**-STEP1-**

**What is Arduino?**

Arduino is an open-source electronics platform based on easy-to-use hardware and software. [Arduino boards](https://www.arduino.cc/en/Main/Products) are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of [accessible knowledge](http://forum.arduino.cc/) that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The [software](https://www.arduino.cc/en/Main/Software), too, is open-source, and it is growing through the contributions of users worldwide.

**Why Arduino?**

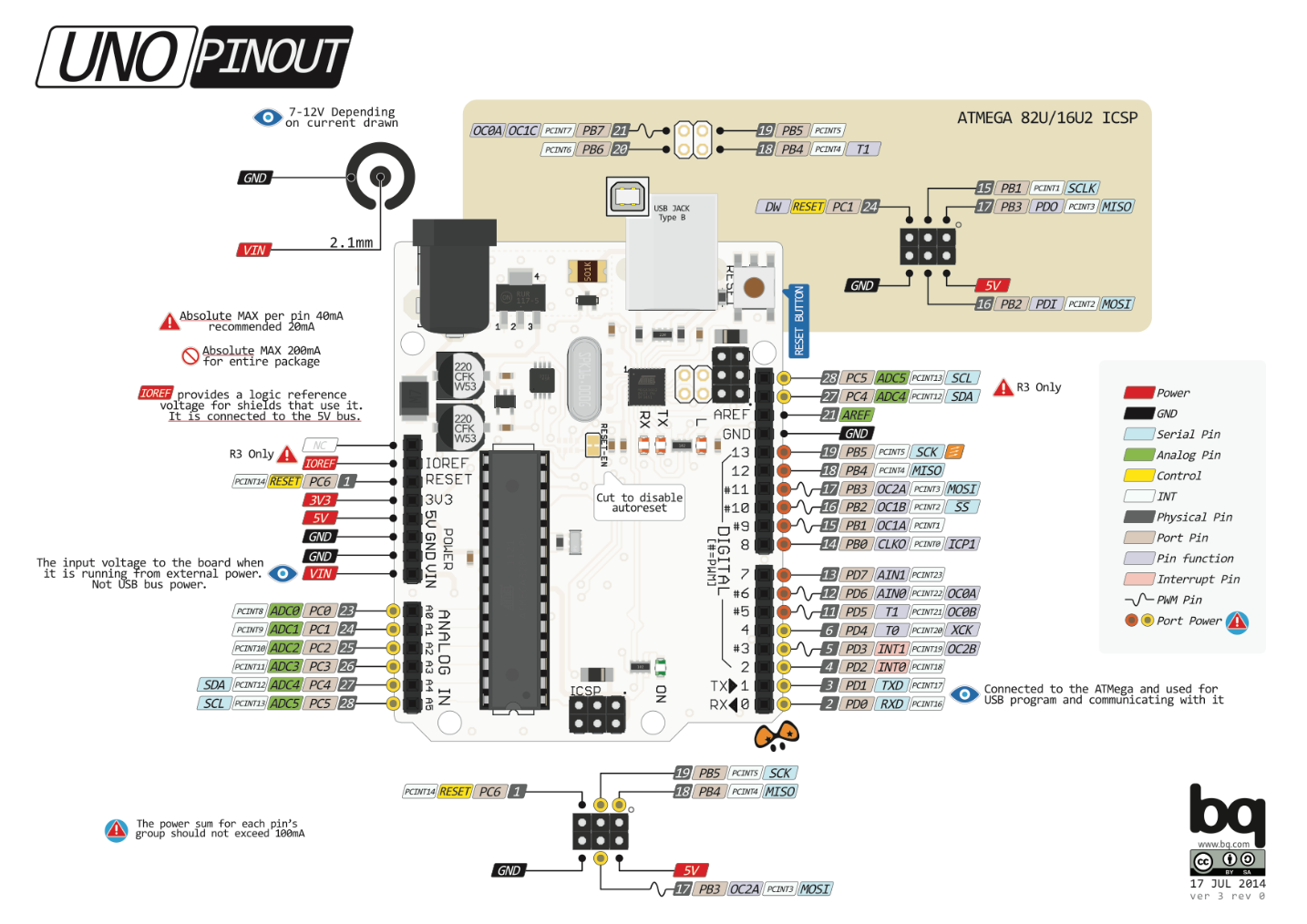
Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality.

All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
* Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
* Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
* Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
* Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the [breadboard version of the module](https://www.arduino.cc/en/Main/Standalone) in order to understand how it works and save money.

The Arduino Board itself is a blue circuit board, the size of a credit card (but they also have models in other sizes). It has two rows of connectors (the 'headers'), a power connector and a USB connector. The brain of the board is an Atmel microcontroller. It's like a really small, very low power 'computer'. (It only has 32KB of storage, 2KB of RAM, and the 8-bit processor runs at only 16MHz.) For most applications, however, this is more than enough. The pins of the processor connect to the headers, and you can connect them to virtually anything you can imagine.



You just need to write some lines of code to control them. The I/O pins (Input/Output) can be used as input pins, to connect buttons or knobs, temperature sensors, light sensors, or even keyboards and mouses, digital musical instruments … or they can be used as output pins, to connect LEDs, drive motors, control the lights in your home, connect to small displays or even connect to the Internet, so that it can check your mail, post tweets ... Through the USB connection you can also use it to control your computer or use your computer to control the Arduino.

**-STEP2-**

**How not to fry your Arduino**

Before you begin plugging things into your new Arduino, it may be good to know what can damage the board.

**1-Drawing more than 40mA from an output pin.**

An Arduino can only supply 40mA per output pin, so you cannot drive a motor or a speaker directly, for example, and you cannot connect an LED directly (without a resistor). In the course of this Instructable, I'll explain what you should do instead.  
Shorting an output in to the +5v, +3.3v or the ground pins, will also kill your board: If an output pin is at 5v for example, and you connect it to the ground, it draws an enormous amount of current, and kills your Arduino almost instantly.  
The pins go through the circuit board, so make sure you don't place the Arduino on a conductive (metal) surface, because it will short out the pins.

**2- Drawing more than 200mA from all output pins together.**

The ATmega chip on your Arduino can only supply 200mA in total, so driving more than 10 LEDs @ 20mA each, for example, will eventually damage your board.

**3- Supplying more than 5v (3.3v) to an input pin.**

Supplying more than the operating voltage of the Arduino on any pin is very dangerous. Some Arduinos that run at 3.3v have 5v tolerant pins, but that's about it. This also holds true for other devices, like sensors or wireless chips: always check the voltages: if you connect the output of a 5V Arduino to a 3.3V chip, you might kill it.

**4- Supplying more than 5v to the 5v pin.**

The 5v of the Arduino board goes directly to the ATmega chip, that is rated for an absolute maximum of 6v.

**5- Supplying more than 12v to the Vin pin.**

There's an onboard 5v voltage regulator on the board, that will overheat and die if you feed it with more than 12v.

**6- Drawing more than 500mA from the 5v pin (when running off an external power supply.**

The onboard 5v voltage regulator can only supply 500mA of current. The 5vUSB has a polyfuse to limit the current to 500mA.

**7- Drawing more than 50mA from the 3.3v pin.**

The onboard 3.3v voltage regulator can only supply 50mA of current. This means that you can not connect power hungry 3.3v devices like an ESP8266 or nRF24L01 directly to the Arduino: you need an external 3.3v voltage regulator.

**8- Reversing the polarity of the power supply.**

If you swap the 5v or Vin pin with the GND pin, you'll kill the board almost instantly.  
The barrel jack has a diode to protect against reverse polarity.

**9- Connecting a load to the Vin pin while using USB power.**

If you connect a load to the Vin pin while the 5v to the Arduino comes from the USB connection, current will flow backwards through the voltage regulator, damaging it.

**10- Static electricity**

Although most chips have clamping diodes as protection against ESDs (electrostatic discharges), it may be wise to us an anti-static wrist strap, or to remove the carpet under your desk.