Aberystwyth Robotics Club - Ardunio Tutorials - Button Tutorial

Introduction

This example shows you how to monitor the state of a switch by establishing serial communication between your Arduino or Genuino and your computer over USB.

Hardware Required

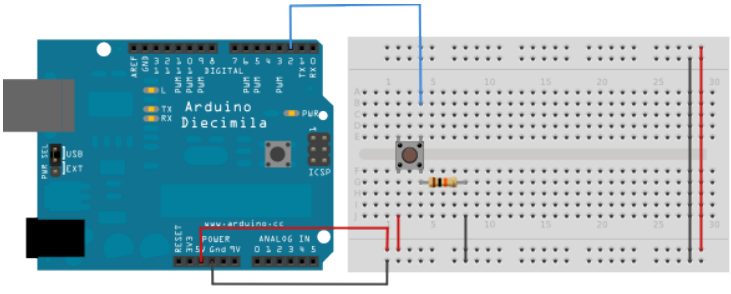
- Arduino or Genuino Board
- A momentary switch, button, or toggle switch
- 10k ohm resistor
- hook-up wires
- breadboard

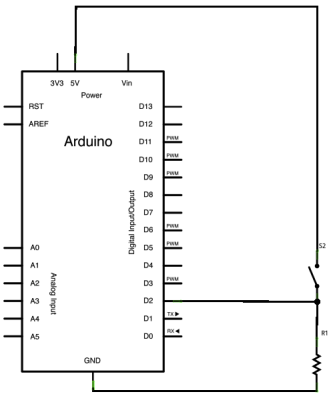
Circuit

1. **Connect two wires, red and black, to the two long vertical rows on the side of the breadboard.**

   The wires connect to the rows on the side of the breadboard to provide access to the 5 volt supply and ground.

2. **Connect a third wire from digital pin 2 to one leg of the push button.**
3. **Connect the same leg of the push button to ground through a pull-down resistor (here 10k ohm).**
4. **Connect the other leg of the button to the 5 volt supply.**





Circuit and Schematic https://www.arduino.cc/en/tutorial/blink

When the push button is open (unpressed) there is no connection between the two legs of the push button, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that we read a HIGH. (This may be represented as a 1 and a 0 in Arduino).

You can also wire this circuit the opposite way, with a pull-up resistor keeping the input HIGH, and going LOW when the button is pressed. If so, the behaviour of the sketch will be reversed, with the LED normally on and turning off when you press the button.

If you disconnect the digital i/o pin from everything, the LED may blink erratically. This is because the input is "floating" - that is, it doesn't have a solid connection to voltage or ground, and it will randomly return to either HIGH or LOW. That's why you need a pull-down resistor in the circuit.

Code

After building the circuit, you will need to enter the code featured below (https://www.arduino.cc/en/tutorial/button).

1. **Plug in your Arduino or Genuino board into your computer.**
2. **Start the Arduino Software (IDE).**
3. **In the setup function, begin serial communications, at 9600 bits of data per second, between your board your computer with the line:**

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

4. **In the setup function, initialise digital pin 2, the pin that will read the output from your button, as an input:**

```
pinMode(2,INPUT);
```

5. **Move into the main loop of your code.**
6. **Establish an integer variable to hold information coming in from your switch. Call this variable sensorValue and set it equal to whatever is being read on pin 2:**

```
int sensorValue = digitalRead(2);
```

   When your button is pressed, 5 volts will freely flow through your circuit, and when it is not pressed, the input pin will be connected to the ground through the 10 ohm resistor. This is a digital input, meaning that the switch can only be in either an on state (seen by your Arduino as a "1", or HIGH) or an off state (seen by your Arduino as a "0" or LOW), with nothing in between.

   Since the information coming in from the switch will be either a "1" or a "0", you can use an int datatype for your variable.

7. **Make the board print the input back to the computer as a decimal value:**

```
Serial.println(sensorValue);
```

   Now, when you open your Serial Monitor in the Arduino Software (IDE), you will see a stream of "0"s if your switch is open, or "1"s if your switch is closed.

Example Code

```
int pushButton = 2; // digital pin 2 has a pushbutton attached to it.

void setup() {
  Serial.begin(9600);//initialize serial communication at 9600 bits per
     second:
  pinMode(pushButton, INPUT); // make the pushbutton's pin an input:
  }

void loop() { // the loop function runs over and over again forever
  int buttonState = digitalRead(pushButton); // read the input pin:
  Serial.println(buttonState); // print out the state of the button:
  delay(1); // delay in between reads for stability
  }
```