# RTOS Task 1 : Interrupt Handling

**Name : Omkar Sulakhe**

**USN: 2GI17EC065**

**Div: B**

Interrupt is a signal emitted by hardware or software when a process needs immediate attention. It alerts the processor to a high priority process requiring interruption of the current working process.

**Hardware interrupt**

There are many internal components in a microcontroller like timers, counters etc. that require attention of the processor. Since all the devices can’t obtain the attention of the processor at all times, the concept of “Interrupts” comes in to picture. **An Interrupt**, as the name suggests, interrupts the microcontroller from whatever it is doing and draws its attention to perform a special task.

In the event of an interrupt, the source of the interrupt (like a Timer, Counter etc.) sends a special request to the processor called **Interrupt Request (IRQ)** in order to run a special piece of code. The special code or function is called as **Interrupt Service Routine (ISR).**

There is a special function called attachInterrupt() in Arduino, using which you configure the External Interrupts. The syntax of attachInterrupt() function is as follows.

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

* digitalPinToInterrupt(pin)” is to specify the Interrupt pin number.
* ISR is the Interrupt Service Routine function call.
* mode is to indicate when to trigger the interrupt.
  + CHANGE: Trigger the Interrupt when there is a change in the pin value.
  + RISING: Trigger the Interrupt when the pin value rises from LOW to HIGH.
  + FALLING: Trigger the Interrupt when the pin value falls from HIGH to LOW.
  + LOW: Trigger the Interrupt when the pin value becomes LOW.

**Problem statement :**

A traffic system which follows certain sequence of led colours to maintain the traffic order.consider a situation where there is accident in the circle and immediately we are suppose to stop all the vehicles from passing by. In such situation one can interrupt the regular sequence for signaling and hold the vehicles for some time by just pressing the intrupt button.

The most important point is that introducing the concept of Interrupts into this situation, the controller can do whatever it wants (I mean anything other than checking for the status of the button, as per your code) and whenever the button is pressed, it will automatically stop the current execution and puts its full attention to the button press event.

**Code**

const int redledPin = 13; //output

const int yellowledPin = 14; //output

const int greenledPin = 15; //output

const int buttonPin = 2; //interrupt pin

int ledToggle = LOW; //initial state

void setup()

{

pinMode(ledPin, OUTPUT);

pinMode(buttonPin, INPUT\_PULLUP);

attachInterrupt(digitalPinToInterrupt(buttonPin), button\_ISR, CHANGE); //interrupt decleration

}

void loop() // normal sequence

{

digitalWrite(redledPin, HIGH);

delay(300);

digitalWrite(redledPin, LOW);

digitalWrite(yellowledPin, HIGH);

delay(100);

digitalWrite(yellowledPin, LOW);

digitalWrite(greenledPin, HIGH);

delay(200);

}

void button\_ISR() // note there should not be delays used in ISR

{

digitalWrite(redledPin, HIGH);

}

In the above code, the button is connected to Pin 2 (INT0) of Arduino and an interrupt is attached with respect to this pin on an event of CHANGE in the value of the button pin and triggering an ISR named button\_ISR.