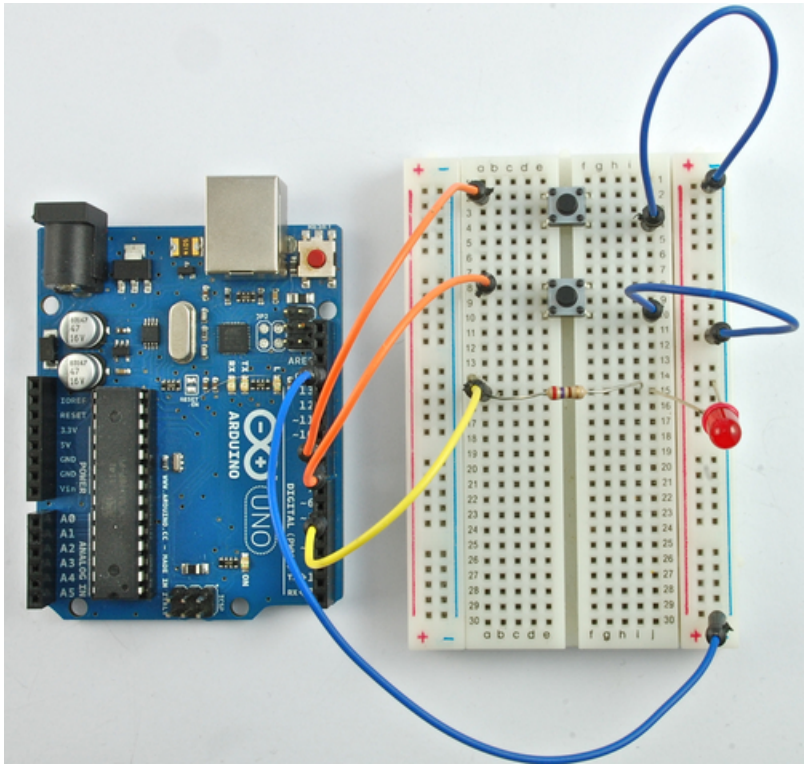
adafruit learning system

Arduino Lesson 6. Digital Inputs

Created by Simon Monk



Last updated on 2018-08-22 03:32:08 PM UTC

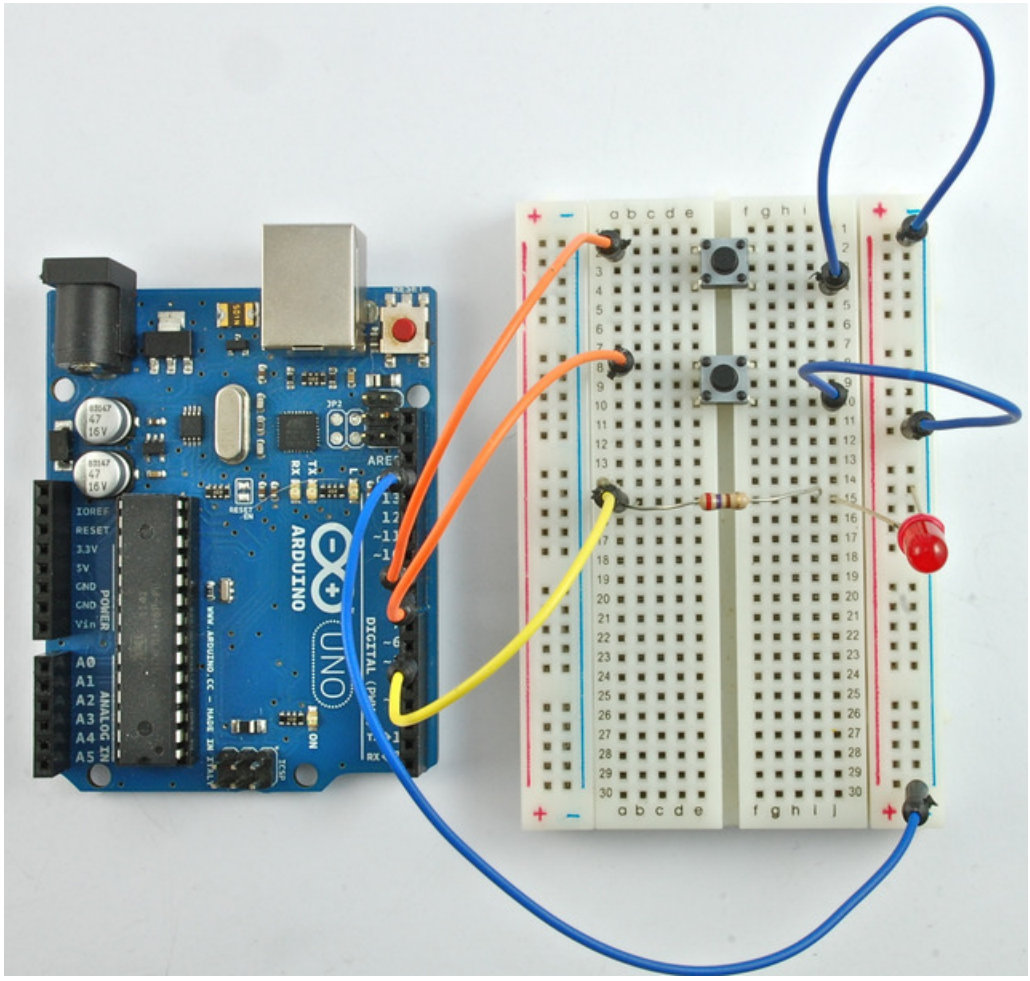
Guide Contents

Guide Contents 2
Overview 3
Parts 4
- Part 4
- Qty 4

Breadboard Layout 5
Arduino Code 6
Push Switches 8
Other Things to Do 9

Overview

In this lesson, you will learn to use push buttons with digital inputs to turn an LED on and off.



Pressing the button nearer the top of the breadboard will turn the LED on, pressing the other button will turn the LED off.

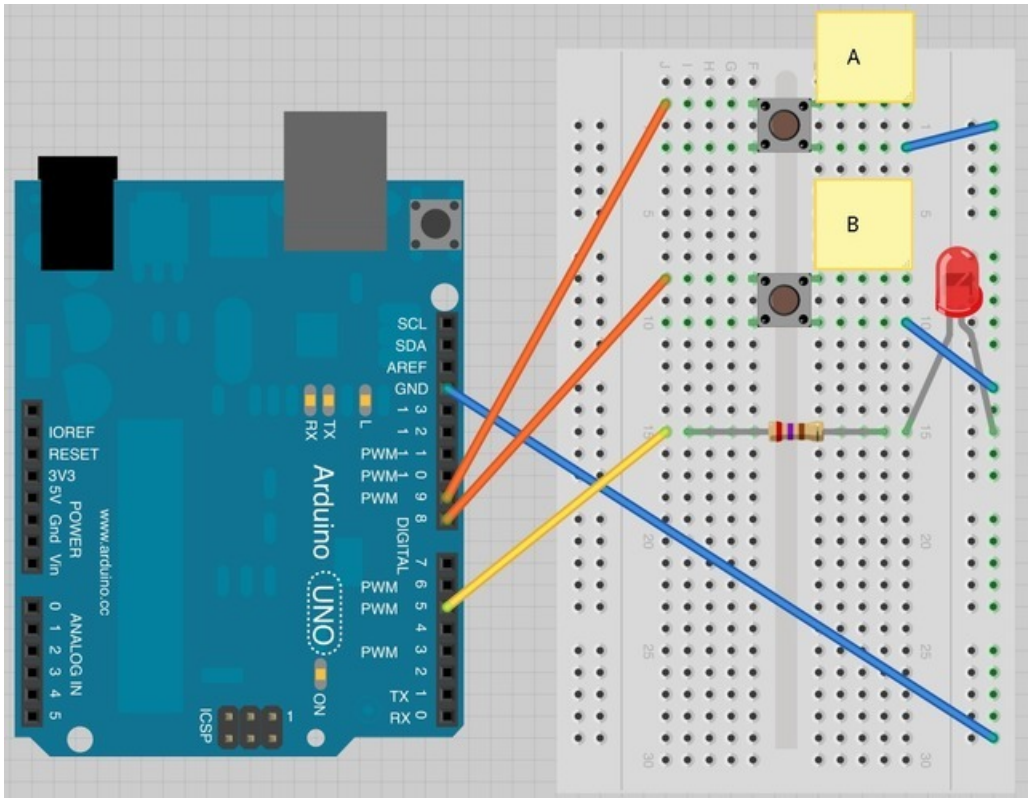
Parts

To complete this lesson, you will need the following parts.

Part	Qty
5mm Red LED	1
270 Ω Resistor (red, purple, brown stripes)	1
Tactile push switch	2
Half-size Breadboard	1
Arduino Uno R3	1
Jumper wire pack	1

Breadboard Layout

Although the bodies of the switches are square, the pins protrude from opposite sides of the switch. This means that the pins will only be far enough apart when they are the correct way around on the breadboard.



Remember that the LED has to be the correct way around with the shorter negative lead to the right.

Arduino Code

Load the following sketch onto your Arduino board. Pressing the top button will turn the LED on, pressing the bottom button will turn it off again.

```
/*
Adafruit Arduino - Lesson 6. Inputs
*/

int ledPin = 5;
int buttonApin = 9;
int buttonBpin = 8;

byte leds = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);
}

void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

The first part of the sketch defines three variable for the three pins that are to be used. The 'ledPin' is the output pin and 'buttonApin' will refer to the switch nearer the top of the breadboard and 'buttonBpin' to the other switch.

The 'setup' function defines the ledPin as being an OUTPUT as normal, but now we have the two inputs to deal with. In this case, we use the set the pinMode to be 'INPUT_PULLUP' like this:

```
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);
```

The pin mode of INPUT_PULLUP means that the pin is to be used as an input, but that if nothing else is connected to the input it should be 'pulled up' to HIGH. In other words, the default value for the input is HIGH, unless it is pulled LOW by the action of pressing the button.

This is why the switches are connected to GND. When a switch is pressed, it connects the input pin to GND, so that it is no longer HIGH.

Since the input is normally HIGH and only goes LOW, when the button is pressed, the logic is a little up-side-down. We will handle this in the 'loop' function.

```
void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

In the 'loop' function there are two 'if' statements. One for each button. Each does an 'digitalRead' on the appropriate input.

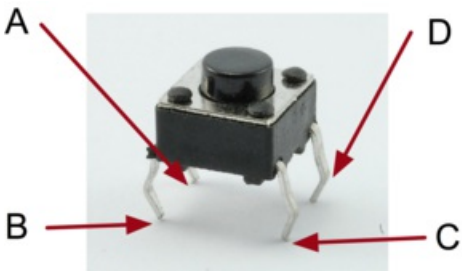
Remember that if the button is pressed, the corresponding input will be LOW, if button A is low, then a 'digitalWrite' on the ledPin turns it on.

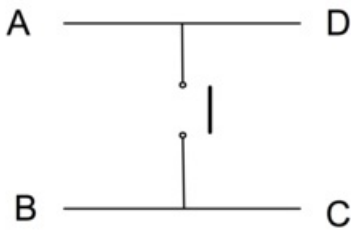
Similarly, if button B is pressed, a LOW is written to the ledPin.

Push Switches

Switches are really simple components. When you press a button or flip a lever, they connect two contacts together so that electricity can flow through them.

The little tactile switches that are used in this lesson have four connections, which can be a little confusing.





Actually, there are only really two electrical connections, as inside the switch package pins B and C are connected together, as are A and D.

Other Things to Do

There are a couple of things we could try with this hardware.

Firstly, you could try taking what you learnt in lesson 5 and adding some commands to the sketch that print something to the Serial Monitor whenever either switch is pressed.

Remember that as well as printing out a message using something like this in your 'loop' function:

```
Serial.println(“Button A Pressed”);
```

You will also need to start serial communication in the 'setup' function by doing this:

```
while (!Serial);
Serial.begin(9600);
```

A second modification that you could make would be to make the buttons do something different. So, for example you could change the sketch so that if button A is pressed, the LED turns on, but then turns off again after 30 seconds.

Hint: Think of this as being a very slow blink.

https://adafru.it/aUv

https://adafru.it/aUv

About the Author

Simon Monk is author of a number of books relating to Open Source Hardware. The following books written by Simon are available from Adafruit: Programming Arduino (http://adafru.it/1019), 30 Arduino Projects for the Evil Genius (http://adafru.it/868) and Programming the Raspberry Pi (https://adafru.it/aM5).