YTP - Module YTP01

Arduino & Micro-Controllers

Teachers Handbook
LESSON 5
Buttons

Goals & Success Factors

Lesson Goal

- To understand how momentary buttons/switches work

Lesson Success Factor(s)

- Build and code a circuit activated with a momentary button

Runsheet (Guideline)

00:00 Introduction
00:05 Goals & Success Factors
00:10 Recap of last week
00:15 Introduction to switches
00:20 The momentary switch
00:30 Build an LED circuit with a switch
00:40 Code/Test/Run the circuit
01:00 Finish

Extension(s)

Add 1 more LED and another switch to your circuit.
Code your circuit so that the second LED turns on.

Students Pre-work

Watch the following video:
Arduino Button Tutorial: bit.ly/ytp01-button-circuit

Teachers Pre-work

Read the following tutorial about buttons:
https://programmingelectronics.com/tutorial-17-using-a-button-old-version/

Watch the following video:
Arduino Button Tutorial: bit.ly/ytp01-button-circuit

Student Homework

Handouts (optional)

Crossword

Please download from:
https://github.com/nexgencodecamp/ytp/tree/master/modules/arduino/05-buttons/handouts

Materials Needed

Arduino IDE

Arduino UNOs
Breadboards
Momentary switches
Jumper wires
LEDs
220 Ohm resistors
10000 Ohm resistors

Handbook

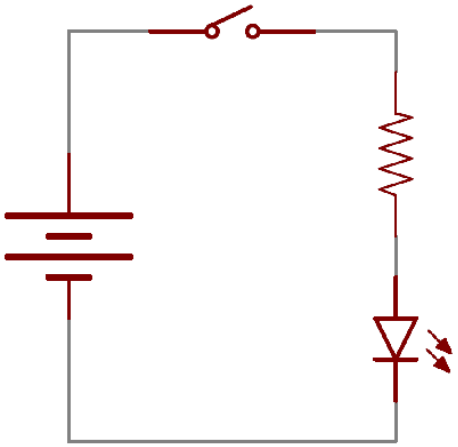
What are Switches?

A switch is a component which controls the open-ness or closed-ness of an electric circuit. They allow control over current flow in a circuit (without having to actually get in there and manually cut or splice the wires). Switches are critical components in any circuit which requires user interaction or control.

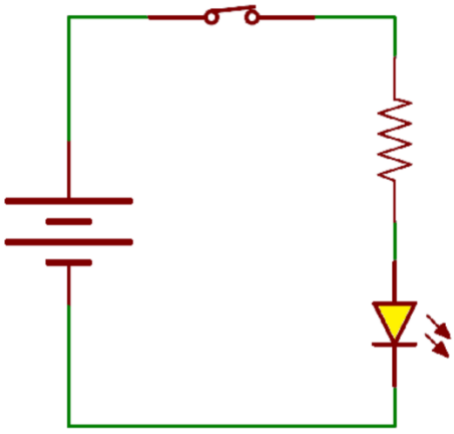
A switch can only exist in one of two states: open or closed. In the **off** state, a switch looks like an open gap in the circuit. This, in effect, looks like an **open circuit**, preventing current from flowing.

In the **on** state, a switch acts just like a piece of perfectly-conducting wire. A short. This **closes the circuit**, turning the system "on" and allowing current to flow unimpeded through the rest of the system.

An open circuit: Current cannot flow through the circuit.



A closed circuit: Current flows through the circuit

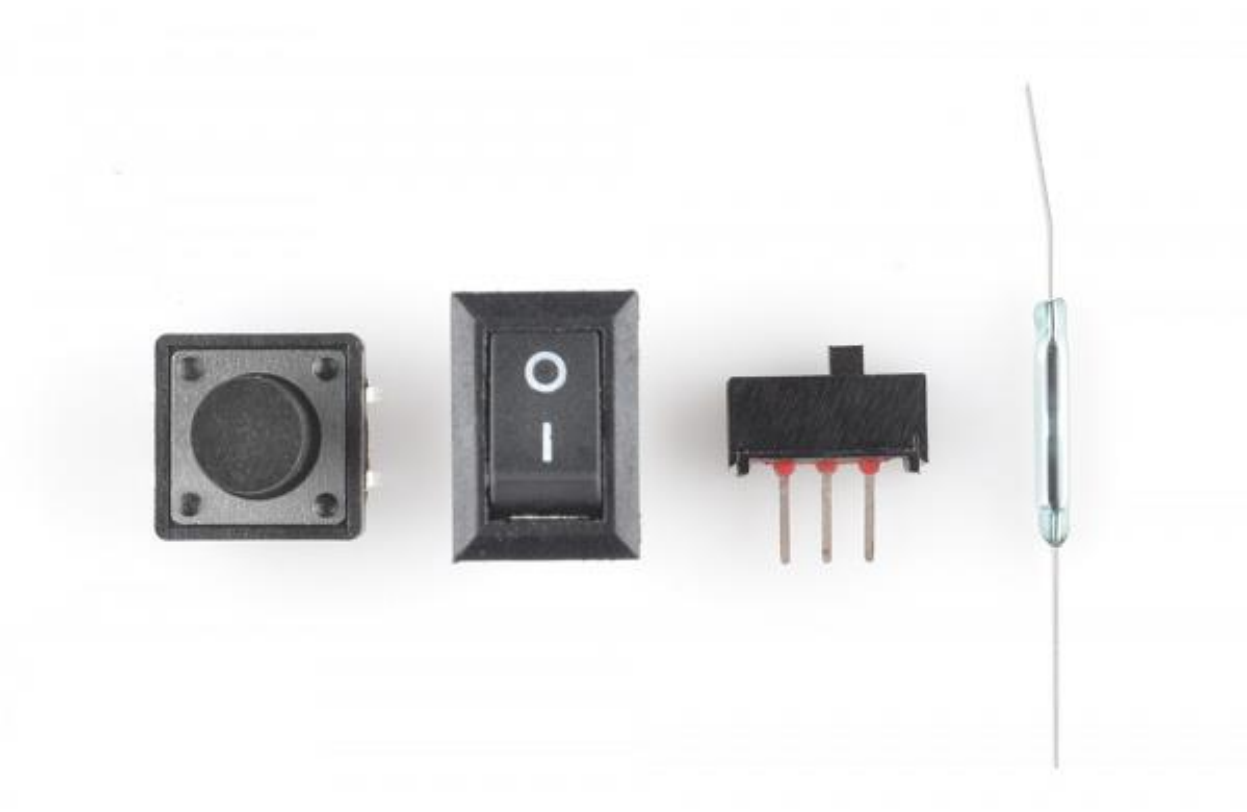


Switch Types

There are many types of switches:

- Push switches
- Rocker switches
- Slider switches
- Electric switches
- Magnetic switches
- Toggle switches
- Momentary switches

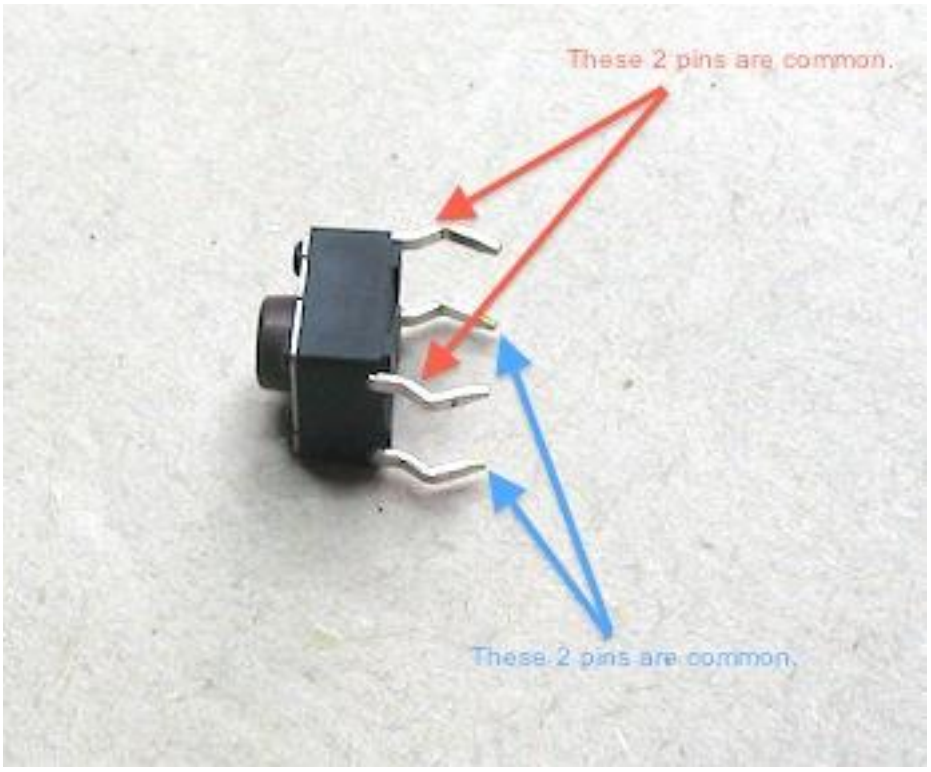
For now, we are only concerned with momentary switches, however, here is a picture of a variety of switches below:



If you look at the first switch from left to right above, that is a momentary switch. It is only on for as long as you keep your finger on it. When you take your finger off, it moves back to the off position. Let's take a closer look at this switch…

A momentary switch has 4 legs. When the switch is in an un-pressed state, 2 legs on one side are electrically connected as are the other 2 legs. However, when the switch is pressed a connection is made across the 2 sets of legs.

Essentially this means that this kind of switch can be used to keep a circuit 'open' until it is pressed. Lets find out how we can 'read' input from button presses.



Using Momentary Buttons as Inputs

When we bring an Arduino to the party we can use a switch such as the one above as an input device. That means the Arduino can tell if the switch is being used as an input device or an output device.

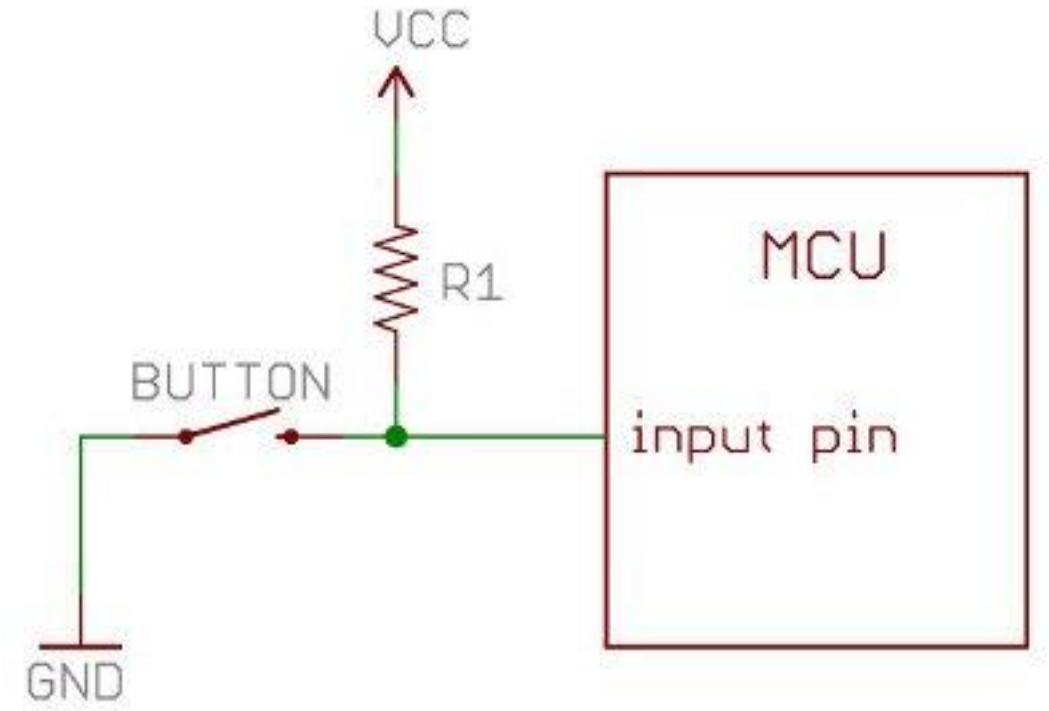
Stop and think!! What push button devices do you use every day? If you stop and think (as we suggested) about this for a minute, you will realise that whenever you press a button on one of these devices, some electronics circuit, code or something must be reading the fact that you just pressed the button… in order to do something.

Such a button must be either on or off – a perfect digital component. However, when we use them in circuits, we must be careful to use them correctly because otherwise they can have fluctuating values where they are neither on nor off. This is explained next.

What is a Pullup 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.



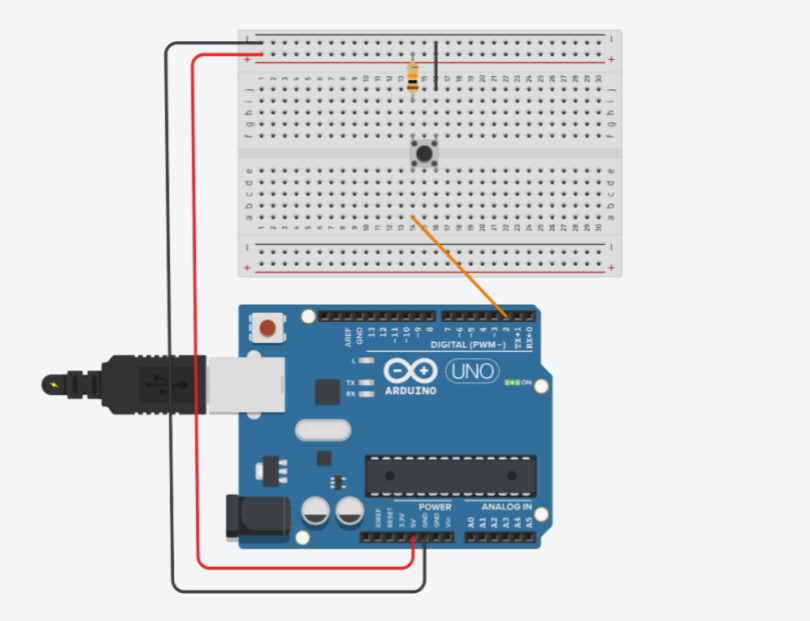
With a pull-up resistor, the input pin will read a high state when the button is not pressed. In other words, a small amount of current is flowing between VCC and the input pin (not to ground), thus the input pin reads close to VCC. When the button is pressed, it connects the input pin directly to ground.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

Let's create a simple circuit with a single button and a pull-up resistor that pulls a digital pin up to 5v. Take a look at the circuit diagram below:



The key thing here is that when the button isn't pressed, it connects pin 2 on the Arduino to the 5v rail via the 10k Ohm resistor. In other words it is 'pulled-up' to a HIGH value of 1.

When the button is pressed, the switch internally connects the two sides together and the current takes the easier path through the switch and up to ground as there is less resistance on that particular side. Now pin 2 has a value of 0 i.e. LOW.

The Arduino code is below:

// Here, I am creating a variable called 'button'
// and setting it to the value 2 because it is going
// to be my button pin and I am using pin 2
int button = 2;

```
void setup(){
  // Here, I am setting up the Serial Monitor
  // so that I can read the value of Pin 2 in the loop() function.
  Serial.begin(9600);

  // Here, I am setting pin 2 to be an INPUT pin
  pinMode(button, INPUT);
}

void loop(){
  // Here we are reading the value of pin 2
  // and setting the result to a new variable called valueOfButton
  int valueOfButton = digitalRead(button);

  // Here we are writing the value to the SerialMonitor
  Serial.println(valueOfButton);

  // Here, we are just waiting a quarter of a second
  // before we continue executing the loop another time.
  delay(250);
}
```

Handouts

Handouts are optional but encouraged. They are normally useful for homework but in some cases you may deem them useful for pre-work depending on the upcoming session content. The handouts are shown here but are alternatively available for download at:

https://github.com/nexgencodecamp/ytp/tree/master/modules/arduino/05-buttons/handouts

Resources

Using a Button Tutorial
https://programmingelectronics.com/tutorial-17-using-a-button-old-version/

Arduino Button Tutorial
bit.ly/ytp01-button-circuit