An output pin is only useful if we can write a value to it. We do that with this instruction:

```
digitalWrite(LED_BUILTIN, HIGH);
```

HIGH is a constant representing the number 1. You will also see LOW which you can read as 0. Note that we do not have a need to read from any pins. If we were to do so we would use the instruction: `digitalRead`.

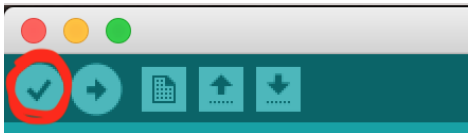
Waiting…

The code delay(1000) waits for one thousand milliseconds.

And that's it !! Well not quite….

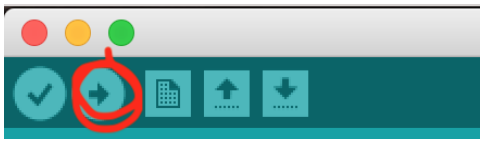
Verify the Code

As described in the previous section, you verify your code by clicking the button in the toolbar shown below. Do this now. The code should compile because it is one of the standard examples and not something that we have written ourselves.



Upload the Code

Once verified, click the button shown below. This physically transfers the code to the Arduino. The code will be stored on the Arduino until it is overwritten by another program.



Analogue & Digital

So there is one concept that we haven't covered yet that is crucial to understanding electronics. The concept of analogue and digital signals. You know what these are, you just may have trouble articulating the definitions.

A Digital signal can have one of two values: on or off, 1 or 0, true or false, yes or no.

An Analogue signal is continuous - in other words it is 0, 1 and potentially every value in between.

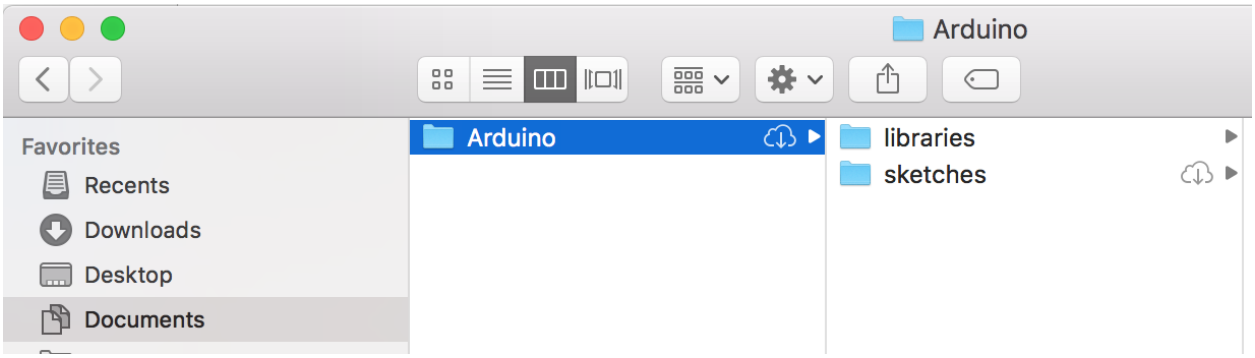
An oldie but a goodie: https://youtu.be/-43VsMXz7U0
Here is a more modern explanation: https://youtu.be/btgAUdbj85E

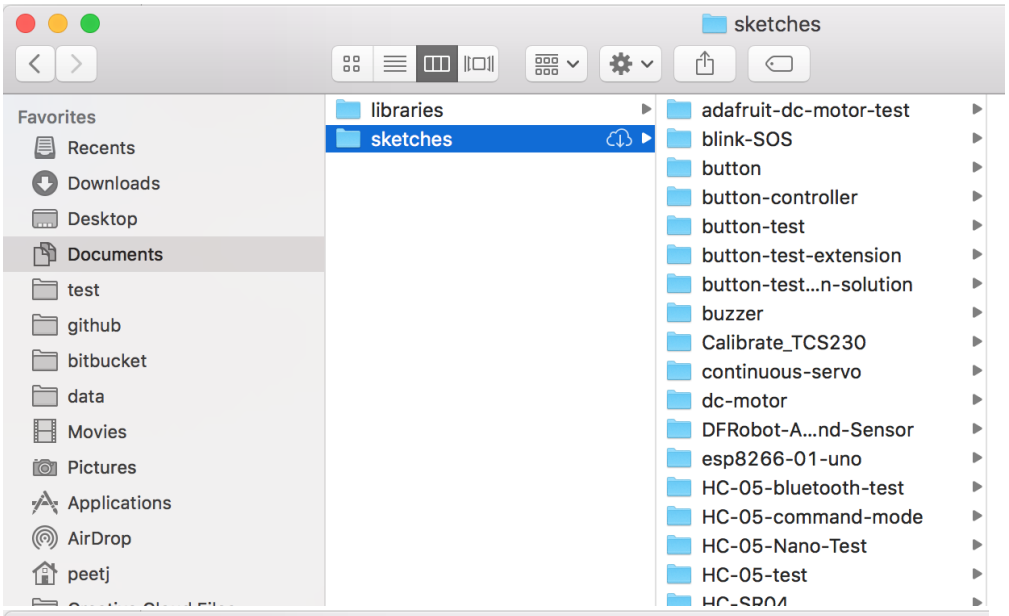
Arduino Folders and File Locations

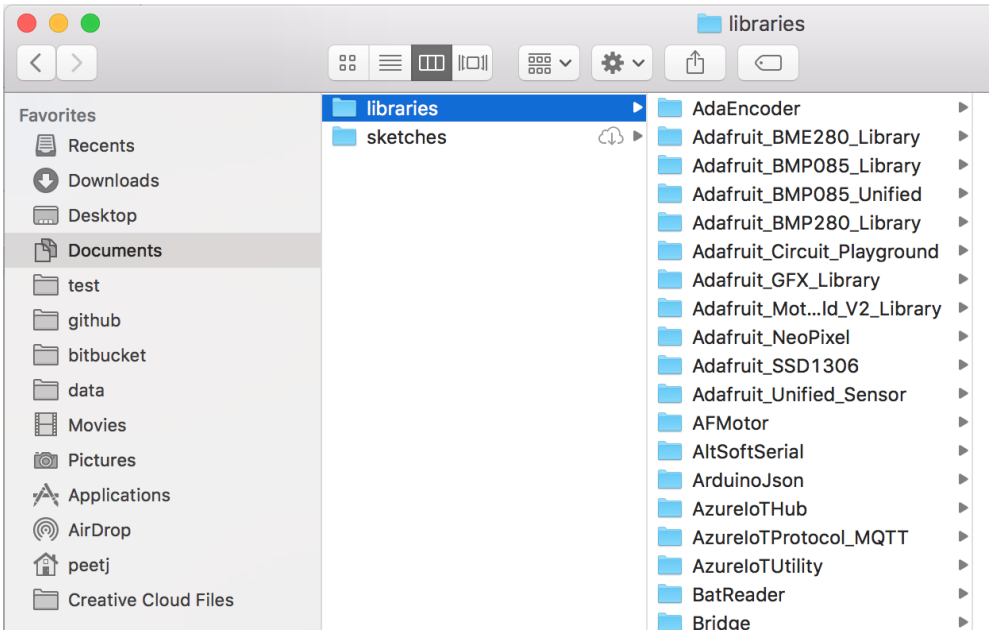
It is definitely worth knowing a bit more about where the Arduino stores all of its files. This will help you when you need to install libraries or move sketches (sketch is the Arduino term for a computer program). Our examples are taken from a Mac but Windows applies also.

When you install the Arduino IDE, it creates an 'Arduino' directory (folder) in your Documents directory. Inside the Arduino folder you will have the following directories (sometimes these are not created but you can go ahead and create them):

libraries - contains third party library code
sketches - contains your sketches (programs)

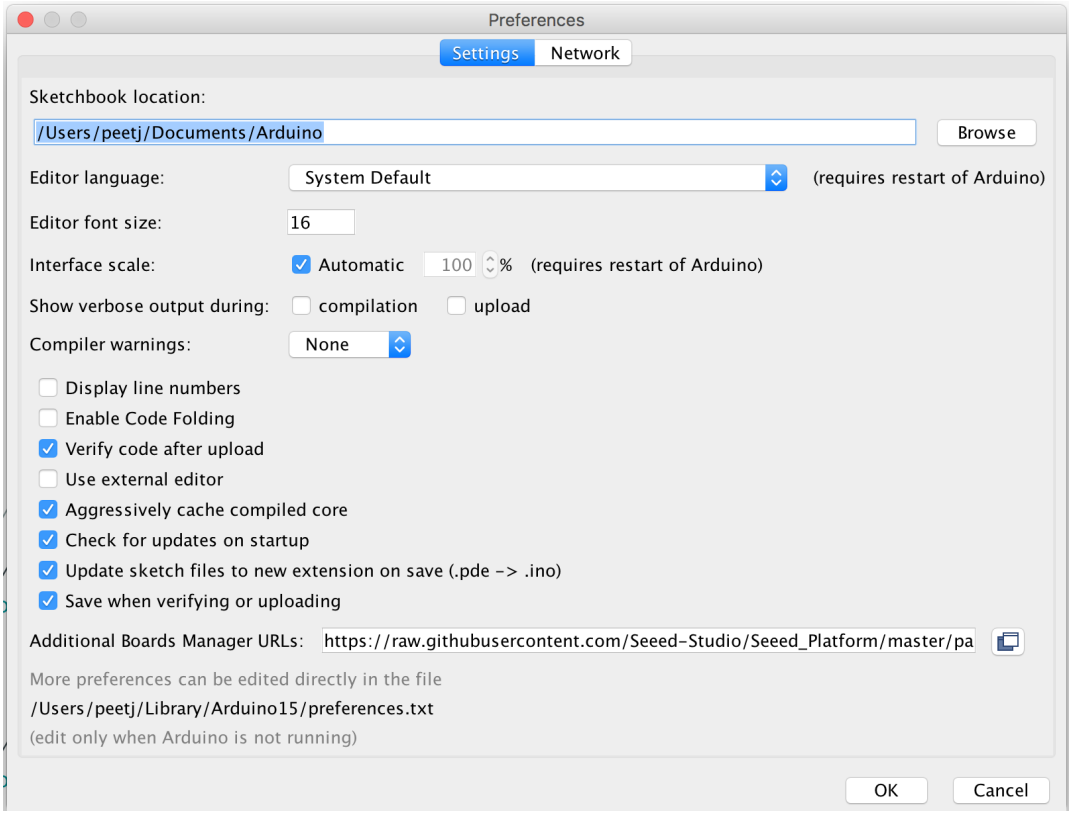

Arduino
Favorites
Recents
Downloads
Desktop
Documents
Arduino
libraries
sketches


sketches
Favorites
Recents
Downloads
Desktop
Documents
test
github
bitbucket
data
Movies
Pictures
Applications
AirDrop
peetj
libraries
sketches
adafruit-dc-motor-test
blink-SOS
button
button-controller
button-test
button-test-extension
button-test...n-solution
buzzer
Calibrate_TCS230
continuous-servo
dc-motor
DFRobot-A...nd-Sensor
esp8266-01-uno
HC-05-bluetooth-test
HC-05-command-mode
HC-05-Nano-Test
HC-05-test


libraries
Favorites
Recents
Downloads
Desktop
Documents
test
github
bitbucket
data
Movies
Pictures
Applications
AirDrop
peetj
Creative Cloud Files
libraries
sketches
AdaEncoder
Adafruit_BME280_Library
Adafruit_BMP085_Library
Adafruit_BMP085_Unified
Adafruit_BMP280_Library
Adafruit_Circuit_Playground
Adafruit_GFX_Library
Adafruit_Mot...ld_V2_Library
Adafruit_NeoPixel
Adafruit_SSD1306
Adafruit_Unified_Sensor
AFMotor
AltSoftSerial
ArduinoJson
AzureIoTHub
AzureIoTProtocol_MQTT
AzureIoTUtility
BatReader
Bridge

Arduino Settings

Open your Arduino Preferences... menu option:



Note the Sketchbook location which you can access from inside the Arduino IDE from the File -> Sketchbook menu option.

Note also that you can change the font size of the text in the code window.

What happens when it all goes wrong?

The following issues are the most common issues that we have seen from plugging in Arduinos thousands of times. This is by no means an exhaustive list. Instead we have listed a more comprehensive list in the Resources section at the end of this lesson. The issues below are the ones we believe to be the ones you are most likely to see in practice:

I cannot see the correct port when I plug in my USB cable
Pull the cable out and reconnect. If this doesn't work, try another USB port. If this still doesn't work, restart the IDE and reconnect the cable.

When I try and upload my code, it won't upload
Is the USB cable connected properly? Yes? Check the Tools -> Port to make sure the port is selected. If all else fails, shut down the IDE, pull your cable out and reconnect

The port is disabled when I go to select it (greyed out)
Sometimes it takes a few seconds to show up, however if it does not show up at all then you probably haven't installed the necessary drivers. With genuine Arduinos, the drivers are installed with the IDE. With clones, you must download the CH340 drivers (see the resources section at the end).

My code won't compile
You more than likely have a syntax error. Have a look at the error message in the editor console. It usually tries to tell you where the mistake is and what it was. Usually for beginners, the mistake is a missing bracket or semi-colon. Remember in code we typically use the full range of bracket types: (), {} and [].

Oops - I didn't mean to press that key combination
Ok so you made a mistake in the Editor? It happens. Just use this key combination to get out of trouble: CTRL-Z

Where do I install third party libraries?
Easy - in your Documents/Arduino/libraries directory (folder). If the libraries folder does not exist, go ahead and create it.

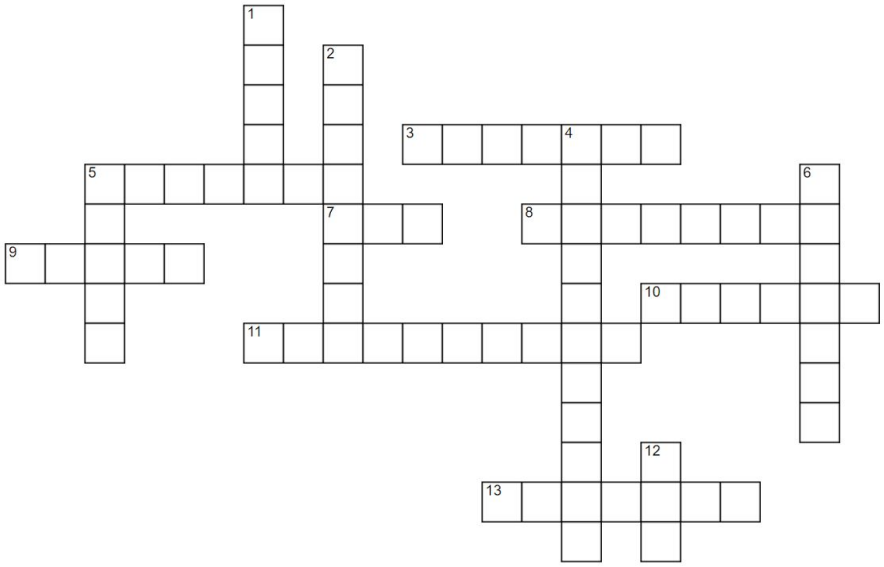
Handouts

Handouts are optional but encouraged. They are normally useful for homework but in some cases you may deem them useful for pre-work depending on the upcoming session content. The handouts are shown here but are alternatively available for download at:

https://github.com/nexgencodecamp/ytp/tree/master/modules/arduino/02-intro-to-arduino/handouts

Name: _____ Date:_____

Arduino and component Crossword



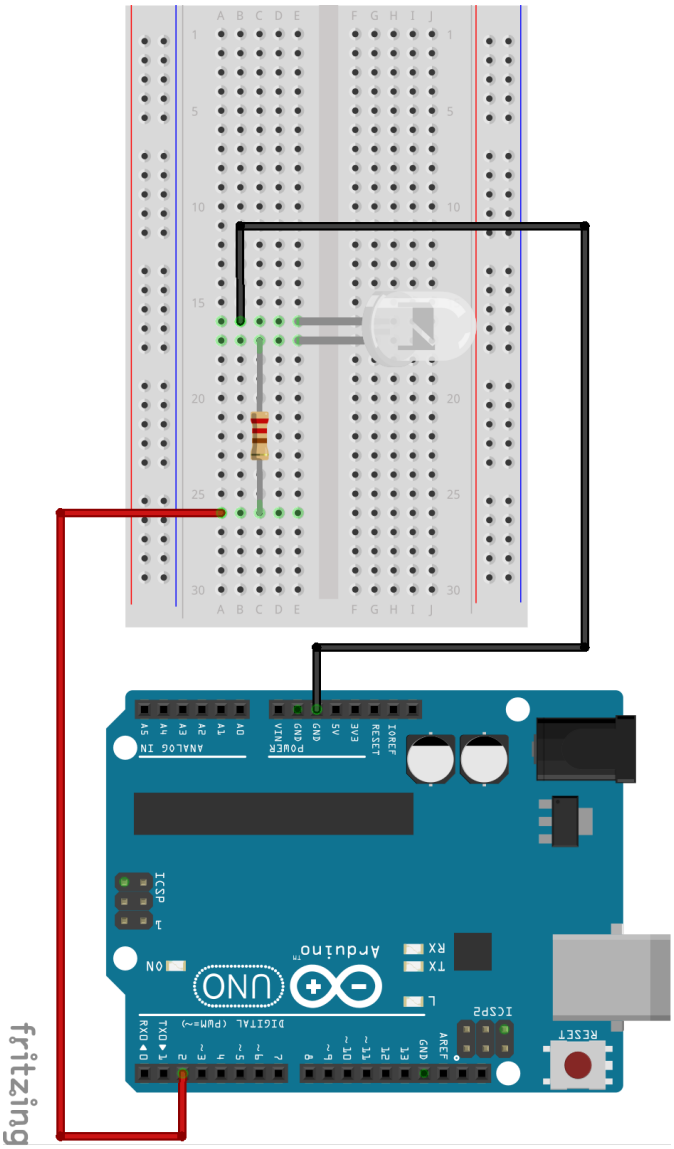
Down

1 the opposite of output, Arduino pins can be these
2 a input and output method where there can be many options
4 the type of energy in a lightning strike
5 the command (code) used when you want the Arduino to wait
6 very simple computer often called a micro-controller
12 a component, a small light

Across

3 the flow of electrical charge through a circuit
5 a input and output method where there can only be two options on or off
7 the unit of resistance
8 a component used to reduce the current in a circuit
9 the unit of electrical charge or power (used in batteries)
10 the opposite of input, Arduino pins can be these
11 a plastic slab with holes in it that can connect wires
13 powers all of your electronic devices, measured in volts

Blink Circuit Diagram



Resources

Introduction to Arduino - 1
https://www.youtube.com/watch?v=CqrQmQqpHXc

Introduction to Arduino - 2
https://www.allaboutcircuits.com/projects/getting-started-with-the-arduino/

Arduino: Troubleshooting
https://www.arduino.cc/en/Guide/Troubleshooting

Arduino Clone Drivers
https://sparks.gogo.co.nz/ch340.html

How to spot an Arduino Clone
https://www.arduino.cc/en/Products/Counterfeit

10 Great Arduino Projects for Beginners
https://www.makeuseof.com/tag/10-great-arduino-projects-for-beginners/

10 Arduino Projects for Beginners anyone can make
https://www.circuito.io/blog/arduino-projects-for-beginners/

16 Beginner Projects
https://create.arduino.cc/projecthub/projects/tags/beginner

130 Arduino Projects
https://circuitdigest.com/arduino-projects

The Absolute Beginner's Guide to Arduino
http://forefront.io/a/beginners-guide-to-arduino/