Interrupt Overview

Extenal interrupts are useful for making things happen automatically in microcontroller programs and can help solve timing problems. Good tasks for using an interrupt may include reading a rotary encoder, or monitoring user input.

If you wanted to ensure that a program always caught the pulses from a external devices like button or sensor, so that it never misses a pulse, it would make it very tricky to write a program to do anything else, because the program would need to constantly poll the sensor lines for the signal, in order to catch pulses when they occurred. Other sensors have a similar interface dynamic too, such as trying to read a sound sensor that is trying to catch a click, or an infrared slot sensor (photo-interrupter) trying to catch a coin drop. In all of these situations, using an interrupt can free the microcontroller to get some other work done while not missing the input.

About Interrupt Service Routines

ISRs are special kinds of functions that have some unique limitations most other functions do not have. An ISR cannot have any parameters, and they shouldn’t return anything.

ISR

loop()

function

Interrupt

call

return

Generally, an ISR should be as short and fast as possible. If your sketch uses multiple ISRs, only one can run at a time, other interrupts will be executed after the current one finishes in an order that depends on the priority they have. millis() relies on interrupts to count, so it will never increment inside an ISR. Since delay() requires interrupts to work, it will not work if called inside an ISR. micros() works initially but will start behaving erratically after 1-2 ms. delayMicroseconds() does not use any counter, so it will work as normal.

Typically global variables are used to pass data between an ISR and the main program. To make sure variables shared between an ISR and the main program are updated correctly, declare them as volatile.

How to use external interrupt on Arduino

**Digital Pins With Interrupts**

There are only two external interrupt pin in arduino uno. They are Digital pin 2 and Digital pin 3. To use external interrupt on Arduino you should call function attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)

**Parameters**

interrupt: the number of the interrupt. Allowed data types: int.

pin: the Arduino pin number.

ISR: the ISR to call when the interrupt occurs, this function must take no parameters and return nothing. This function is sometimes referred to as an in terrupt service routine.

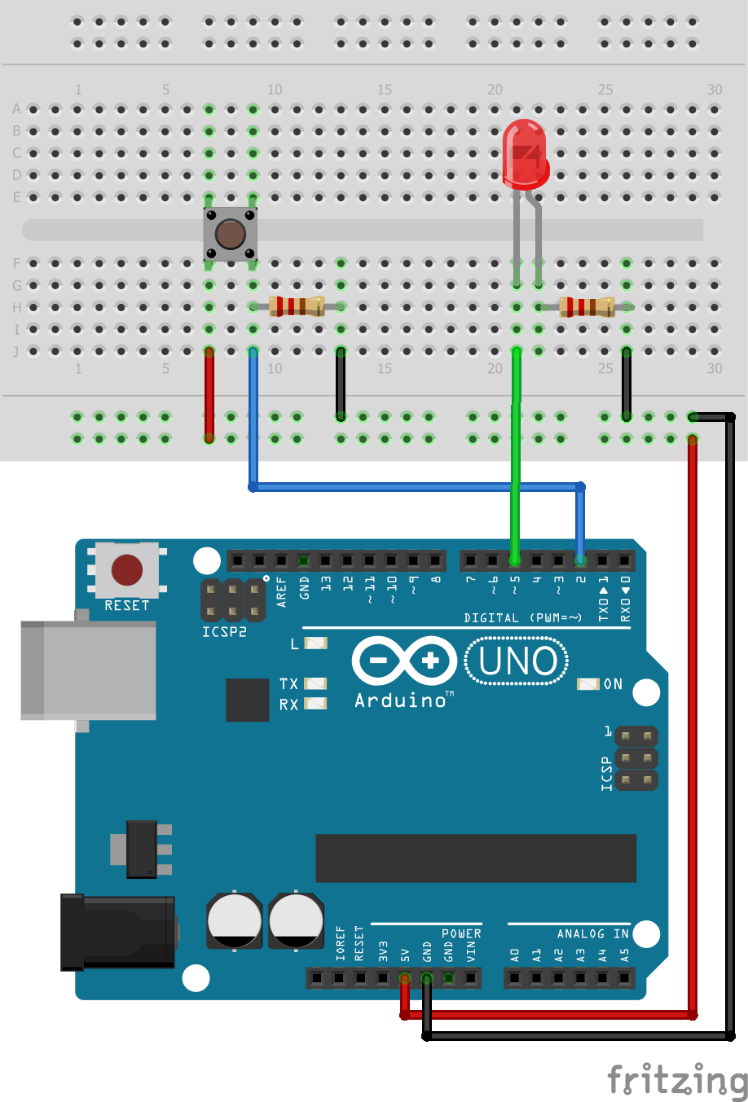
mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:

* **LOW** to trigger the interrupt whenever the pin is low,
* **CHANGE** to trigger the interrupt whenever the pin changes value
* **RISING** to trigger when the pin goes from low to high,
* **FALLING** for when the pin goes from high to low.
* **HIGH** to trigger the interrupt whenever the pin is high.

Hardware Required

* Arduino or Genuino Board
* Momentary button or Switch
* LED
* 330 ohm resistor
* hook-up wires
* breadboard

Circuit



Code

// define led pin is 5 and interrupt pin is 0, that mean we will use pin 2 for interrupt

const int ledPin = 5, int0Pin = 2;

void setup(){

 pinMode(ledPin, OUTPUT);

 // setup for interrupt 0 with rising mode; ISR is blink function.

 attachInterrupt(digitalPinToInterrupt(int0Pin), blink, RISING);

}

void loop(){

 //Add your task

 delay(10000);

 blink();

}

void blink(){

 digitalWrite(ledPin, !digitalRead(ledPin)); // Toggle LED on/off if interrupt happen.

}