











A simple example of Interrupts is touch screen mobile phones which has the highest priority to the “Touch” sense. Almost every electronic device has some kind to interrupts to ‘interrupt’ the regular process and do some higher priority things on particular event. The regular process is resumed after serving the Interrupt.

So technically, **Interrupts is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and gets itself serviced like it has higher priority**. For example a processor doing a normal execution can be interrupted by some sensor to execute a particular process that is present in ISR (Interrupt Service Routine). After executing the ISR processor can again resume the normal execution.



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 insure that a program always caught the pulses from a rotary encoder, 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 encoder, 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.



There are two types of interrupts:-

* **Hardware Interrupt:** It happens when an external event is occurred like an external interrupt pin changes its state from LOW to HIGH or HIGH to LOW.
* **Software Interrupt:** It happens according to the instruction from the software. For example Timer interrupts are software interrupt.

On basis of classification we will see how to use **interrupts in Arduino** Board. It has two types of interrupts:

* External Interrupt
* Pin Change Interrupt

# External Interrupt:

These interrupt are interpreted by hardware and are very fast. These interrupts can be set to trigger on the event of RISING or FALLING or LOW levels.

|  |  |
| --- | --- |
| **Arduino Board** | **External Interrupt pins:** |
| Uno , Nano, Mini | 2, 3 |
| Mega | 2, 3, 18, 19, 20, 21 |
| Micro | 0, 1, 2, 3, 7 |

# Pin Change Interrupts:

Arduinos can have more interrupt pins enabled by using pin change interrupts. In at mega 168/328 based Arduino boards any pins or all the 20 signal pins can be used as interrupt pins. They can also be triggered using RISING or FALLING edges.



In order to use interrupts in Arduino the following concepts are need to be understood.

# Interrupt Service Routine (ISR):

Interrupt Service Routine or an Interrupt handler is an event that has small set of instructions in it. When an external interrupt occurs, the processor first executes these code that is present in ISR and returns back to state where it left the normal execution.

ISR has following **syntax in Arduino:**

**attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)**;

# digitalPinToInterrupt(pin):

In Arduino Uno, Nano, Mini the pins used for interrupt are 2,3 in Micro 0,1,2,3,7 and Mega 2,3,18,19,20,21. Specify the input pin that is used for external interrupt here.

# Mode:

Type of transition to trigger on, are rising, falling and change-

* RISING: To trigger an interrupt when the pin transits from LOW to HIGH.
* FALLING: To trigger an interrupt when the pin transits from HIGH to LOW.
* CHANGE: To trigger an interrupt when the pin transits from LOW to HIGH or HIGH to LOW (i.e., when the pin state changes ).

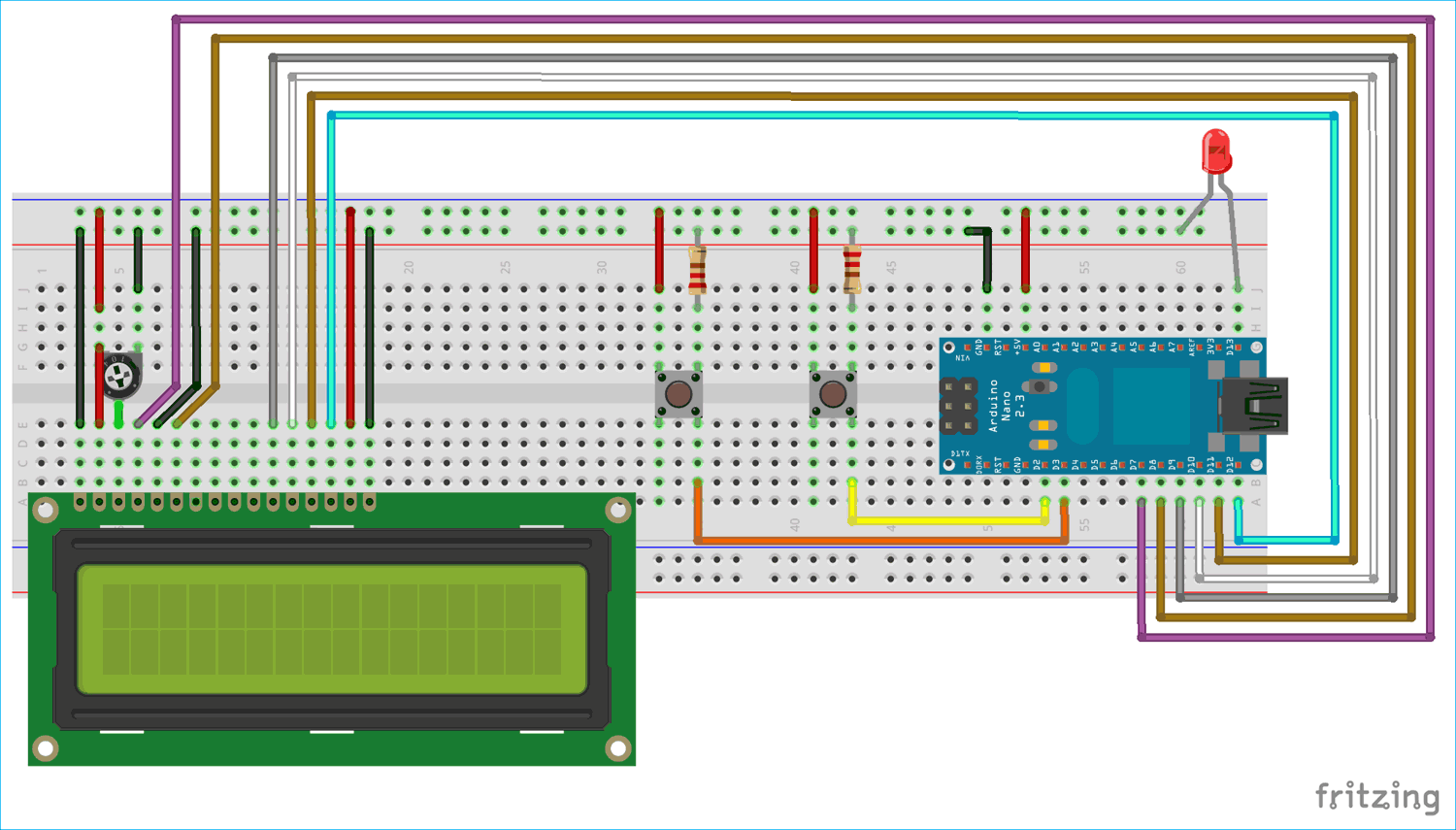
# Some Conditions while using Interrupt

* Interrupt Service Routine function (ISR) must be as short as possible.
* Delay () function doesn’t work inside ISR and should be avoided.



* Arduino Board (I have used Arduino NANO)
* Push button - 2
* LED - 1
* Resistor (10K) - 2
* LCD (16x2) - 1
* Bread Board
* Connecting Wires





**Circuit Connection between Arduino Nano and 16x2 LCD display:**

|  |  |
| --- | --- |
| **LCD** | **Arduino Nano** |
| VSS | GND |
| VDD | +5V |
| V0 | To Potentiometer Centre PIN  For Controlling Contrast of the LCD |
| RS | D7 |
| RW | GND |
| E | D8 |
| D4 | D9 |
| D5 | D10 |
| D6 | D11 |
| D7 | D12 |
| A | +5V |
| K | GND |

Two push buttons are connected to Arduino Nano at pin D2 & D3. They are used for using two external interrupts, one for turning LED ON and another for turning OFF a LED.

Each push button has a pull down resistor of 10k connected to ground. So when push button is pressed it is logic HIGH (1) and when not pressed it is logic LOW (0). A Pull down resistor is compulsory otherwise there will be floating values at the input pin D2 & D3.

A LED is also used to indicate that a Interrupt has been triggered or a button has been pressed.



In this example a number is incremented from 0 which displays continuously in (16x2) LCD connected to the Arduino Nano, whenever the left push button (interrupt pin D3) is pressed the LED goes ON and display shows Interrupt2, and when the right push button (interrupt pin D2) is pressed the LED goes OFF and display shows Interrupt1.

# Code :

//Interrupts using Arduino

#include<LiquidCrystal.h> LiquidCrystal lcd (7,8,9,10,11,12);

volatile int output = LOW; int i = 0;

void setup()

{

lcd.begin(16,2); lcd.setCursor(0,0); lcd.print("CIRCUIT DIGEST");

lcd.setCursor(0,1); lcd.print("ArduinoInterrupt"); delay(3000);

lcd.clear(); pinMode(13,OUTPUT);

// Including lcd display library

// Define LCD display pins RS,E,D4,D5,D6,D7

// setting LCD as 16x2 type

attachInterrupt(digitalPinToInterrupt(2),buttonPressed1,RISING); // function for creating external interrupts at pin2 on Rising (LOW to HIGH) attachInterrupt(digitalPinToInterrupt(3),buttonPressed2,RISING); // function for creating external interrupts at pin3 on Rising (LOW to HIGH)

}

void loop()

{

lcd.clear(); lcd.print("COUNTER:"); lcd.print(i);

++i;

delay(1000);

digitalWrite(13,output); //Turns LED ON or OFF depending upon output value

}

void buttonPressed1()

{

output = LOW; lcd.setCursor(0,1); lcd.print("Interrupt 1");

}

void buttonPressed2() pressed

{

output = HIGH; lcd.setCursor(0,1); lcd.print("Interrupt2");

}

//ISR function excutes when push button at pinD2 is pressed

//Change Output value to LOW

//ISR function excutes when push button at pinD2 is

//Change Output value to HIGH

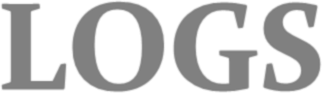


Interrupts are a simple way to make your system more responsive to time sensitive tasks. They also have the added benefit of freeing up your main

`loop ()` to focus on some primary task in the system. It’s found that this tends to make codes a little more organized when they are used - it's easier to see what the main chunk of code was designed for, while the interrupts handle periodic events. The examples shown in the document are just about the most basic cases for using an interrupt - we can use them for reading an I2C devices, sending or receiving wireless data, or even starting or stopping a motor.



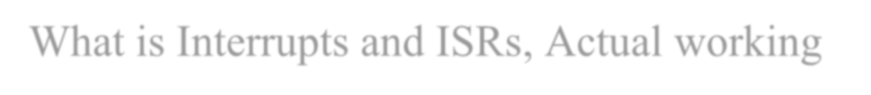
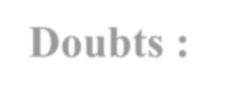
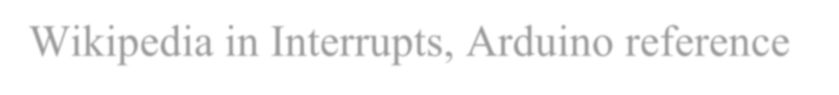
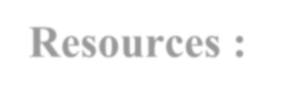
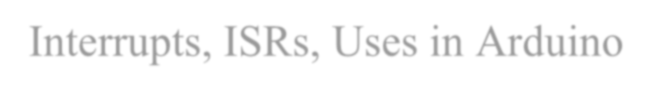
* <https://en.wikipedia.org/wiki/Interrupt>
* <https://en.wikipedia.org/wiki/Interrupt_handler>
* [https://circuitdigest.com/microcontroller-projects/arduino-interrupt-tutorial- with-examples](https://circuitdigest.com/microcontroller-projects/arduino-interrupt-tutorial-with-examples)
* [https://www.arduino.cc/reference/en/language/functions/external- interrupts/attachinterrupt/](https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/)
* [https://www.allaboutcircuits.com/technical-articles/using-interrupts-on- arduino/](https://www.allaboutcircuits.com/technical-articles/using-interrupts-on-arduino/)
* <https://www.tutorialspoint.com/arduino/arduino_interrupts.htm>
* <https://www.instructables.com/id/Arduino-Interrupts/>
* <https://www.youtube.com/watch?v=J2J8vRWx96k>



**INTERRUPTS IN ARDUINO**



**20 / 05 / 2019**



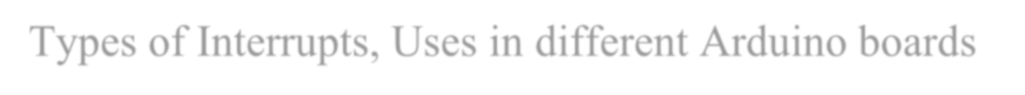
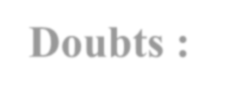
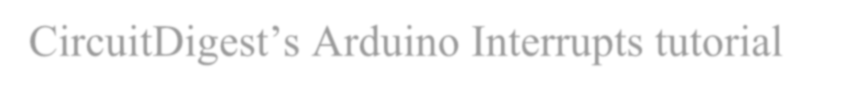
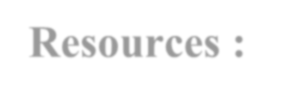
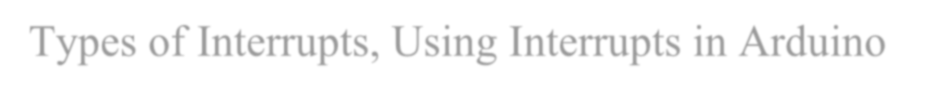
**Topics :** Interrupts, ISRs, Uses in Arduino

**Resources :** Wikipedia in Interrupts, Arduino reference

**Doubts :** What is Interrupts and ISRs, Actual working



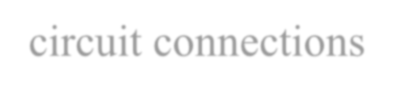
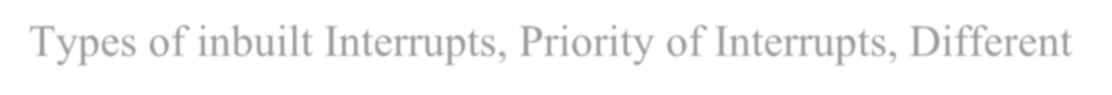
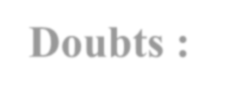
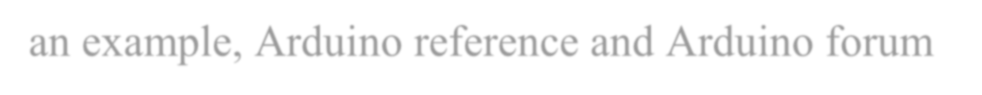
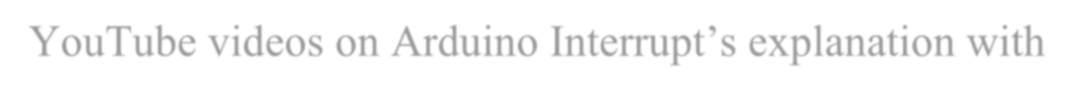
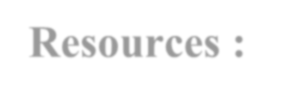
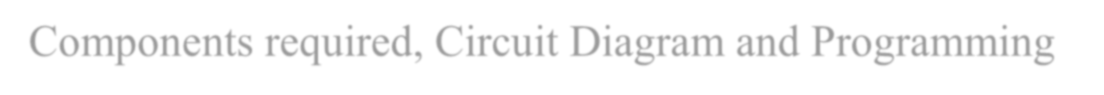
**22 / 05 / 2019**



**Topics :** Types of Interrupts, Using Interrupts in Arduino **Resources :** CircuitDigest’s Arduino Interrupts tutorial **Doubts :** Types of Interrupts, Uses in different Arduino boards



**23 / 05 / 2019**



**Topics :** Components required, Circuit Diagram and Programming

**Resources :** YouTube videos on Arduino Interrupt’s explanation with an example, Arduino reference and Arduino forum

**Doubts :** Types of inbuilt Interrupts, Priority of Interrupts, Different circuit connections