

Challenge 3 - Button / Crash Sensor

In this challenge you'll be learning how to include a button in a circuit. You'll programme the Arduino so that when the user presses the button, the onboard LED will light up. You'll then learn how to change the circuit to use a crash sensor, which is essentially a button, to perform the same task.

Resources Needed

Hardware:

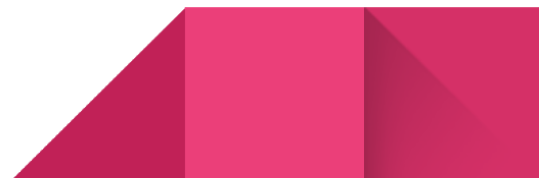
- Arduino Uno,
- Push Button,
- 10KΩ Resistor,
- [Crash Sensor](#),
- Jumper Cables

Software:

- Arduino IDE
- Fritzing

Theory

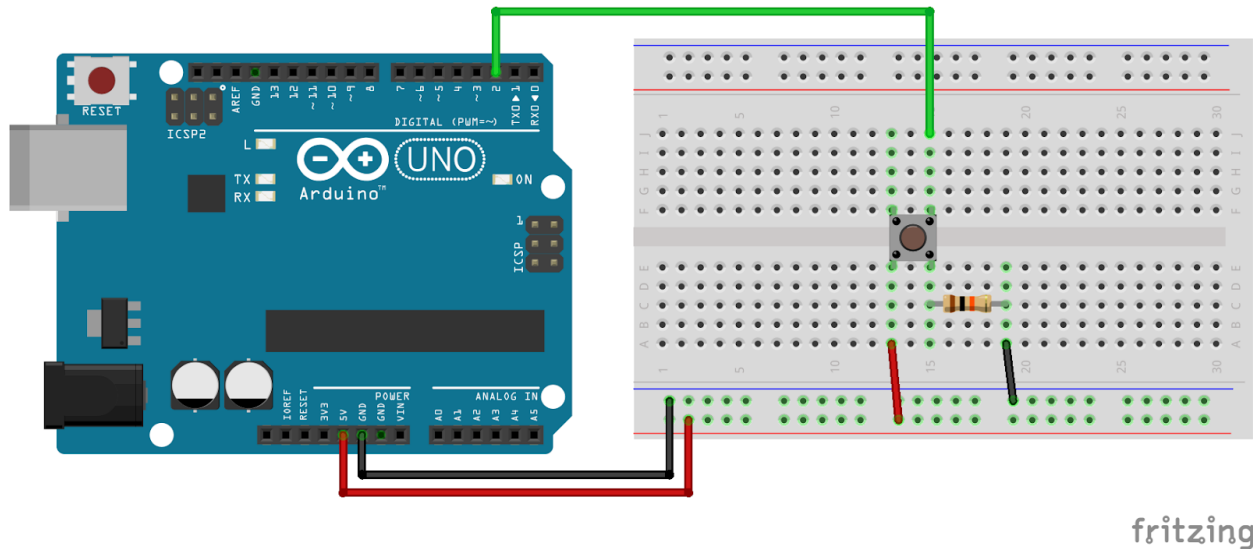
\$(Theory Topics)



Version 1

This first version uses the supplied Arduino Button code to have the onboard LED light up when the pushbutton is pressed down.

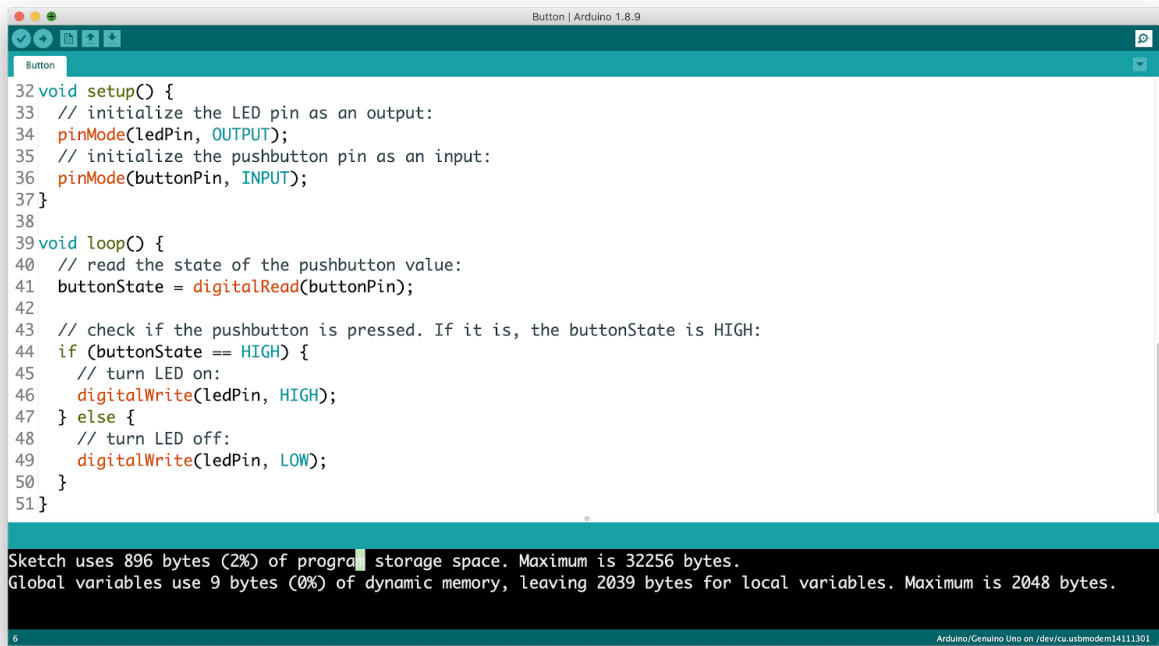
Circuit



Code

The code in this example is the supplied button code in the Arduino IDE. Open it in File -> Examples -> 0.2Digital.

3 - Button / Crash Sensor



```
32 void setup() {
33   // initialize the LED pin as an output:
34   pinMode(ledPin, OUTPUT);
35   // initialize the pushbutton pin as an input:
36   pinMode(buttonPin, INPUT);
37 }
38
39 void loop() {
40   // read the state of the pushbutton value:
41   buttonState = digitalRead(buttonPin);
42
43   // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
44   if (buttonState == HIGH) {
45     // turn LED on:
46     digitalWrite(ledPin, HIGH);
47   } else {
48     // turn LED off:
49     digitalWrite(ledPin, LOW);
50   }
51 }
```

Sketch uses 896 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

6 Arduino/Genuino Uno on /dev/cu.usbmodem14111301

/*

Button

Turns on and off a light emitting diode(LED) connected to digital pin 13, when pressing a pushbutton attached to pin 2.

The circuit:

- LED attached from pin 13 to ground
- pushbutton attached to pin 2 from +5V
- 10K resistor attached to pin 2 from ground
- Note: on most Arduinos there is already an LED on the board attached to pin 13.

created 2005

by DojoDave <<http://www.0j0.org>>

modified 30 Aug 2011

by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/Button>

*/

// constants won't change. They're used here to set pin numbers:

const int buttonPin = 2; // the number of the pushbutton pin

const int ledPin = 13; // the number of the LED pin

// variables will change:

int buttonState = 0; // variable for reading the pushbutton status

void setup() {

// initialize the LED pin as an output:

pinMode(ledPin, OUTPUT);

// initialize the pushbutton pin as an input:

pinMode(buttonPin, INPUT);

}

void loop() {

// read the state of the pushbutton value:

buttonState = digitalRead(buttonPin);

// check if the pushbutton is pressed. If it is, the buttonState is HIGH:

if (buttonState == HIGH) {

// turn LED on:

digitalWrite(ledPin, HIGH);

} else {

// turn LED off:

digitalWrite(ledPin, LOW);

}

```
}
```

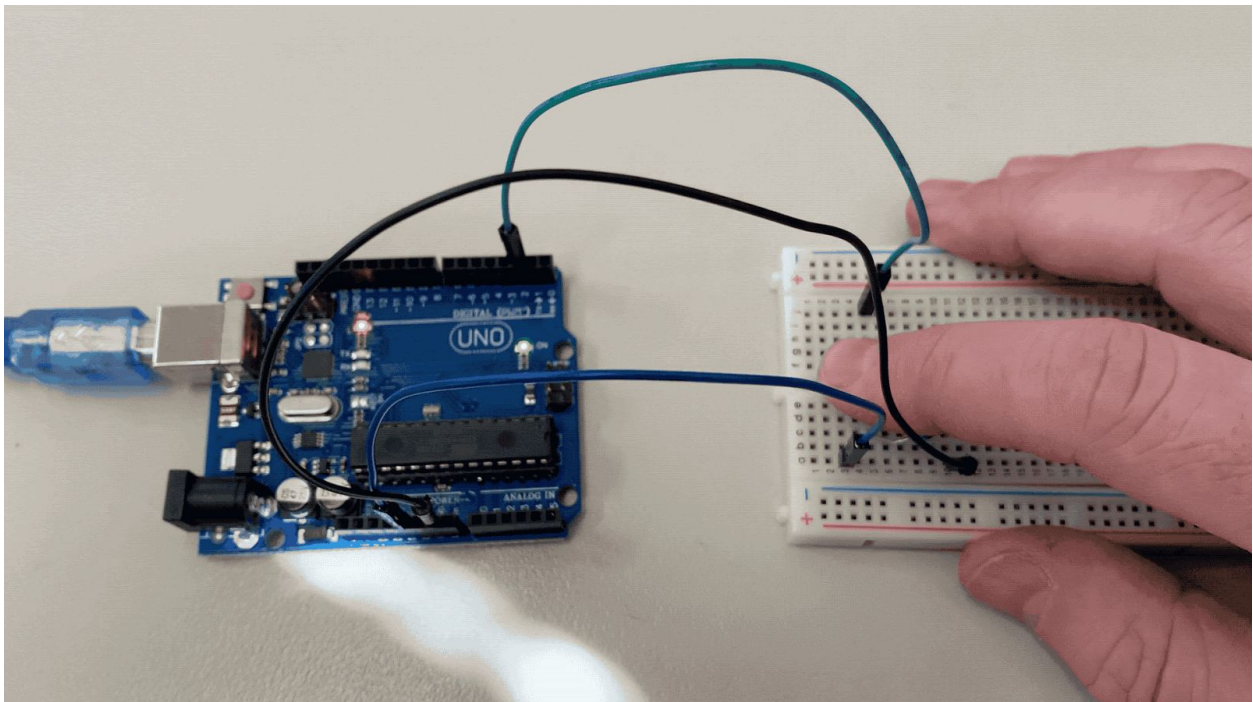
Code Explanation

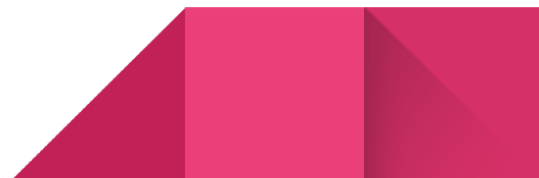
The code is relatively straight forward. After defining the pins for the LED and the button, it sets them to OUTPUT and INPUT as appropriate in the `setup()` function.

An addition variable is declared called `buttonState` and this will be used in the `loop()` function to store the state of the button (pressed or not pressed). `buttonState` is tested in the `if` block to determine if it's HIGH (the button is being pressed). If that's true, then HIGH is written to the `ledPin`, turning the LED on. If the value of `buttonState` is not HIGH (i.e. it is LOW) then the LOW is written to the LED pin.

Process

Publish the code to the arduino and test it.





Version 2

This version is similar to the first, except the pushbutton is replaced by a crash sensor, which modifies the circuit to accommodate the new input, and changes the code to have the onboard LED alternate on and off when the button is pressed.

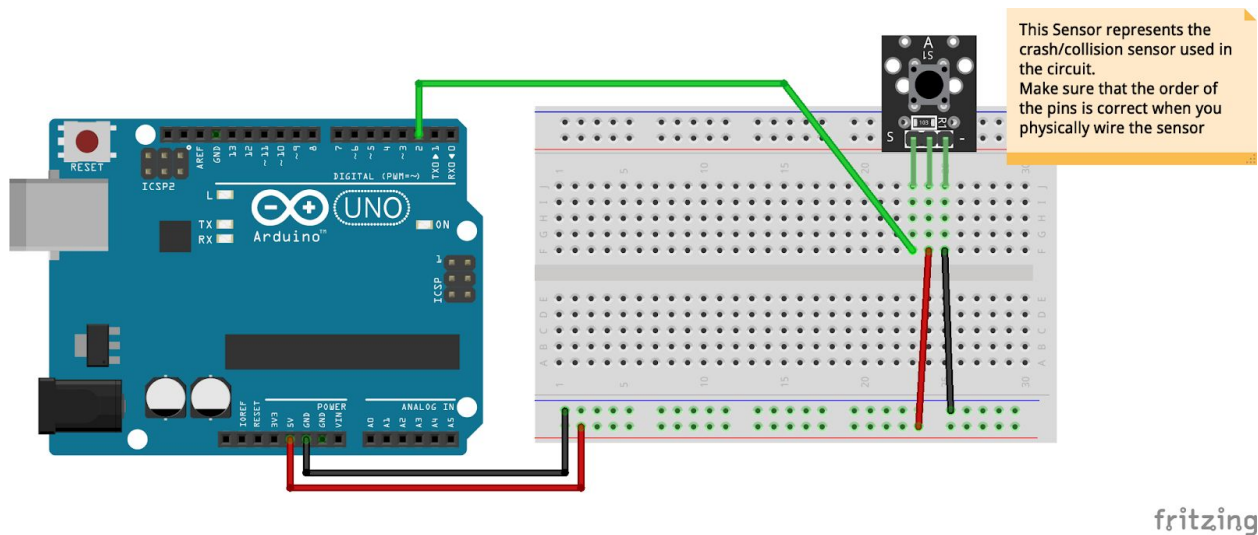
The crash sensor used acts as a simple button, but with a wire protruding from the front. This type of sensor can be used in Robots to detect if the robot has collided with another object.

There are different types of crash sensors, so you must be careful when you wire them to the arduino. The one used in this tutorial is the one shown below.



The ordering of the pins on the physical device is different from the wiring diagram - it's just something to be aware of and adjust your wiring as necessary.

Circuit



fritzing

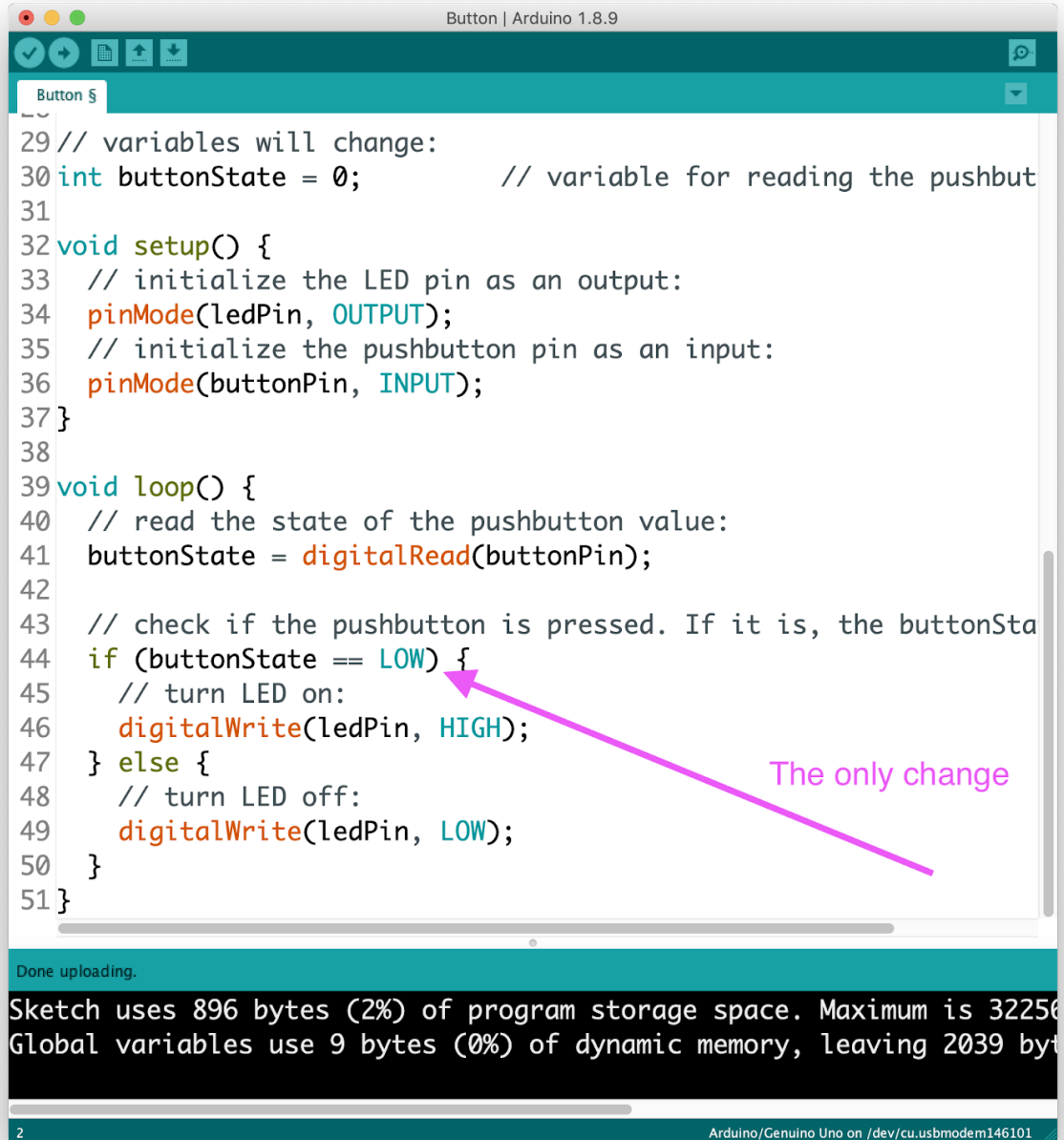
In fritzing, there is no equivalent crash sensor, so a button sensor is used in the diagram above. Functionally this is exactly the same as the crash sensor, so the remainder of the logic is correct.

Code

The code user here is a slightly modified version of the previous code. When the crash sensor is wired, it outputs the *opposite* value that the previous button did.

I.E. in the previous example, if the button is not being pushed, the circuit was reading LOW on pin 2. This meant that the LED attached to pin 13 was not lit. When the button was pushed, pin 2 was reading HIGH meaning the LED was turned on.

In this case, the output from the crash sensor when not being pushed is HIGH, meaning the LED is on. The code has to be changed to match the functionality of the first example.



```

Button $
29 // variables will change:
30 int buttonState = 0;          // variable for reading the pushbut
31
32 void setup() {
33   // initialize the LED pin as an output:
34   pinMode(ledPin, OUTPUT);
35   // initialize the pushbutton pin as an input:
36   pinMode(buttonPin, INPUT);
37 }
38
39 void loop() {
40   // read the state of the pushbutton value:
41   buttonState = digitalRead(buttonPin);
42
43   // check if the pushbutton is pressed. If it is, the buttonSta
44   if (buttonState == LOW) {
45     // turn LED on:
46     digitalWrite(ledPin, HIGH);
47   } else {
48     // turn LED off:
49     digitalWrite(ledPin, LOW);
50   }
51 }

```

Done uploading.

Sketch uses 896 bytes (2%) of program storage space. Maximum is 32256.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes.

2 Arduino/Genuino Uno on /dev/cu.usbmodem146101

/*

Button

Turns on and off a light emitting diode(LED) connected to digital pin 13,

when pressing a pushbutton attached to pin 2.

The circuit:

- LED attached from pin 13 to ground
 - pushbutton attached to pin 2 from +5V
 - 10K resistor attached to pin 2 from ground
- Note: on most Arduinos there is already an LED on the board attached to pin 13.

created 2005

by DojoDave <<http://www.0j0.org>>

modified 30 Aug 2011

by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/Button>

*/

// constants won't change. They're used here to set pin numbers:

const int buttonPin = 2; // the number of the pushbutton pin

const int ledPin = 13; // the number of the LED pin

// variables will change:

int buttonState = 0; // variable for reading the pushbutton status

void setup() {

// initialize the LED pin as an output:

pinMode(ledPin, OUTPUT);

// initialize the pushbutton pin as an input:

pinMode(buttonPin, INPUT);

}

```
void loop() {  
  // read the state of the pushbutton value:  
  buttonState = digitalRead(buttonPin);  
  
  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:  
  if (buttonState == LOW) {  
    // turn LED on:  
    digitalWrite(ledPin, HIGH);  
  } else {  
    // turn LED off:  
    digitalWrite(ledPin, LOW);  
  }  
}
```

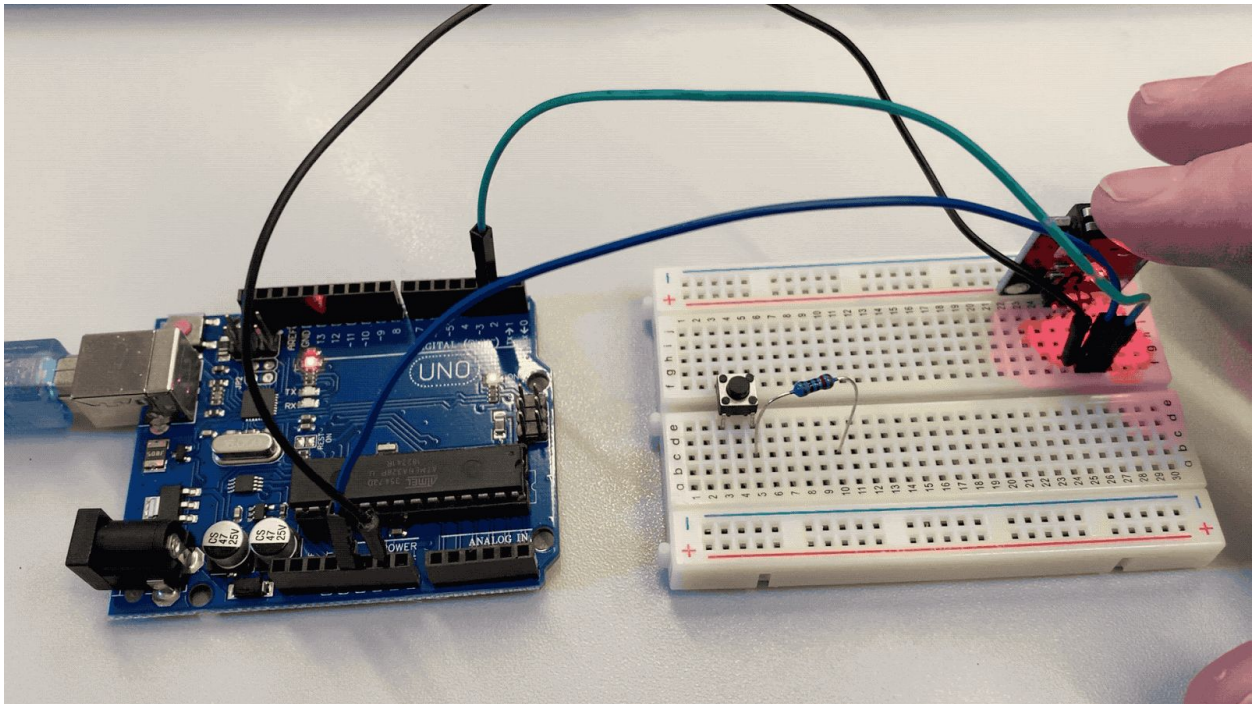
Code Explanation

As the code has only changed slightly, the logic is exactly the same apart from the change already described.

Process

Publish the code to the arduino and test it.

3 - Button / Crash Sensor



Challenge Extension

#{Challenge Extension Description}

Circuit

TODO: Circuit

Code

```
#{Challenge Code}
```

Code Explanation

TODO: Code Explanation

Process

TODO: Describe Process