Digital inputs

Lets suppose that we want to detect when the value of an external signal to microcontroller is low. But, in addition, the microcontroller needs to do some other tasks which last X seconds.

```
#define SECONDS X
delay(SECONDS*1000);
```

Digital read

The most common approach to this problem is to poll the signal. This is the same as read the signal value and ask for it.

```
if (digitalRead(INPUT_PIN) == LOW)
    Serial.println("Digital signal detected");
```

This approach works well, but in many cases it isn't the most efficient solution.

```
void loop()
{
  // put your main code here, to run repeatedly:
  delay(SECONDS*1000);
  if (digitalRead(INPUT_PIN) == LOW)
    Serial.println("Digital signal detected");
}
```

The problem is that while the microcontroller is doing other tasks (delay at above code), it wont be able to read the signal state. As a consequence, is pretty probable that some state changes couldn't be detected. This is because the state change detection is being realized by software and not by hardware.

External interrupts

Interrupts are more efficient than poll system, this is because the event **is detected by hardware and not by software**. This means that, even the microcontroller could be doing other tasks, it will be able to detect the event. Interrupts **link a function to an event**. In order to define an external interrupt, is needed:

- An Arduino pin which will detect the event
- A trigger condition
- An interrupt service routine

Pins which can be used by external interrupts depend on Arduino board as shown below:

Board	Digital pins used for interruption

Board	Digital pins used for interruption
Uno, Nano (Based on AtMega 328)	2, 3
Mega, Mega ADK, Mega 2560	2, 3, 18, 19, 20, 21
Micro, Leonardo (Based on 32u4)	0, 1, 2, 3, 7
Zero	All pins, except 4
Due	All pins

The most commons trigger conditions are:

- CHANGE: Interrupt is triggered when occurs a falling edge or a rising edge
- FALLING: Interrupt is triggered when occurs a falling edge, this happens when signal goes from HIGH to LOW
- RISING: Interrupt is triggered when occurs a rising edge, this happens when signal goes from LOW to HIGH

Now we are ready to solve the initial problem in the most efficient way. Microcontroller will run *signalDetected()* interrupt service routine when it detect a falling edge on pin *INTERRUPT_PIN*

```
attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), signalDetected, FALLING);
```

A 'flag' is activated inside the *signalDetected()* function. This flag indicates that the wished event happened.

```
void signalDetected()
{
  isDetected = true;
}
```

In the main program, after doing all other tasks (delay at below code), the flag value is readed.

```
void loop()
{
  // put your main code here, to run repeatedly:
  delay(1000*SECONDS);
  if (isDetected)
  {
    Serial.println("Digital signal detected");
    isDetected = false;
  }
}
```

This solution is more efficient than the previous one based on 'poll system' because all events will be detected. Every time that a falling edge occurs on *INTERRUPT_PIN*, the microcontroller will run the interrupt

service routine, after this the main program will read the 'flag' value in order to know if an event occurs.

Things to consider with the use of interrupts

- Inside an interrupt service routine, most of timing functions wont work as expected (millis, delay, etc). This is because this functions use interrupt service routines which can't be executed while an external interrupt is being executed.
- Any variable which will be modified inside an interrupt service routine, should be declare as 'volatile'
- Data received through serial port may be lost while an external interrupt is being executed
- Interrupt service routines should be as shorts as possible.