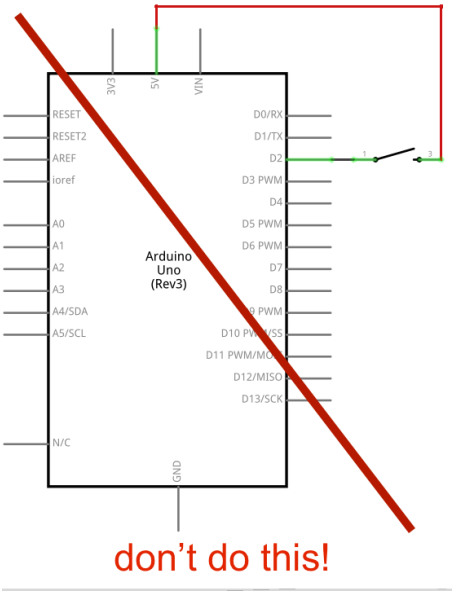
Basic switch/button use on an Arduino

Switches and buttons are easy to wire up once you understand a few basics:

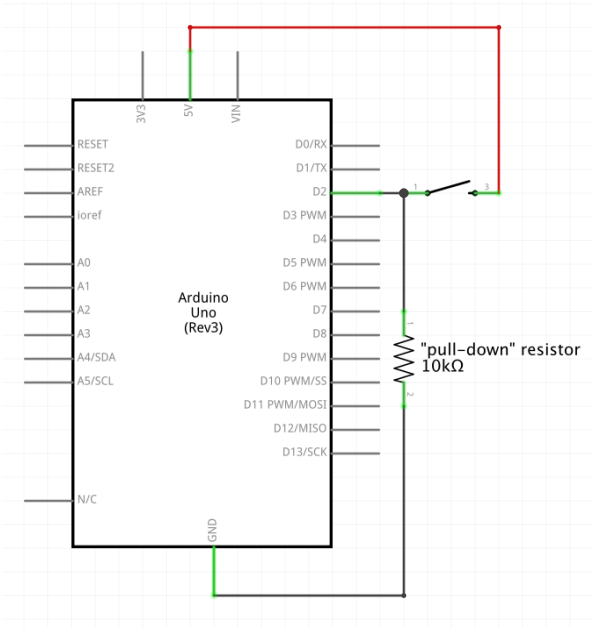
1) For reliable readings, some electrical input must always be attached to the input pin
2) On an Arduino Uno, 5 volts means `HIGH` and 0 volts (ground) means `LOW`
3) Buttons and switches may have non-obvious internal wiring!

An easy mistake

Many beginners often build a circuit like the one to the right—>



Makes sense. When the button is pushed, the input pin sees 5V, and when it's not pushed, it will see 0V so it will be grounded (because nothing is plugged in to it). Right?



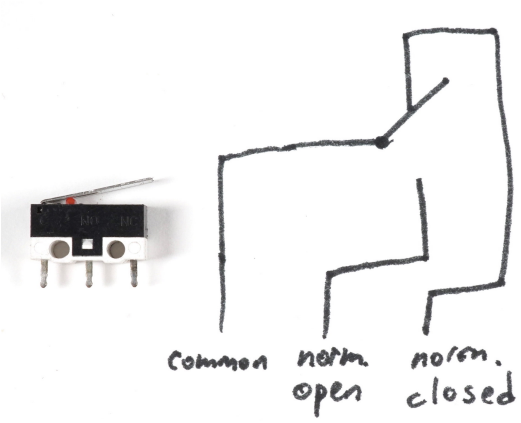
Nope. As point #1 says above, you must wire *some* electrical input to an input pin at all times. When the button on the right is not pushed, pin 2 is connected to nothing but a bit of wire. It will *not* reliably read 0 volts, because of the way digital inputs are read on the Arduino.

The solution? Add a "pull-down resistor" to make it so that when the button is not being pushed, the input pin will be connected to ground, as shown on the left.

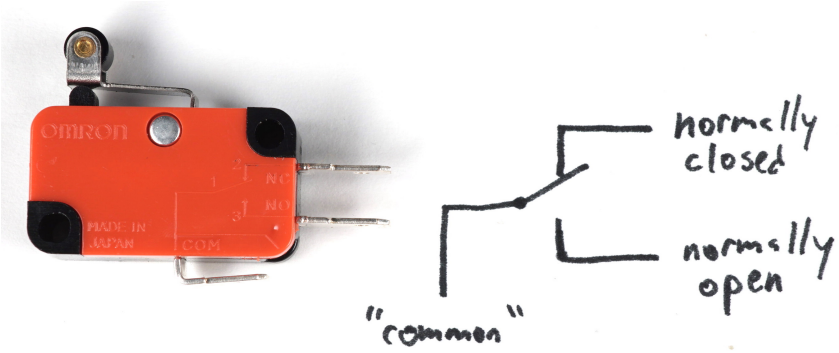
Simple Arduino code to read a button or switch and send the result via Serial monitor:

```
const int BUTTONPIN = 2; // button plugged into pin 2

void setup() {
  pinMode(BUTTONPIN, INPUT); // set the pin up as an electrical input
  Serial.begin(9600); // start serial communication with the computer
}

void loop() {
  int buttonVal = 0; // make a new variable to hold the button state
  buttonVal = digitalRead(BUTTONPIN); // find the current button state
  Serial.println(buttonVal); // send the value back to the computer
  delay(50); // wait a bit before doing it again
}
```

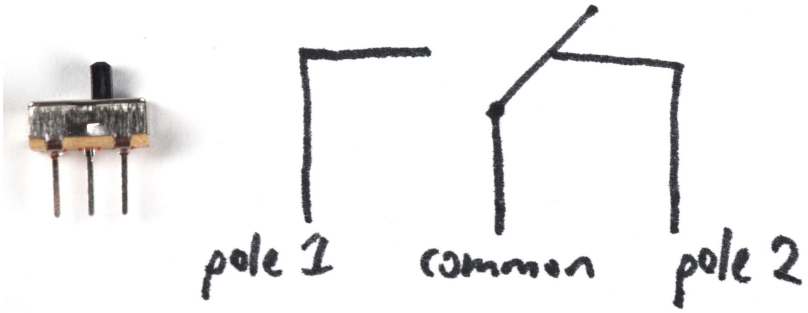
micro lever switch:



lever switch:



single pole double throw (SPDT) switch:



“tactile” pushbutton switch: