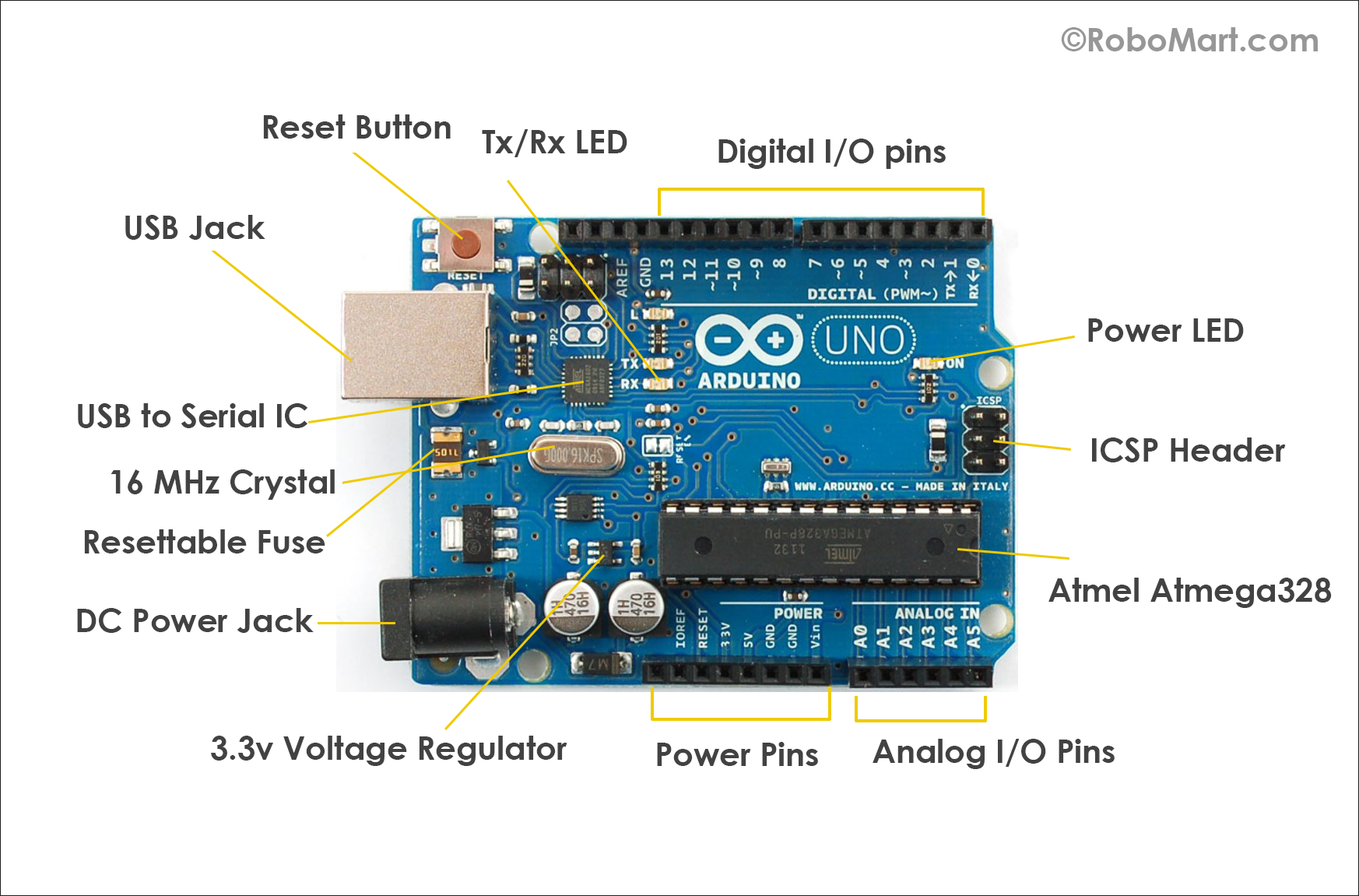
**Arduino Hardware & Pinout**



The PCB design of the Arduino UNO uses SMD (Surface Mount Device) components.

After your code is compiled using Arduino IDE, it should be uploaded to the main microcontroller of the Arduino UNO using a USB connection. Because the main microcontroller doesn’t have a USB transceiver, you need a bridge to convert signals between the serial interface (UART interface) of the microcontroller and the host USB signals.

The bridge in the latest revision is the ATmega16U2, which has a USB transceiver and also a serial interface (UART interface).

**Power (USB / Barrel Jack)**

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. The USB connection is also how you will load code onto your Arduino board.

**Reset Button**

Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn’t repeat, but you want to test it multiple times.

**Power LED Indicator**

This LED should light up whenever you plug your Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong. Time to re-check your circuit!

**TX RX LEDs**

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication.

In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we’re loading a new program onto the board).

**Main IC**

Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC’s from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC’s, reading the datasheets is often a good idea.

The ATmega328 microcontroller is the MCU used in Arduino UNO R3 as a main controller. ATmega328 is an MCU from the AVR family; it is an 8-bit device, which means that its data-bus architecture and internal registers are designed to handle 8 parallel data signals.

ATmega328 has three types of memory:

* **Flash memory:** 32KB nonvolatile memory. This is used for storing application, which explains why you don't need to upload your application every time you unplug arduino from its power source.
* **SRAM memory:** 2KB volatile memory. This is used for storing variables used by the application while it's running.
* **EEPROM memory:** 1KB nonvolatile memory. This can be used to store data that must be available even after the board is powered down and then powered up again.

**Voltage Regulator**

The voltage regulator is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it’s for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don’t hook up your Arduino to anything greater than 20 volts.

**The USB-to-UART Bridge**

As we discussed in the “Arduino UNO System Overview” section, the role of the USB-to-UART bridge part is to convert the signals of USB interface to the UART interface, which the ATmega328 understands, using an ATmega16U2 with an internal USB transceiver. This is done using special firmware uploaded to the ATmega16U2.

From an electronic design perspective, this section is similar to microcontroller section. This MCU has an ICSP header, an external crystal with load capacitors (CL), and a Vcc filter capacitor.

Notice that there are series resistors in the D+ and D- USB lines. These provide the proper termination impedance for the USB signals.

Z1 and Z2 are voltage-dependent resistors (VDRs), also called varistors. They are used to protect the USB lines against ESD transients.

The 100nF capacitor connected in series with the reset line allows the Atmega16U2 to send a reset pulse to the Atmega328.

**16 MHz Ceramic Resonator (CSTCE16M0V53-R0)**

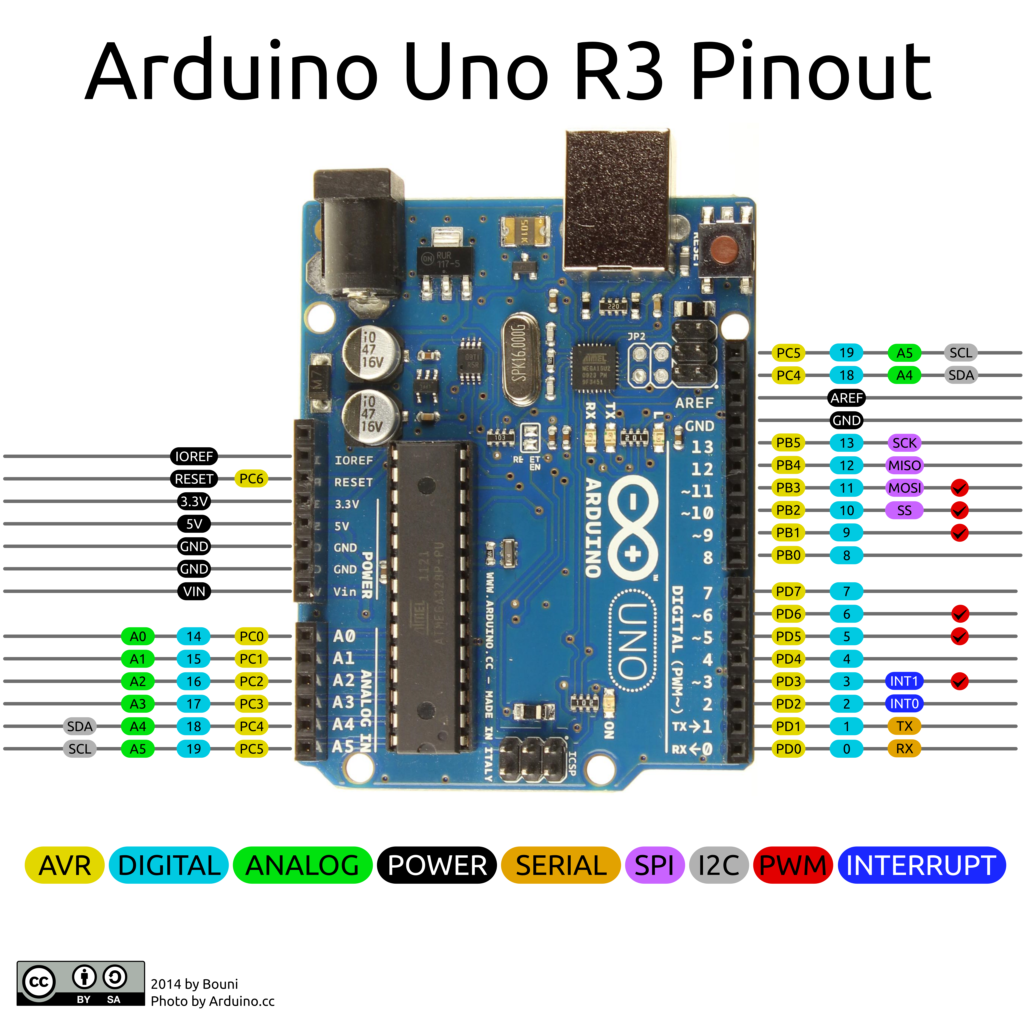
Connected with XTAL2 and XTAL1 from the MCU.

**C4 and C6 100nF Capacitors**

These are added to filter supply noise. The impedance of a capacitor decreases with frequency: The capacitors give high-frequency noise signals a low-impedance path to ground. 100nF is the most common value.

**ICSP (In-Circuit Serial Programming) Header**

This is used to program the ATmega328 using an external programmer. It’s connected to the In-System Programming (ISP) interface (which uses the SPI pins). Usually, you don’t need to use this way of programming because bootloader handles the programming of the MCU from the UART interface which is connected using a bridge to the USB. This header is used when you need to flash the MCU, for example, with a bootloader for the first time in production.

****

* **GND**: Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
* **5V & 3.3V**: As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
* **Analog**: The area of pins under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
* **Digital**: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
* **PWM**: You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). You can think of these pins as being able to simulate analog output (like fading an LED in and out).
* **AREF**: Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.
* **SPI (SCK, MISO, MOSI, SS)**: Check SPI Word Document.
* **I2C (SCL, SDA)**: Check I2C Word Document.
* **Reset Pin:** This is pulled up with a 10K resistor to help prevent spurious resets in noisy environments; the pin has an internal pull-up resistor, but according to the AVR Hardware Design Considerations application note (AVR042), “if the environment is noisy, it can be insufficient and reset may occur sporadically.” Reset occurs if the user presses the reset button or if a reset is issued from the USB bridge. You can also see the D2 diode. The role of this diode is described in the same app note: “If not using High Voltage Programming it is recommended to add an ESD protection diode from RESET to Vcc, since this is not internally provided due to High Voltage Programming”.

**In-system Programming:**

In-system programming (ISP), also called in-circuit serial programming (ICSP), is the ability of some [programmable logic devices](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUHJvZ3JhbW1hYmxlX2xvZ2ljX2RldmljZQ), [microcontrollers](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTWljcm9jb250cm9sbGVy), and other [embedded devices](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvRW1iZWRkZWRfZGV2aWNl) to be programmed while installed in a complete system, rather than requiring the chip to be programmed prior to installing it into the system.

There are several mutually-incompatible in-system programming protocols for programming [microcontroller](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTWljcm9jb250cm9sbGVy) devices, including [PIC microcontrollers](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUElDX21pY3JvY29udHJvbGxlcg), [AVRs](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQXRtZWxfQVZS), and the [Parallax Propeller](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUGFyYWxsYXhfUHJvcGVsbGVy). ICSP has been primarily implemented by [Microchip Technology](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTWljcm9jaGlwX1RlY2hub2xvZ3k) for programming PIC and dsPIC devices.

The primary advantage of this feature is that it allows manufacturers of electronic devices to integrate programming and testing into a single production phase, and save money, rather than requiring a separate programming stage prior to assembling the system. This may allow manufacturers to program the chips in their own system's production line instead of buying preprogrammed chips from a manufacturer or distributor, making it feasible to apply code or design changes in the middle of a production run.

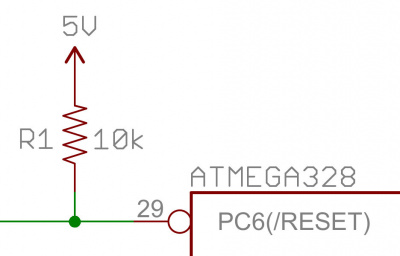
Microcontrollers are typically soldered directly to a printed circuit board and usually do not have the circuitry or space for a large external programming cable to another computer.

Typically, chips supporting ISP have internal circuitry to generate any necessary programming voltage from the system's normal supply voltage, and communicate with the programmer via a serial protocol. Most programmable logic devices use a variant of the [JTAG](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvSlRBRw) protocol for ISP, in order to facilitate easier integration with automated testing procedures. Other devices usually use proprietary protocols or protocols defined by older standards. In systems complex enough to require moderately large [glue logic](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvR2x1ZV9sb2dpYw), designers may implement a JTAG-controlled programming subsystem for non-JTAG devices such as [flash memory](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvRmxhc2hfbWVtb3J5) and microcontrollers, allowing the entire programming and test procedure to be accomplished under the control of a single protocol.

An example of devices using ISP is the [AVR](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQXRtZWxfQVZS) line of micro-controllers by [Atmel](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQXRtZWw) such as the [ATmega](http://www.wiki-zero.co/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQVRtZWdhI0Jhc2ljX2ZhbWlsaWVz" \o "ATmega) series.

## 

## Introduction

[](https://cdn.sparkfun.com/assets/0/5/9/0/8/513901dfce395f671a000000.jpg)

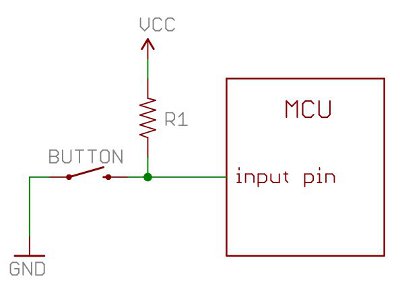
Pull-up resistors are very common when using microcontrollers (MCUs) or any digital logic device. This tutorial will explain when and where to use pull-up resistors, then we will do a simple calculation to show why pull-ups are important.

## What is a Pull-up Resistor

Let’s say you have an MCU with one pin configured as an input. If there is nothing connected to the pin and your program reads the state of the pin, will it be high (pulled to VCC) or low (pulled to ground)? It is difficult to tell. This phenomena is referred to as floating. To prevent this unknown state, a pull-up or pull-down resistor will ensure that the pin is in either a high or low state, while also using a low amount of current.

For simplicity, we will focus on pull-ups since they are more common than pull-downs. They operate using the same concepts, except the pull-up resistor is connected to the high voltage (this is usually 3.3V or 5V and is often refereed to as VCC) and the pull-down resistor is connected to ground.

Pull-ups are often used with buttons and switches.

[](https://cdn.sparkfun.com/assets/6/f/b/c/7/511568b6ce395f1b40000000.jpg)

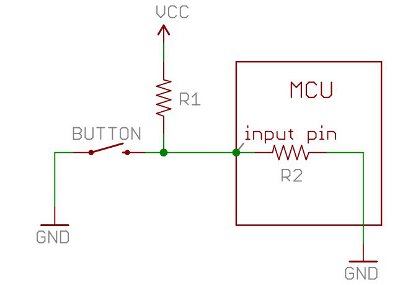
With a pull-up resistor, the input pin will read a high state when the button is not pressed. In other words, a small amount of current is flowing between VCC and the input pin (not to ground), thus the input pin reads close to VCC. When the button is pressed, it connects the input pin directly to ground.

The current flows through the resistor to ground, thus the input pin reads a low state. Keep in mind, if the resistor wasn’t there, your button would connect VCC to ground, which is very bad and is also known as a short.

So what value resistor should you choose?

The short and easy answer is that you want a resistor value on the order of 10kΩ for the pull-up.

**A low resistor value is called a strong pull-up (more current flows), a high resistor value is called a weak pull-up (less current flows).**

[](https://cdn.sparkfun.com/assets/f/1/4/a/b/511568b7ce395f613f000004.jpg)

The value of the pull-up resistor needs to be chosen to satisfy two conditions:

1. When the button is pressed, the input pin is pulled low. The value of resistor R1 controls how much current you want to flow from VCC, through the button, and then to ground.
2. When the button is not pressed, the input pin is pulled high. The value of the pull-up resistor controls the voltage on the input pin.

For condition 1, you don’t want the resistor’s value too low. The lower the resistance, the more power will be used when the button is hit. You generally want a large resistor value (10kΩ), but you don’t want it too large as to conflict with condition 2. A 4MΩ resistor might work as a pull-up, but its resistance is so large (or weak) that it may not do its job 100% of the time.

The general rule for condition 2 is to use a pull-up resistor (R1) that is an order of magnitude (1/10th) less than the input impedance (R2) of the input pin. An input pin on a microcontroller has an impedance that can vary from 100k-1MΩ. For this discussion, impedance is just a fancy way of saying resistance and is represented by R2 in the picture above. So, when the button is not pressed, a very small amount of current flows from VCC through R1 and into the input pin. The pull-up resistor R1 and input pin impedance R2 divides the voltage, and this voltage needs to be high enough for the input pin to read a high state.

For example, if you use a 1MΩ resistor for the pull-up R1 and the input pin’s impedance R2 is on the order of 1MΩ (forming a voltage divider), the voltage on the input pin is going to be around half of VCC, and the microcontroller might not register the pin being in a high state. On a 5V system, what does the MCU read on the input pin if the voltage is 2.5V? Is it a high or a low? The MCU doesn’t know and you might read either a high or a low. A resistance of 10k to 100kΩ for R1 should avoid most problems.

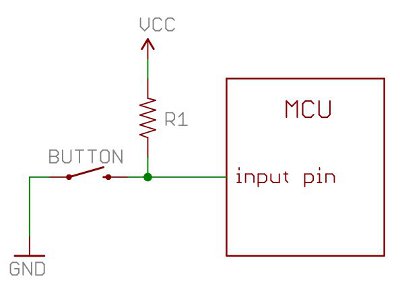
Since pull-up resistors are so commonly needed, many MCUs, like the ATmega328 microcontroller on the Arduino platform, have internal pull-ups that can be enabled and disabled. To enable internal pull-ups on an Arduino, you can use the following line of code in your setup() function:

pinMode(5, INPUT\_PULLUP); // Enable internal pull-up resistor on pin 5

Another thing to point out is that the larger the resistance for the pull-up, the slower the pin is to respond to voltage changes. This is because the system that feeds the input pin is essentially a capacitor coupled with the pull-up resistor, thus forming a RC filter, and RC filters take some time to charge and discharge. If you have a really fast changing signal (like USB), a high value pull-up resistor can limit the speed at which the pin can reliably change state. This is why you will often see 1k to 4.7KΩ resistors on USB signal lines.

All of these factors play into the decision on what value pull-up resistor to use.

## Calculating a Pull-up Resistor Value

[](https://cdn.sparkfun.com/assets/6/f/b/c/7/511568b6ce395f1b40000000.jpg)

Let’s say you want to limit the current to approximately 1mA when the button is pressed in the circuit above, where Vcc = 5V. What resistor value should you use?

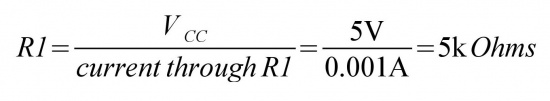
It is easy to show how to calculate the pull-up resistor using Ohm’s Law:

[alt text](https://cdn.sparkfun.com/assets/a/5/0/2/6/5113d140ce395f777e000000.gif)

Referring to the schematic above, Ohm’s Law now is:

[alt text](https://cdn.sparkfun.com/assets/9/5/c/3/2/51391363ce395f2d25000000.jpg)

Rearrange the above equation with some simple algebra to solve for the resistor:

[](https://cdn.sparkfun.com/assets/4/4/e/d/9/51391364ce395ffb24000000.jpg)

Remember to convert all of your units into volts, amps and Ohms before calculating (e.g. 1mA = 0.001 Amps). The solution is to use a 5kΩ resistor.