Interruption avec Arduino

Considérons une voiture qui se déplace rapidement, si elle est soudainement frappée par une autre voiture dans la direction opposée, la première chose qui se passe est que, le capteur accéléromètre présent dans la voiture détecte une décélération soudaine et déclenche une interruption externe au microcontrôleur présent dans la voiture. En fonction de cette interruption, le microcontrôleur produit un signal électrique pour déployer immédiatement les airbags. Les microcontrôleurs présents dans la voiture surveillent de nombreuses choses simultanément, comme la détection de la vitesse de la voiture, la vérification d'autres capteurs, le contrôle de la température du climatiseur, etc. Alors qu'est-ce qui provoque l'ouverture soudaine d'un airbag en quelques secondes ? La réponse est l'interruption, un signal d'interruption est utilisé ici qui a la plus haute priorité de tous.

Un autre exemple simple d'interruptions est celui des téléphones portables à écran tactile, qui accordent la plus haute priorité au sens du "toucher". Presque tous les appareils électroniques sont dotés d'un type d'interruptions permettant d'interrompre le processus normal et de faire des choses plus prioritaires lors d'un événement particulier. Le processus normal est repris après avoir servi l'interruption.

Techniquement, les interruptions sont donc un mécanisme par lequel une entrée/sortie ou une instruction peut suspendre l'exécution normale du processeur et être traitée comme si elle avait une priorité plus élevée. Par exemple, un processeur effectuant une exécution normale peut être interrompu par un capteur pour exécuter un processus particulier qui est présent dans l'ISR (Interrupt Service Routine). Après avoir exécuté l'ISR, le processeur peut reprendre son exécution normale.

Types d'interruptions

Il existe deux types d'interruptions :

L'interruption matérielle : Elle se produit lorsqu'un événement externe se produit, par exemple lorsqu'une broche d'interruption externe change d'état de BASSE à HAUTE ou de HAUTE à BASSE.

Interruption logicielle : Elle se produit selon l'instruction du logiciel. Par exemple, les interruptions de minuterie sont des interruptions logicielles.

Interruptions dans Arduino

Nous allons maintenant voir comment utiliser les interruptions dans la carte Arduino. Il existe deux types d'interruptions :

Interruption externe

Interruption pour changement de broche

Interruption externe :

Ces interruptions sont interprétées par le matériel et sont très rapides. Ces interruptions peuvent être paramétrées pour se déclencher en cas de montée, de descente ou de niveau bas.

Arduino Board	External Interrupt pins:
UNO , NANO	2,3
Mega	2,3,18,19,20,21

Les Arduinos peuvent avoir plus de broches d'interruption activées en utilisant les interruptions de changement de broche. Dans les cartes Arduino basées sur ATmega168/328, n'importe quelle broche ou les 20 broches de signal peuvent être utilisées comme broches d'interruption. Elles peuvent également être déclenchées par des fronts montants ou descendants.

Utilisation des interruptions dans Arduino

Afin d'utiliser les interruptions dans Arduino, les concepts suivants doivent être compris.

Routine de service d'interruption (ISR)

Un Interrupt Service Routine ou un Interrupt handler est un événement qui contient un petit ensemble d'instructions. Lorsqu'une interruption externe se produit, le processeur exécute d'abord le code présent dans l'ISR et retourne à l'état où il a laissé l'exécution normale.

ISR a la syntaxe suivante dans Arduino :

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

digitalPinToInterrupt(pin) : Dans Arduino Uno, NANO les broches utilisées pour l'interruption sont 2,3 & dans mega 2,3,18,19,20,21. Spécifiez ici la broche d'entrée qui est utilisée pour l'interruption externe.

ISR : C'est une fonction qui est appelée quand une interruption externe est faite.

Mode : Type de transition à déclencher, par exemple, descente, montée, etc.

RISING : Pour déclencher une interruption lorsque la broche passe de BASSE à HAUTE.

FALLING : Pour déclencher une interruption lorsque la broche passe de HIGH à LOW.

CHANGE : Pour déclencher une interruption lorsque la broche passe de BASSE à HAUTE ou de HAUTE à BASSE (c'est-à-dire lorsque l'état de la broche change).

Certaines conditions lors de l'utilisation de l'interruption

La fonction Routine de service d'interruption (ISR) doit être aussi courte que possible.

La fonction Delay () ne fonctionne pas dans l'ISR et doit être évitée.

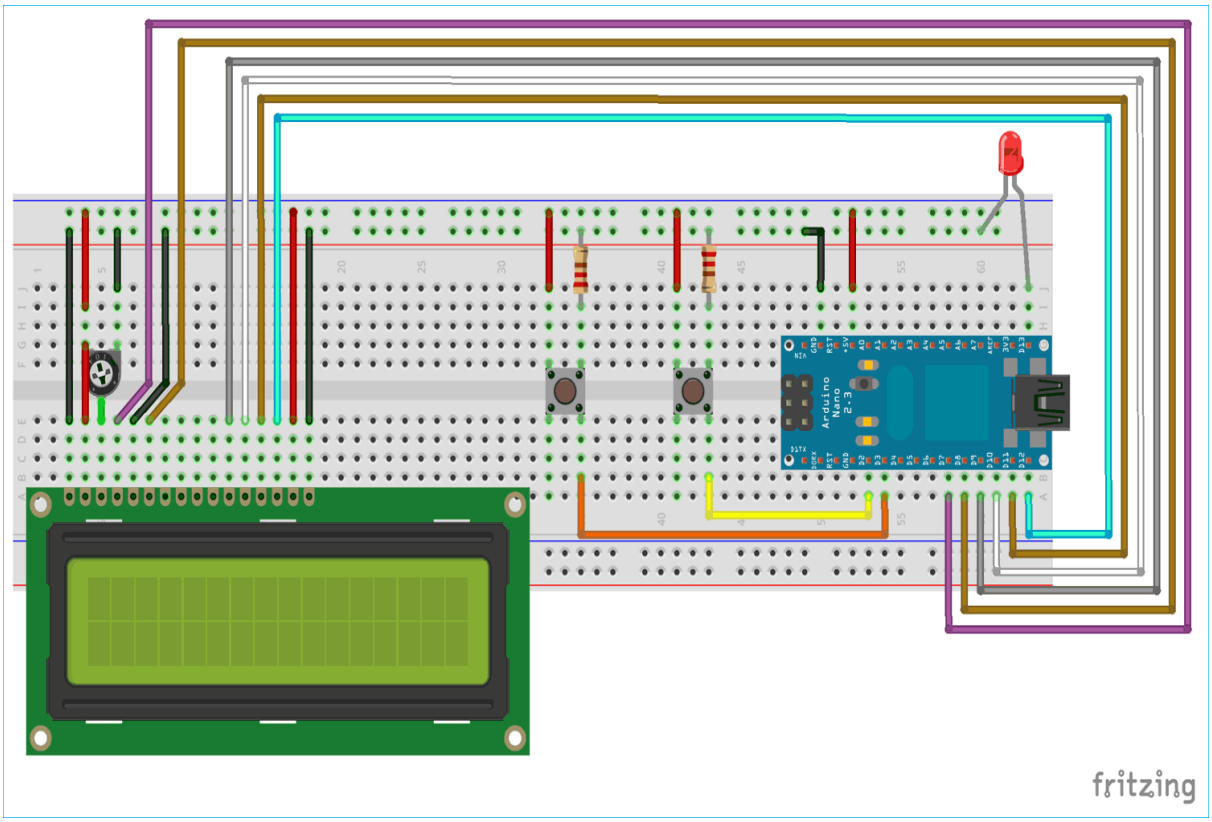
Dans ce tutoriel Arduino Interrupt, un nombre est incrémenté à partir de 0 et deux boutons poussoirs sont utilisés pour déclencher l'interruption, chacun est connecté à D2 & D3. Une LED est utilisée pour indiquer l'interruption. Si un bouton est pressé, la LED s'allume et l'écran affiche

Interrupt2 et s'éteint, et si un autre bouton est pressé, la LED s'éteint et l'écran affiche Interrupt1 et s'éteint.

Composants requis : Carte Arduino (Dans ce tutoriel, Arduino NANO est utilisé)

Bouton-poussoir – 2, LED – 1, Résistance (10K) – 2, LCD (16x2) – 1,

Câbles de connexion



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

Deux boutons poussoirs sont connectés à Arduino Nano aux broches D2 et D3. Ils sont utilisés pour utiliser deux interruptions externes, une pour allumer la LED et une autre pour l'éteindre. Chaque bouton poussoir a une résistance de 10k connectée à la masse. Ainsi, lorsque le bouton est pressé, il est en logique HAUT (1) et lorsqu'il n'est pas pressé, il est en logique BAS (0). Une résistance d'excursion basse est obligatoire, sinon il y aura des valeurs flottantes sur les broches d'entrée D2 et D3.

Une LED est également utilisée pour indiquer qu'une interruption a été déclenchée ou qu'un bouton a été pressé.

Programmation d'une interruption Arduino

Dans ce tutoriel, un nombre est incrémenté à partir de 0 et s'affiche en continu sur un écran LCD (16x2) connecté à l'Arduino Nano. Chaque fois que le bouton gauche (broche d'interruption D3) est pressé, la LED s'allume et l'écran affiche l'interruption 2, et lorsque le bouton droit (broche d'interruption D2) est pressé, la LED s'éteint et l'écran affiche l'interruption 1.

Le code complet avec une vidéo de travail est donné à la fin de ce tutoriel.

1. Tout d'abord, le fichier d'en-tête pour l'écran LCD est inclus, puis les broches de l'écran LCD qui sont utilisées pour la connexion avec l'Arduino Nano sont définies.

```
2. #include<LiquidCrystal.h>
```

```
3. LiquidCrystal lcd (7,8,9,10,11,12); // Define LCD display pins RS, E, D4, D5
   , D6, D7
```

4. 2. **Inside the void setup ()** function, first display some intro message on LCD display. Learn more about interfacing LCD with Arduino here.

```
5.   lcd.begin(16,2);

6.   lcd.setCursor(0,0);

7.   lcd.print("CIRCUIT DIGEST");

8.   lcd.setCursor(0,1);

9.   lcd.print("ArduinoInterrupt");

10.  delay(3000);

11.  lcd.clear();
```

3. Ensuite, dans la même fonction void setup (), les broches d'entrée et de sortie doivent être spécifiées. La broche D13 est connectée à l'anode de la LED, elle doit donc être définie comme une sortie.
pinMode(13,OUTPUT) ;
4. Maintenant, la partie la plus importante de la programmation est la fonction attachInterrupt(), elle est également incluse dans le void setup().
attachInterrupt(digitalPinToInterrupt(2),buttonPressed1,RISING) ;
attachInterrupt(digitalPinToInterrupt(3),buttonPressed2,RISING) ;
Ici, il est spécifié que la broche 2 est destinée à l'interruption externe, et la fonction buttonPressed1 est appelée lorsqu'il y a un RISING (LOW à HIGH) sur la broche D2. Et la broche 3 est aussi pour l'interruption externe et la fonction buttonPressed2 est appelée quand il y a un RISING sur la broche D3.
5. A l'intérieur de la boucle void(), un nombre (i) est incrémenté à partir de zéro et imprimé sur LCD(16x2).

```
lcd.clear();

 lcd.print("COUNTER:");

 lcd.print(i);

 ++i;

 delay(1000);
```

Dans le même void loop(), digitalWrite() est utilisé sur la broche D13 où l'anode de la LED est connectée. Selon la valeur de la variable output, la LED s'allumera ou s'éteindra.

digitalWrite(13,output) ;

6. La partie la plus importante est la création d'une fonction de gestion des interruptions selon le nom utilisé dans la fonction attachInterrupt().

Comme deux broches d'interruption sont utilisées (2 et 3), deux ISR sont nécessaires. Ici, dans cette programmation, les ISR suivants sont utilisés

buttonPressed1() :

void buttonPressed1()

{

  output = LOW ;

  lcd.setCursor(0,1) ;

  lcd.print("Interruption 1") ;

}

Cette fonction s'exécute lorsque le bouton poussoir sur la broche D2 est pressé (RISING EDGE). Cette fonction change l'état de la sortie en BAS, ce qui fait que la LED s'éteint et imprime "interrupt1" sur l'écran LCD.

boutonPressé2() :

void buttonPressed2()

{

  output = HIGH ;

  lcd.setCursor(0,1) ;

  lcd.print("Interrupt2") ;

}

Example Code

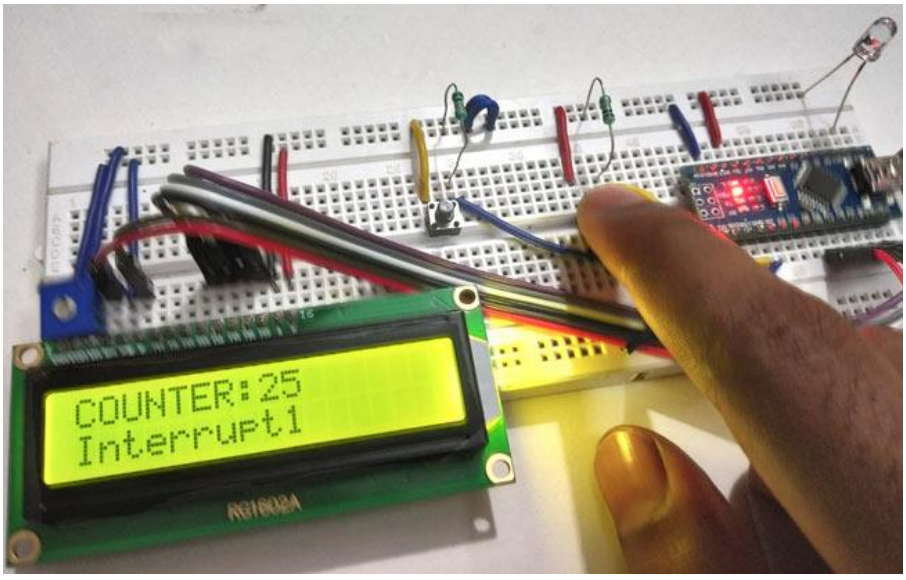
```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}
```

```
void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

BOARD	INT.0	INT.1	INT.2	INT.3	INT.4	INT.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
32u4 based (e.g Leonardo, Micro)	3	2	0	1	7	



Code

```
//Interrupts using Arduino
//Circuit Digest

#include<LiquidCrystal.h>                   // Including lcd display library
LiquidCrystal lcd (7,8,9,10,11,12);           // Define LCD display pins RS,E,D4,D5,D6,D7

volatile int output = LOW;
int i = 0;

void setup()

{
  lcd.begin(16,2);                          //  setting LCD as 16x2 type
  lcd.setCursor(0,0);
  lcd.print("CIRCUIT DIGEST");
```

```
  lcd.setCursor(0,1);
  lcd.print("ArduinoInterrupt");
  delay(3000);
  lcd.clear();
  pinMode(13,OUTPUT);

  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()          //ISR function excutes when push button at pinD2 is pressed
{
   output = LOW;               //Change Output value to LOW
   lcd.setCursor(0,1);
   lcd.print("Interrupt 1");
}

void buttonPressed2()          //ISR function excutes when push button at pinD3 is
pressed
{
   output = HIGH;              //Change Output value to HIGH
   lcd.setCursor(0,1);
   lcd.print("Interrupt2");
}
```