BIT-101 Remember when Bill Gates touched my Mac Book Pro? That was awes

Home About / Contact FAQ

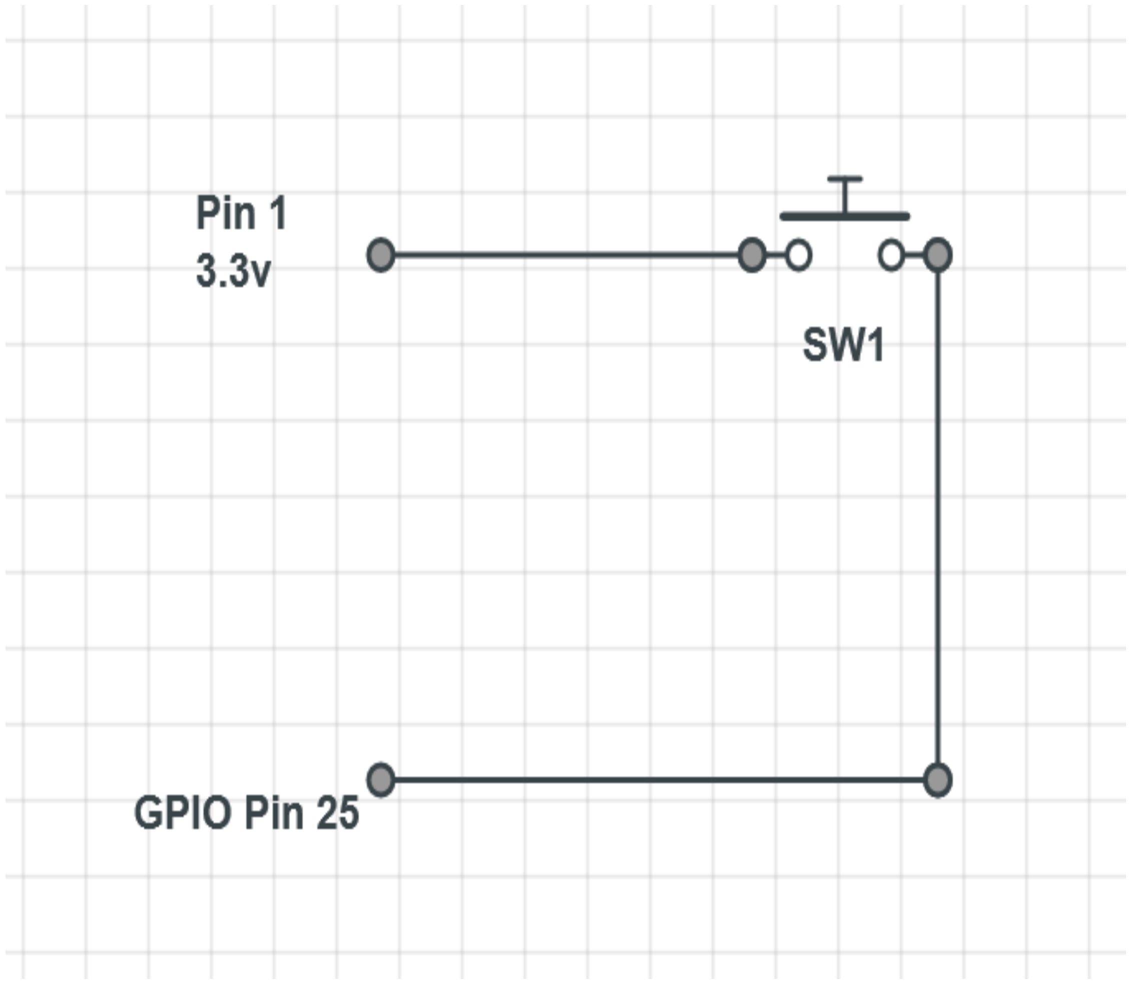
Pull-up and Pull-down Resistors

May 22 2013 Published by keith under Electronics, Physical Computing

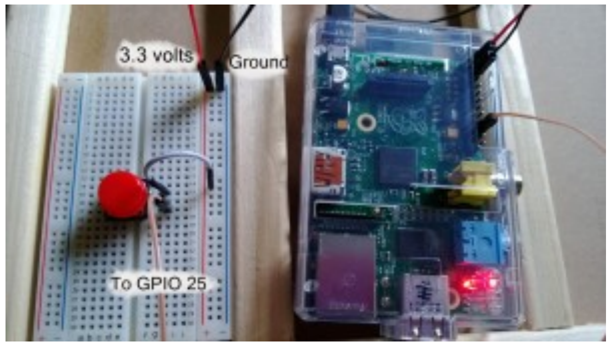
Send to Kindle

Working a lot with Raspberry Pi and Arduino stuff lately. The concept of pull-up and pull-down resistors came up quickly and confused me a little at first. So I thought I'd do a little demo of how they work and why they are needed. Doing this helped to clarify it for me, so maybe it'll help you too.

Common scenario. You want to set up something that reads an input of a GPIO pin. Say, you want to know when the user is pressing a switch. You set up a simple circuit like so:



And here that is wired up:



When you press the button, pin 25 goes high. No press, no high, must be low, right? Well, let's see…

We'll set up a simple Python script to read the status of pin 25:

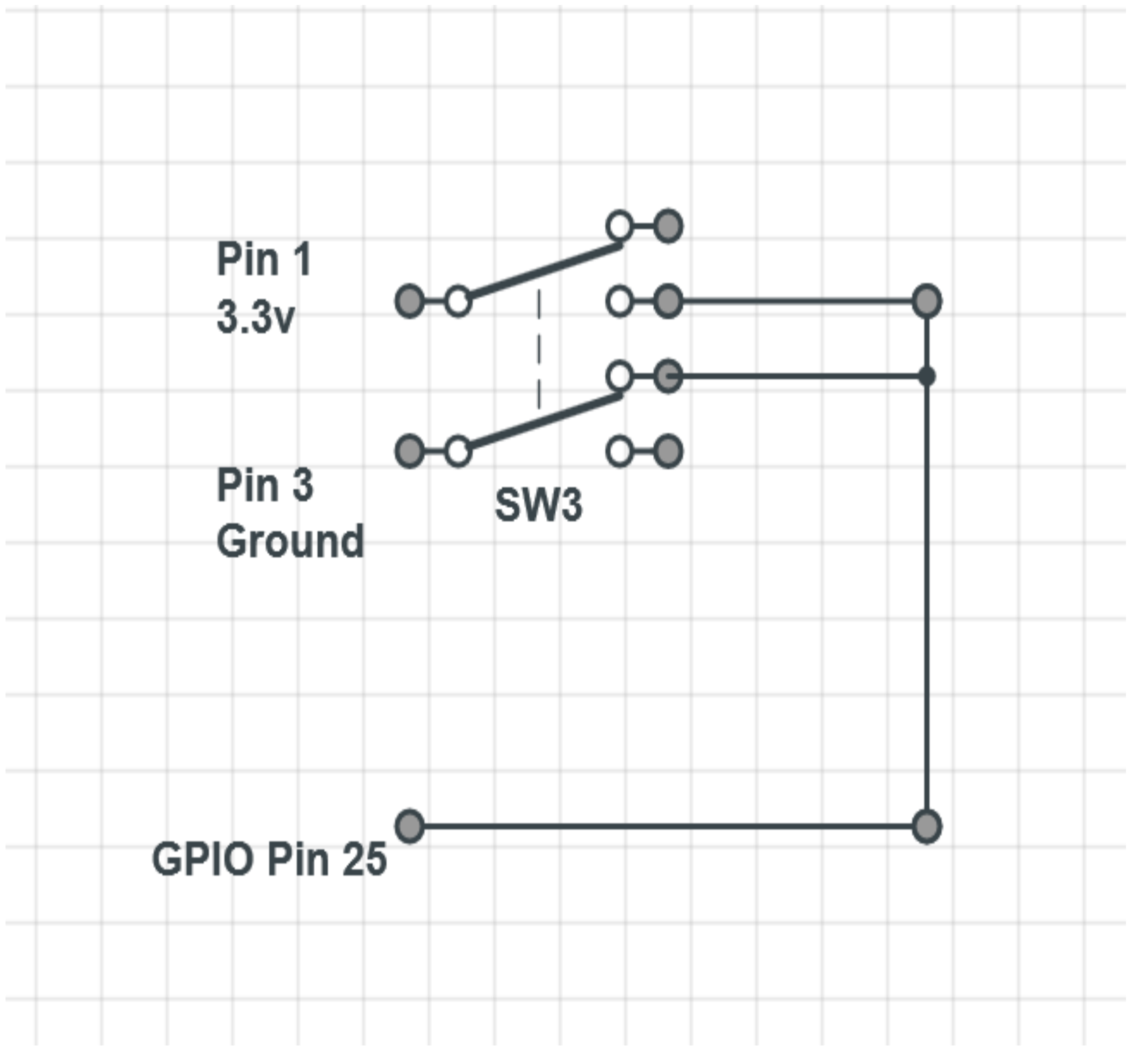
```
#! /usr/bin/python

import RPi.GPIO as GPIO
import time
```

```

PIN = 25

GPIO.setmode(GPIO.BCM)
GPIO.setup(PIN, GPIO.IN)

while True:
        if GPIO.input(PIN) == GPIO.HIGH:
                print("pin is high")
                time.sleep(.1)
        else:
                print("pin is low")
                time.sleep(.1)
```

When I run this, I get 20-30 lines saying pin 25 is high, then 20-30 saying it's low, back and forth, without touching the button. Sure enough when I press the button, it stays high, but apparently the definition of "low" is not "not high".

After too many years in the highly binary world of software programming, some of us delving into hardware may be surprised to learn that a lot of electronics operate on "tri-state logic" as opposed to simple binary. Here, an input can be high, low, or floating. In this particular circuit, high means connected to 3.3 volts, low means connected to ground, and floating means… neither or somewhere in between. Actually, due to the sensitive nature of tiny electronics, system components can pick up various signals that create slight fluctuations in what it is reading. It's rarely if ever going to pick up exactly 0.000 volts just because it's not connected to 3.3 volts.

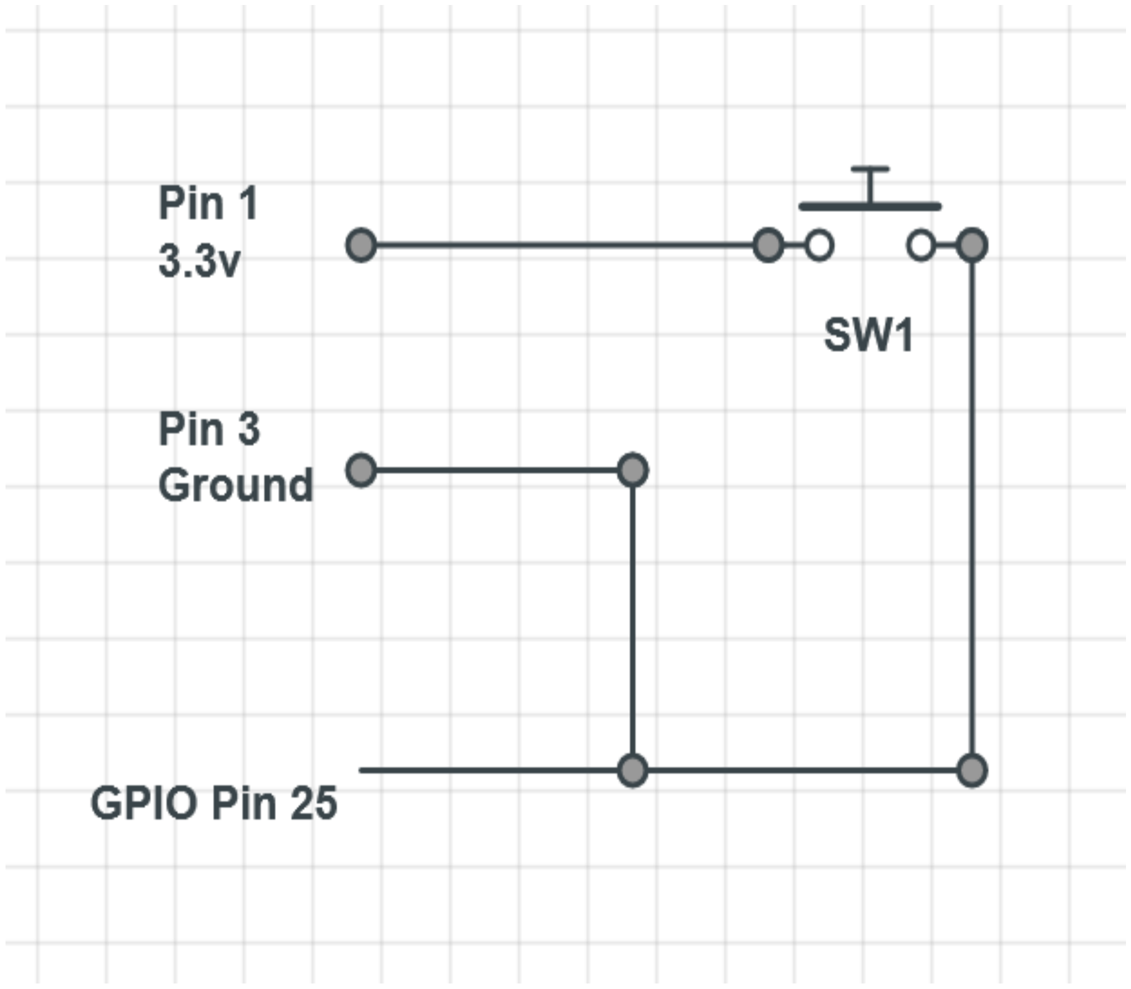
So we need to force the issue and say, "Yo! You are LOW!" My first thought on how to do this would probably have been to do something like this:



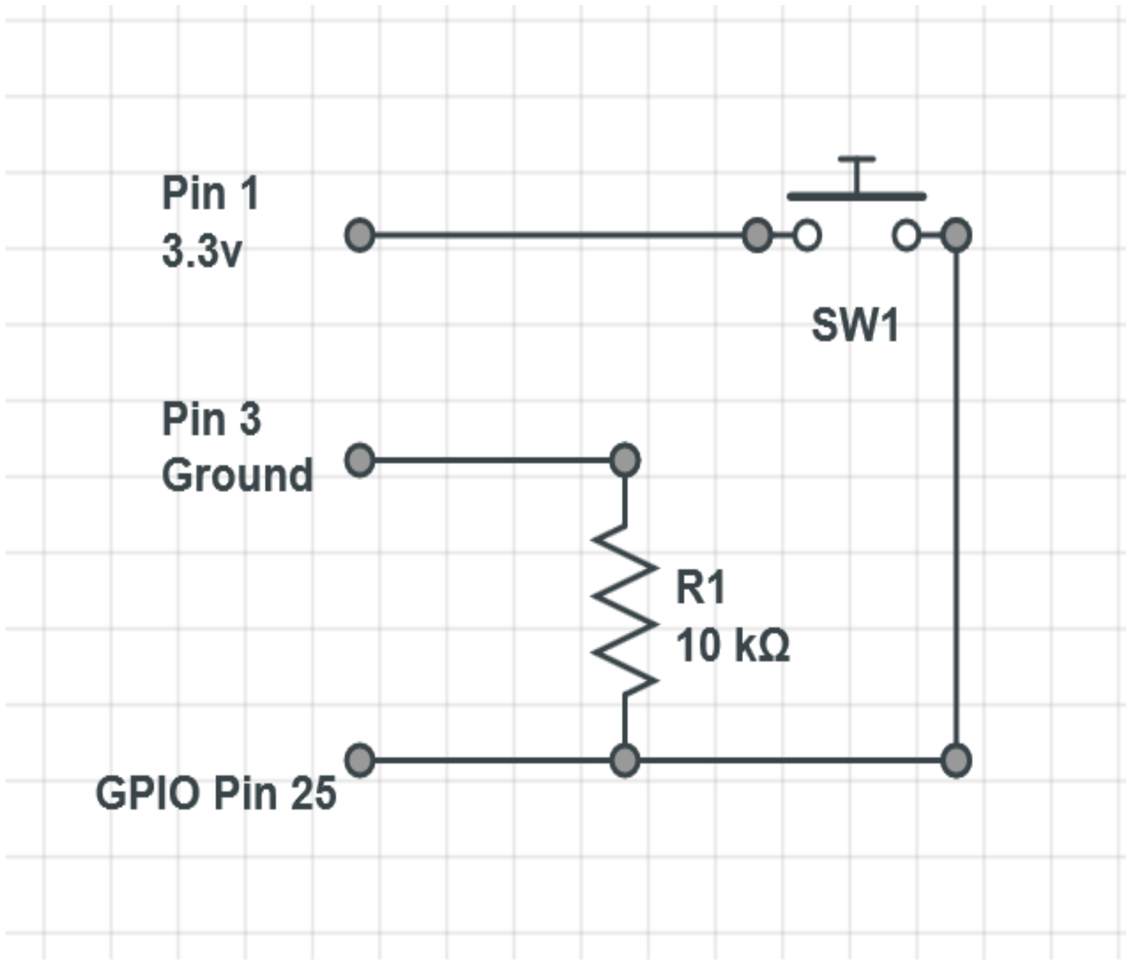
When the switch is in one position, pin 25 is connected directly to ground, sending it solidly low. When the switch is changed, it connects 25 to 3.3 volts, making it high. This will work just fine, but it's a bit overengineered as it turns out.

How about if we go simpler and just connect pin 25 to ground and leave it that way, like so:

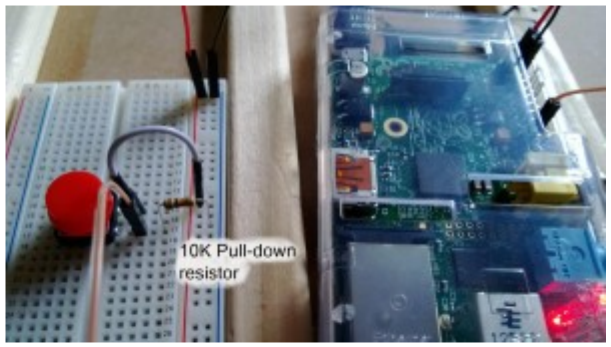
(Note: DO NOT ACTUALLY BUILD THIS CIRCUIT!)



Now, when the switch is open, pin 25 is undoubtedly low. But we have a problem that when you do hit the switch, you now have a short circuit directly between 3.3 volts and ground. Not good. Again, do not do this. The solution? Throw a resistor in there:



And in the real world:



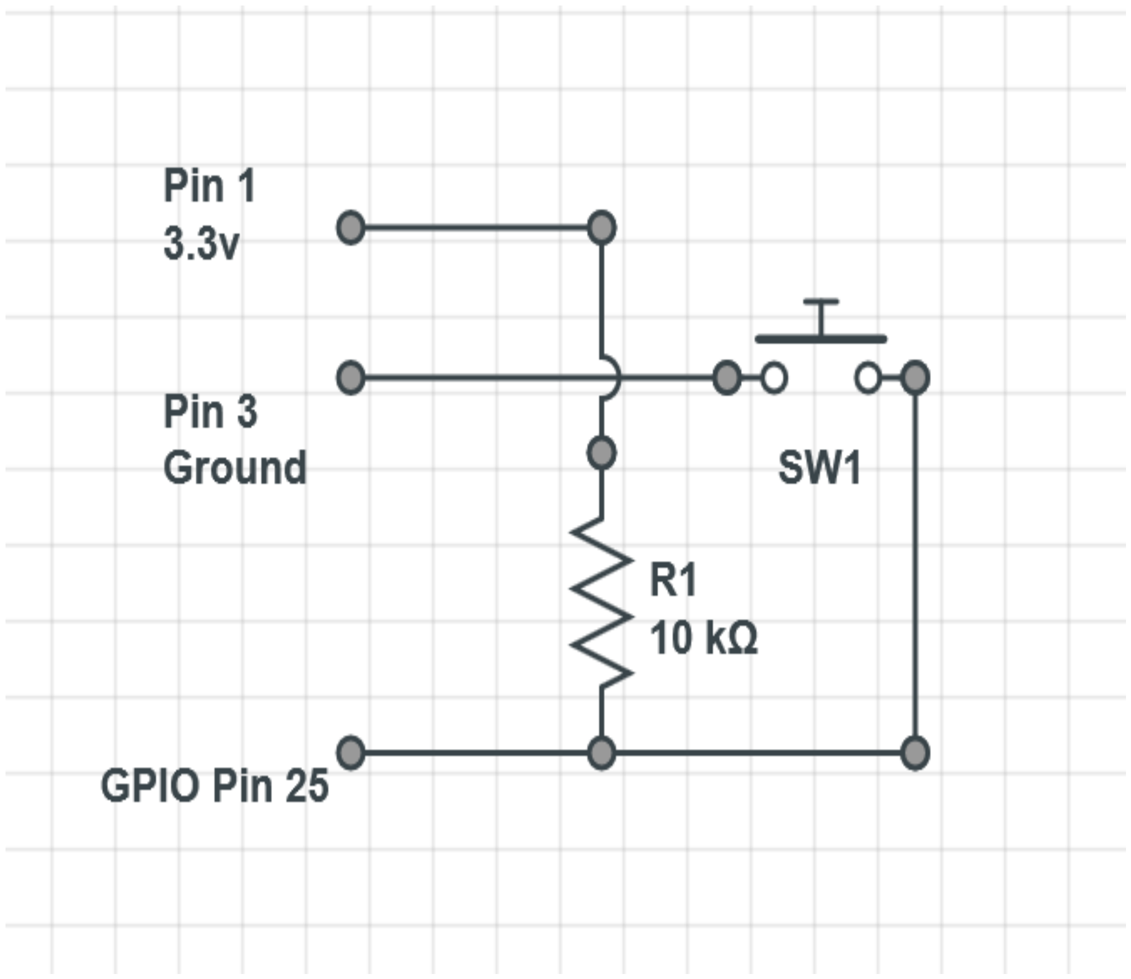
Now, when the switch is open, pin 25 is connected to ground through the 10K resistor. Even with the resistor though, it's enough of a connection to make it solidly low. It's not going to fluctuate.

Then we hit the switch, connecting the lower part of the circuit to the 3.3 volts. Because we have a fairly high resistance there, most of the current is going to go to pin 25, sending it solidly high. Some of it is also going to go to ground via

R1. But Ohm's Law tells us that 3.3 volts divided by 10,000 ohms will let about 0.33 milliamps in. That's not going to break the bank.

So in this circuit, R1 is called a pull-down resistor because in its default state it pulls the signal down to 0 volts or a low state.

Now let's swap things around a bit:



Now pin 25 is hard-wired to 3.3 volts through R1, making its default state high. However when the button is pressed, pin 25 will be directly connected to ground, sending it low. Because the resistor pulls pin 25 up to 3.3 volts and a high state, it's now known as a pull-up resistor.

So, which one do you use? It depends on whether you want your default signal to be high or low.

Hope that helps.

Send to Kindle

Tweet 10

Donate

13 responses so far. Comments will be closed after post is one year old.

Peter Goes May 23, 2013 at 4:18 am

Hi there,

After learning a lot from your software knowledge, now on to the hardware! Nice.
Great post, which not only explains quite well how to wire a pull-up/pull-down resistors.
But along the way it let me connect the dots about when and why to put in resistors!

Thank you!
- Peter

Andrew Gilmartin July 25, 2013 at 10:02 am

First, thanks for the posting these useful notes on electronic projects. The photos in this posting show the hardware parts held in a jig. This looks like a great way to organize project hardware that needs to move (mostly out of the way). Is it just a board with rails or is there more to it that is out of the image's frame?

– Andrew

keith July 25, 2013 at 10:58 am

Yeah, it's just a little something I put together in my shop. A quarter inch board cut to size, then a thin strip across the bottom and four vertical strips glued on. This creates three slots that can hold an arduino/pi/breadboard/etc. You hook up your project, then you can move it around or put it aside or whatever without carrying all the parts separately, hanging together with jumpers. 🙂

cumana August 7, 2013 at 8:18 am

Hello, thank you very much for this explanation! Now it seems pretty simple.

Best wishes.

Gratefull man August 16, 2013 at 10:20 am

This is basically the best "guide" to pullUp/Down Resistors I was able to find on the net- No I finally get it! Thanks alot!

giro August 23, 2013 at 12:05 pm

So, if I understand this correctly (in the first example): Putting the resistor to ground prevents a short circuit when powered (thereby 'pushing' current to pin 25), however it permits floating current/static/noise to go to ground when not powered, as these levels would logically be less than the resistance between the circuit (through the resistor) and ground. I think a light just went on, or maybe turned off, but completely this time 😃

Cheers, and many thanks for the great example!

Arduino Newb October 22, 2013 at 5:53 pm

I'm confused when the switch is not engaged in the pull-up resistor — wouldnt the voltage at pin 25 be 3.3V minus whatever is lost over the resistor? How can this be considered "high", the resistor is very large?

keith October 22, 2013 at 6:01 pm

Any connection to a voltage, even through a resistor, is considered high. Any connection to ground, even through a resistor, is considered low. Anything not connected to either one is considered floating.

Yet another noob December 18, 2013 at 11:03 am

I think Arduino Newb is on the right track there. The pull-up resistor and any resistance at pin 25 (probably quite high) will form a voltage divider. The answer might be "choose a small enough resistor to allow high enough voltage (so that it doesn't drop to floating levels), but large enough to keep the current at a desired level".

Arvind March 24, 2014 at 12:36 am

Exactly what i was looking for.

Thanks for the thread of comments.

Nick February 26, 2014 at 9:16 am

well explained, thanks!

Tyrone April 10, 2014 at 11:30 am

This explanation of pull-up/pull-down resistors was very clear. I finally understand how and when to use them. Thank you very much.

Mike June 11, 2014 at 12:32 pm

Thank you so much, Keith. I also quickly found a need to learn about this topic and have been reading on lots of “professional” sites but this is the best explanation so far.