DIE
RIALES

6 - Interrupciones

- *Conceptos generales*
- *Interrupciones externas*
- *Interrupciones temporales*
- *Ejemplos*

POLITÉCNICA

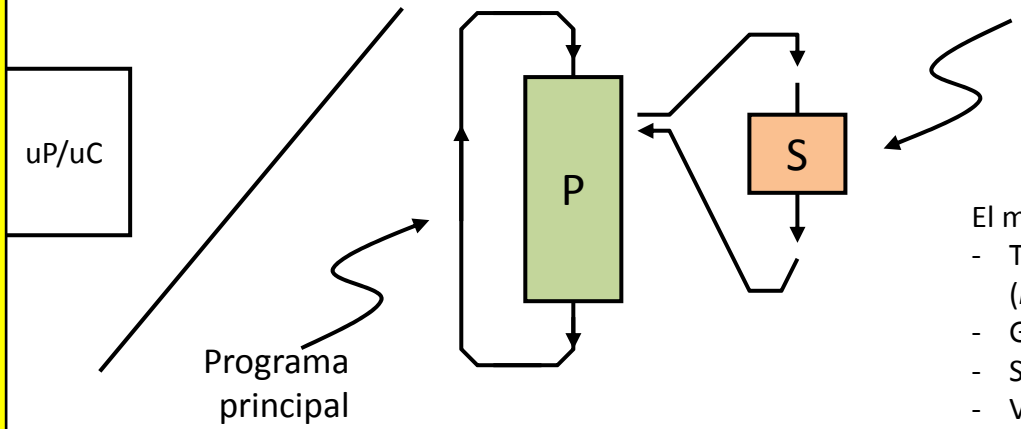
pciones: Conceptos generales (I)

una interrupción?

desvío de la ejecución normal de un programa a petición de un periférico

igen de la petición es un evento externo al uP, y por tanto, asíncrono respecto al rama (no podemos predecir en qué parte del programa, ni cuándo, ocurrirá)

micro la acepta, ejecutará una función asociada para atender al periférico



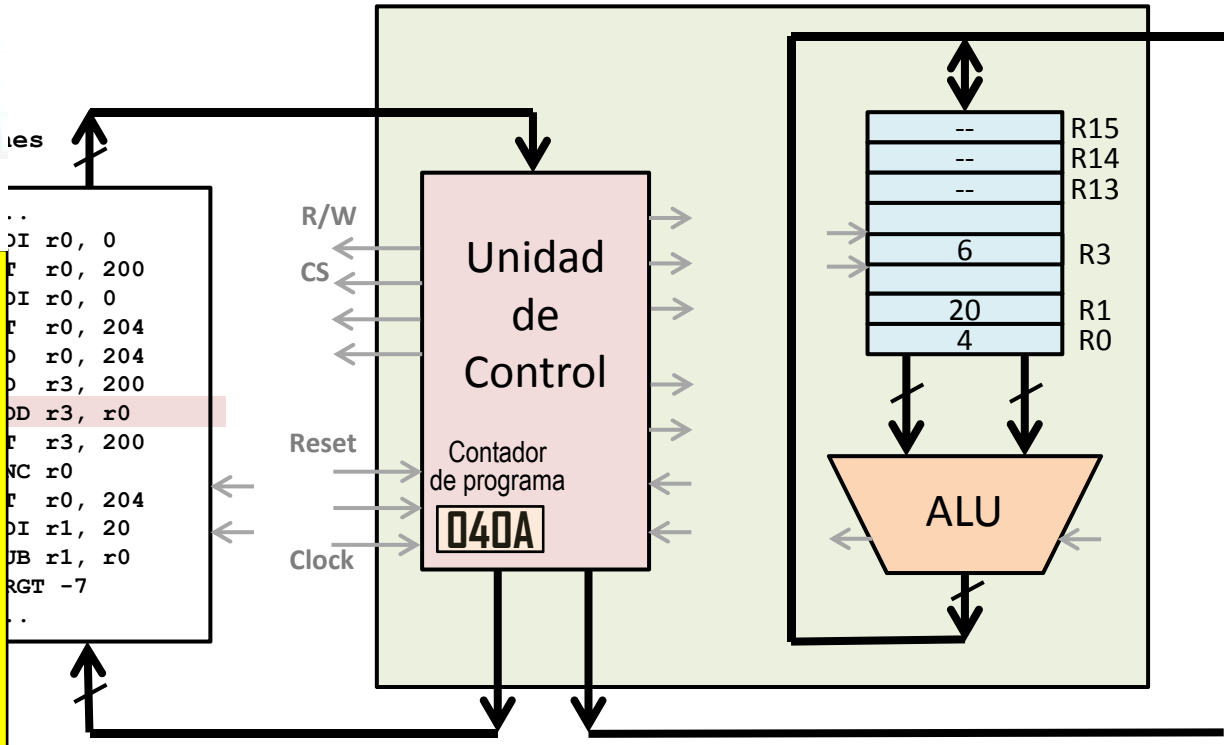
El micro debe (lo hace automáticamente):
- Terminar la instrucción en curso (*no en C, sino en código máquina*)
- Guardar el estado del micro
- Saltar a la subrutina y ejecutarla
- Volver (*dejándolo todo igual, en principio*)

rtancia de las interrupciones en el control

ntrol se simplifica, ya que se atiende a los periféricos cuando éstos lo ieren, y no es necesario hacer *polling* o bloqueo

casiones, un programa de control está formado por un bucle de espera, en el no se hace nada, y sólo se atienden las interrupciones que van llegando

der una interrupción



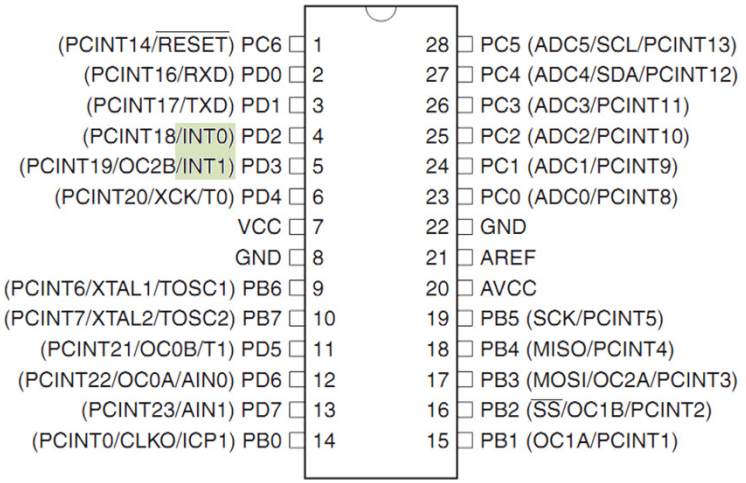
Qué hace el micro cuando tiene que atender una interrupción:

1 – Guarda el valor de Contador de programa
2 – Guarda los registros y el "estado" del micro
3 – Copia en el Contador de programa la dirección donde está la subrutina de interrupción
4 – Ejecuta el código en la nueva posición hasta que aparezca un 'return'
5 – Restauran el valor del Contador de programa, el estado del micro y los registros

pciones externas

os los uC disponen de señales externas que pueden producir interrupciones
rminadas condiciones (flanco de subida o bajada, nivel…)

so del Atmega168, todos los
E/S permiten producir
ciones externas, pero sólo
puerto D son configurables
nco de subida, bajada,
). *Sólo trataremos estos*.



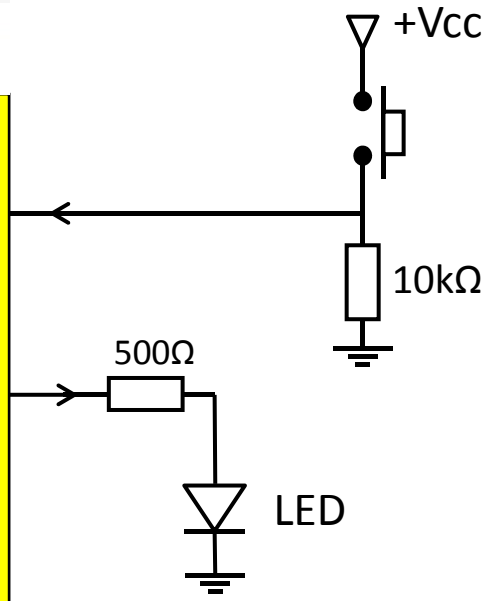
API de Arduino para trabajar con las interrupciones externas:
2 y 3 de la placa de Arduino están conectados a PD2 y PD3, y por lo tanto tienen la posibilidad de interrupciones

terrupt(int_num, isr_name, condition); // Asocia una rutina a una interrupción bajo una determinada condición
m : 0 para el pin 2, 1 para el pin 3
me: nombre de la rutina de servicio (función que no debe tener parámetros)
ion: condición que produce la interrupción (FALLING, RISING, BOTH, LOW)

nterrupt(int_num); // Desactiva la interrupción

o: Controlar un LED con interrupciones

ndo la API de Arduino ¿Qué habría que hacer para conmutar cada vez que se pulsa el botón?




```
#define LED 13

void setup() {
  pinMode( LED , OUTPUT );
  // Asocio la función toggleLed a la interrupción por PD2
  attachInterrupt( 0, toggleLed, RISING );
}

// Función que se llamará en la interrupción
// No debe tener parámetros
void toggleLed() {
  int ledVal = digitalRead( LED );
  digitalWrite( LED, !ledVal );
}

void loop() {
  // Aquí no hay por qué hacer nada
}
```

⁇ Al comprobar el funcionamiento del sistema se observa que al pulsar o soltar, el led cambia de estado de forma no prevista, a veces cambia cuando no debe y otras no cambia cuando debe (debería cambiar de estado sólo cada vez que pulso)
→¿A qué puede ser debido ese comportamiento?
→¿Cómo podría solucionarse?

pciones temporales y periódicas

isponen de contadores programables que permiten producir
ciones de forma periódica o pasado un cierto tiempo (en el caso del
168, dispone de 3 contadores que permiten generar interrupciones
as y temporales de diversas formas con periodos desde pocos µseg hasta
.

API de para trabajar con las interrupciones periódicas y temporales (ATENCIÓN: estas funciones son un añadido a no hecho para esta asignatura. Para trabajar con ellas es necesario instalar un parche a la API de Arduino. Seguir es en la descripción del trabajo voluntario).

odicInterrupt(isr_name, period_in_ms); // Asocia una rutina a una interrupción periódica
: nombre de la rutina de servicio (función que no debe tener parámetros)
_ms: periodo en milisegundos. Debe ser un entero

odicInterrupt(isr_name); // Elimina la asociación

edInterrupt (isr_name, time_in_ms); // Asocia una rutina para que se ejecute
// una sola vez pasado el tiempo especificado

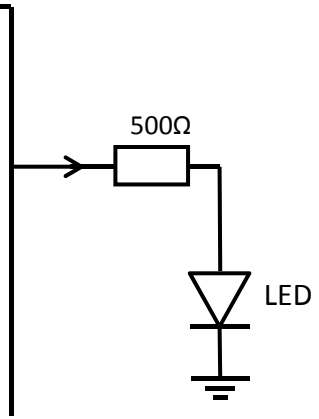
edInterrupt(isr_name); // Elimina la asociación

s asociadas con las interrupciones:

); // Habilita las interrupciones a nivel global
ts(); // Deshabilita las interrupciones a nivel global

o: Parpadear un LED con interrupciones

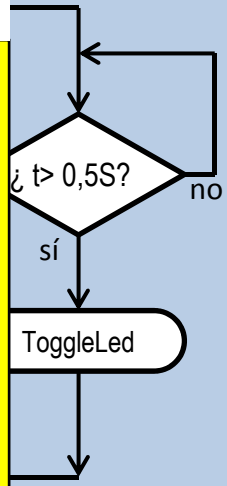
lo la API de Arduino ¿Qué habría que hacer para conmutar el LED cada 0,5




```
#define LED 13

void setup() {

  pinMode( LED , OUTPUT );

  // Asocio la función toggleLed a la interrupción periódica
  // que ocurre cada cada 500 milisegundos

  attachPeriodicInterrupt( toggleLed, 500 );
}

// Función que se llamará en la interrupción
// No debe tener parámetros
void toggleLed() {
  int ledVal = digitalRead( LED );
  digitalWrite( LED, !ledVal );
}

void loop() {
  // Aquí se podrían hacer otras cosas, sin que
  // nos interfiera lo que se hace en la interrupción
  ...
  ...
}
```

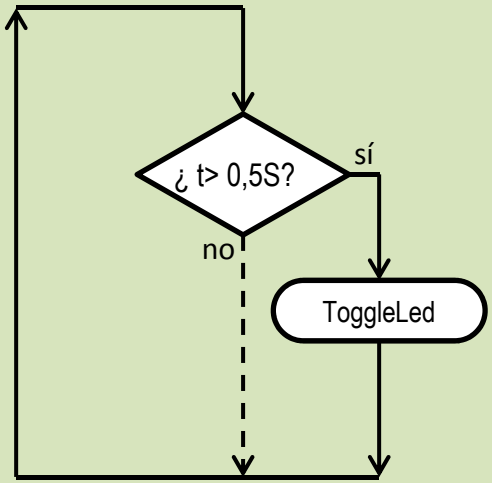
o: Comparando soluciones flujogramas

el proceso:



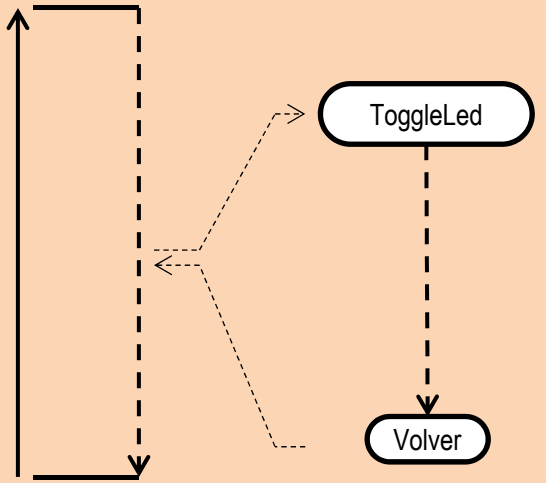
cesador está
esperando que
npo.
r cabida a otras

❑ **Consulta periódica**:



- ❑ El microprocesador comprueba cuánto tiempo ha pasado cada vez que pasa por ese punto del código.
- ❑ El resto del tiempo se puede aprovechar para realizar otras tareas.

❑ **Interrupciones**:



- ❑ Sólo cuando haya transcurrido el tiempo programado, el microprocesador atenderá la interrupción y cambiará el led.

o: Comparando soluciones

a API de Arduino ¿Cómo resultaría el control del LED mediante los distintos
entos de entrada salida que conocemos (bloqueo, consulta periódica e interrupciones)?

```
----------
OCCESO
----------


OUTPUT );


{
gitalRead( LED );
ED, !ledVal );


elay espera
ma bloqueante


e hay aquí
utará cada
```

```
// Solución por
//--------------------------
// CONSULTA PERIÓDICA
//--------------------------

#define LED 13

void setup() {
  pinMode( LED , OUTPUT );
}

void toggleLed() {
  int ledVal = digitalRead( LED );
  digitalWrite( LED, !ledVal );
}

unsigned long nextTime = 500;

void loop() {

  // la función millis() devuelve
  // los ms pasados desde el reset
  if (millis() > nextTime) {
    nextTime = millis() + 500;
    toggleLed();
  }

  // Aquí se podrían hacer
  // otras cosas, siempre que no
  // tarden demasiado
  ...
}
```

```
// Solución mediante
//--------------------------
// INTERRUPCIONES
//--------------------------

#define LED 13

void setup() {
  pinMode( LED , OUTPUT );
  attachPeriodicInterrupt(
    toggleLed, 500 );
}

void toggleLed() {
  int ledVal = digitalRead( LED );
  digitalWrite( LED, !ledVal );
}

void loop() {

  // Aquí se podrían
  // hacer otras cosas,
  // sin que nos interfiera lo que
  // se hace en la interrupción
  // ni viceversa
  ...
}
```