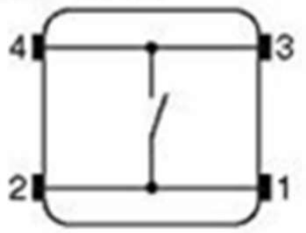
Clase 8 [R]— Pulsadores 2da parte Interrupciones : ¿Qué son y para que se usan? - Recuerdo

Los pines de un uContolador se llaman GPIO = General Purpose Input/Output, Entrada/Salida de Propósito General. Vamos a ver un 1er uso como entradas:

Pulsadores || push button switch || momentary button

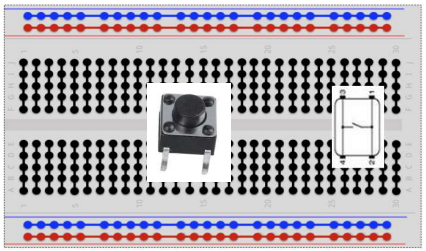
Conecta 2 cables/pines solo mientras esta pulsado

Sirve para dar ordenes al programa del uControlador



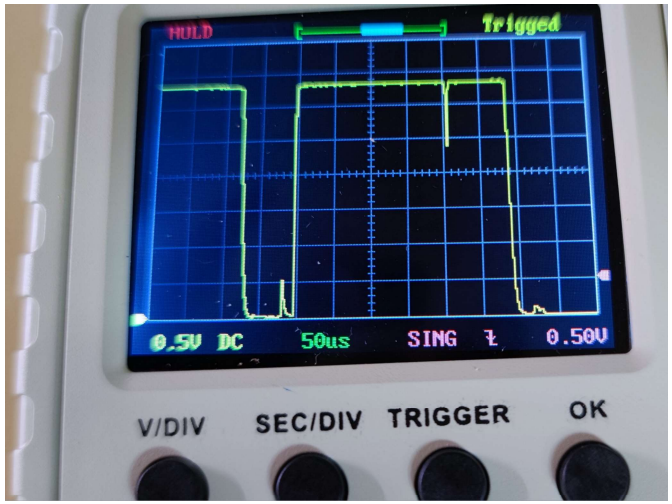
Con Protoboard usaremos normalmente el **pulsador de 4 pines**, ver esquema de conexionado

Posicionado : los lados que NO tienen pines en la dirección Arriba-Abajo, las patas a cada lado de la canaleta



Si el pulsador es **de 2 pines**, cada pata a un numero distinto del Protoboard

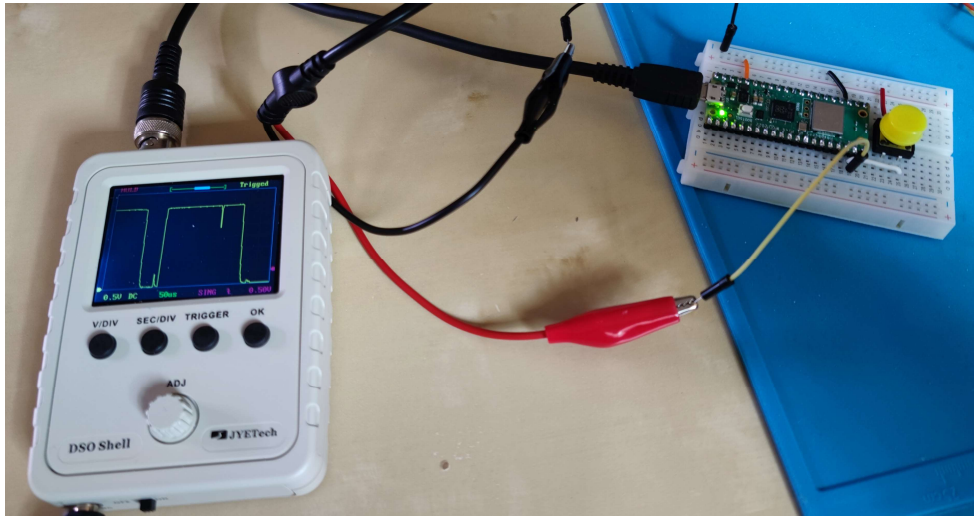
Clase 8 [R]— Pulsadores problema de rebotes (‘Debouncing’=quitar rebotes)



Los pulsadores no producen un cierre inmediato del circuito, ni tampoco una apertura “limpia”.

En la foto puedes ver un ejemplo real de apertura del pulsador NO limpia

(el caso de pulsar el pulsador ‘cierre’ se observo este caso es bastante “limpio”)



¿Cómo “filtrar” los rebotes?:

1. **Añadiendo condensador**
   -> 1 uF va ok entre los pines del pulsador
   Si el condensador es electrolítico cuidado la polaridad
2. **Comprobando estado varias veces**
3. **Añadiendo retardos**
4. **Combinando 2 & 3**

Clase 8 [R]— Pulsadores 2da Interrupciones ¿que son?

Una interrupción **es una suspensión temporal de la ejecución de un proceso**, para pasar a ejecutar una **subrutina de servicio de interrupción**

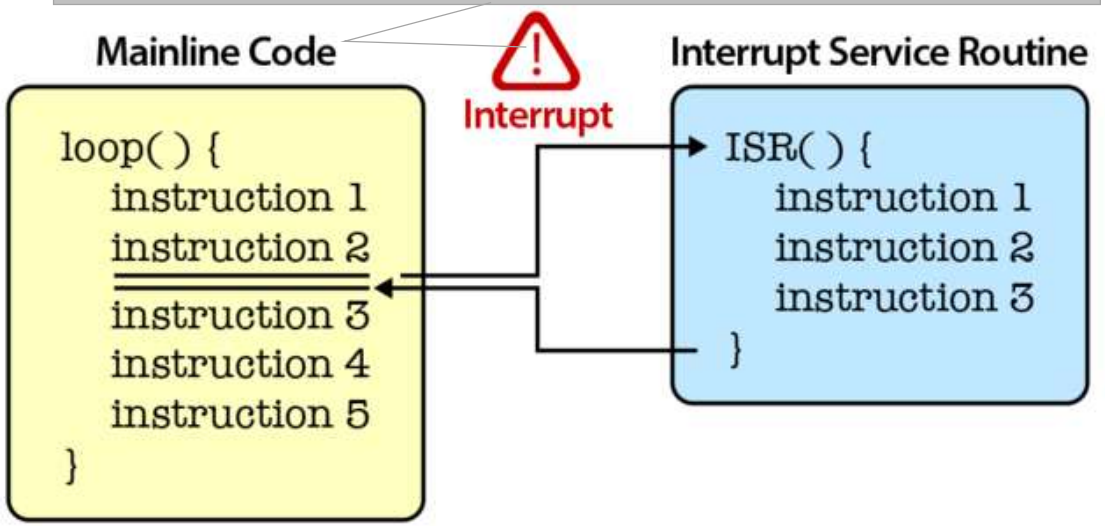
USO : liberar al programa **principal** de estar preguntando por **eventos externos o periódicos y al mismo tiempo responder con rapidez**

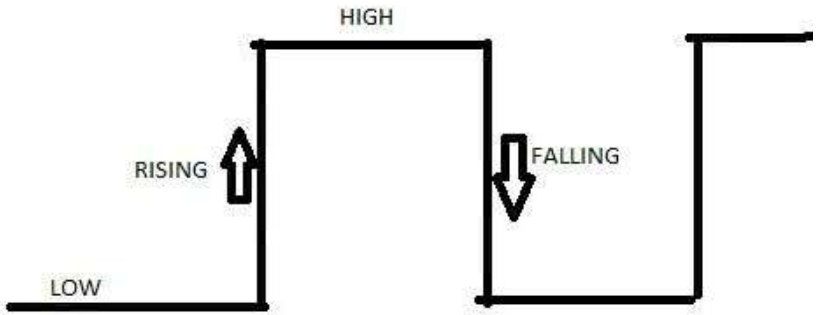
Tipos de Interrupciones:

- **Interrupciones HW o externas**:
  La mas frecuentes son las de
  - **Estado / Cambio de Estado de un pin**

  Son las que vamos a usar con el pulsador
- **Interrupciones SW**
- Un **evento programado o Timer**
- **Excepciones y Errores**

Implica guardar toda la información de lo que estaba ejecutando: registros, lugar del programa principal, etc., para recuperarla cuando regrese de la interrupción





Clase 8 [R]— Manejo con interrupciones

Ejemplo de pag 69 Reaction A single-player game

1- Libro oficial "Get Started with MicroPython on Raspberry Pi Pico - January 2021"

Ejemplo para GUARDAR

```
from machine import Pin
from urandom import uniform
import utime
EXTERNAL_BUTTON = 14
pulsador = Pin(EXTERNAL_BUTTON, Pin.IN, Pin.PULL_DOWN)
intled = Pin("LED", Pin.OUT)
pressed = False

def manejaPulsador(pin):
    global pressed
    if not pressed:
        pressed=True
        tiempoReaccion = utime.ticks_diff(utime.ticks_ms(), tiempo_start)
        print("¡Tu tiempo de reaccion fue de  " + str(tiempoReaccion) + " millisegundos!")
```

BMMR_CL9_switch_react_irq_1_0.py
Con algunos cambios menores

En amarillo sentencias que necesitan explicación

Clase 9 [R]— Manejo con interrupciones
Ejemplo de pag xx Reaction game 1 player

1- Libro oficial "Get Started with MicroPython on Raspberry Pi Pico - January 2021"

Continuación…..

```
# presentacion e instrucciones
print("El led se encendera y apagara. Tienes que pulsar despues de que se apague")


intled.value(1)
utime.sleep(uniform(5, 10))
intled.value(0)
tiempo_start = utime.ticks_ms()
pulsador.irq(trigger=Pin.IRQ_RISING, handler=manejaPulsador)


utime.sleep(2)
print('Fin del juego')
```

En amarillo sentencias que necesitan explicación

Pero esta no es la forma habitual de usar interrupciones en robotica. Lo normal es tener **un programa ejecutándose en bucle y que las interrupciones se usen para "encuestar" eventos externos** como el cambio de un pin, sin tener que estar "usando" el programa principal y a la vez responder rápido

Clase 8 [R]— Manejo con interrupciones
Ejemplo “encuesta” de un pulsador con anti-rebotes (debouncing)

Ejemplo para GUARDAR

Micropython for Kids - 3 - button

```
from machine import Pin
from urandom import uniform
import utime
EXTERNAL_BUTTON = 14
pulsador = Pin(EXTERNAL_BUTTON, Pin.IN, Pin.PULL_DOWN)
veces_pulsadas = 0 # guarda las veces que se presiono el pulsador
last_time = 0 # guarda la ultima marca de tiempo en que se presiono el pulsador

def manejaPulsador(pin):
    global veces_pulsadas, last_time
    new_time = utime.ticks_ms()
    # Si ha pasado mas de 200ms desde el ultimo evento, temenos un nuevo evento. Evita los REBOTES
    if utime.ticks_diff(new_time, last_time) > 200: # ¿Porque no hacemos una resta simplemente ?
        veces_pulsadas += 1
        last_time = new_time
```

BMMR_CL9_switch_react_irq_3_0.py
Con algunos cambios menores respecto a Micropython for kids

En amarillo sentencias que necesitan explicación

Clase 8 [R]— Manejo con interrupciones

Ejemplo para GUARDAR

Ejemplo "encuesta" de un pulsador con anti-rebotes (debouncing)

Micropython for Kids - 3 – button – continuación .....

```
pulsador.irq(trigger=Pin.IRQ_RISING, handler=manejaPulsador)
intled = Pin("LED", Pin.OUT)
intled.on()
veces_pulsadas_viejo = 0
while True: # Bucle principal del prgrma en el microcontrolador
    print('hago cosas y hago parpadear el led')
    utime.sleep(uniform(1, 4))
    if veces_pulsadas_viejo != veces_pulsadas:
       print('Veces pulsadas = ', veces_pulsadas)
       veces_pulsadas_viejo = veces_pulsadas

    intled.toggle() # cambia el led de encendido a apagado y viceversa
```

En amarillo sentencias que necesitan explicación

Hay un ejemplo de "encuesta" de pulsador SIN programación anti-rebotes

Prueba para ver que cuenta las pulsación mas veces de las debidas. También se puede poner un **condensador de1uF (cerámico 105)** entre los pines del pulsador para filtrar

BMMR_CL9_switch_react_irq_2_0.py

Version SIN anti-rebotes