MÓDULO 4: PROJECTO 11.05 " TimerOne"

CURSO PROGRAMACIÓN DE PLACAS ROBÓTICAS

Objetivos

- Presentar las interrupciones programadas o Timers.
- Instalar una librería, la TimerOne.
- Mostrar un ejemplo de programación de un evento con un Timer.

Introducción

- Hasta ahora, hemos medido el tiempo en milisegundos y usado el delay para temporizar las cosas, así usábamos un delay de unos 250 ms para que el blinking LED se encendiese y se apagase, tranquilamente.

- El problema del delay () es que congela Arduino el tiempo especificado. Mientras no sea mucho, bueno, podemos aceptarlo, pero imagínate que queremos hacer el blinking LED para un semáforo, donde los tiempos de espera van de digamos 30 segundos a 1 minuto.

  Podemos pedirle que haga un delay de 60 segundos * 1.000 = 60.000 millis, pero claro esto supone que no podremos hacer ninguna otra cosa hasta que no pase un minuto, ni ver si nos pulsan un botón, o refrescar una pantalla con el tiempo que queda de semáforo.

Introducción

- Así que no parece muy buena idea usar delays en muchas situaciones. No es raro que queramos programar tareas periódicas en nuestros Arduinos en rangos que van desde unos microsegundos hasta varios minutos, pero queremos hacerlo de un modo que entre tanto podamos seguir trabajando.

- Para eso tenemos las **interrupciones programadas o Timers**, para que nos toquen el timbre cuando hay que ejecutar una función programada, sin necesidad de estar continuamente comprobando si es la hora.

- Arduino dispone además de una segunda clase de interrupciones, **los Timers**, que hacen lo mismo que las interrupciones hardware, pero en lugar de dispararse cuando se cumple un cierto proceso hardware en uno de los pines, se dispara cuando ha transcurrido un tiempo preciso, previamente programado.

Es el equivalente del despertador, que cada mañana suena a la misma hora.

Los contadores internos de los Timers

- Nuestro Arduino UNO tiene un cristal que bate a 16 MHz, o 16.000.00 de veces por segundo.
- Teóricamente podríamos fijar una interrupción cada 1/16000000 segundos, lo que no sería muy útil porque cada instrucción de Arduino necesita varios pulsos de reloj para ejecutarse (y algunos muchos pulsos).
- Por eso cada Timer dispone de un registro interno que indica cada cuantos ticks del reloj debe dispararse. Básicamente el que nos interesa es un Timer cuyo registro es un entero sin signo de 16 bits, lo que le permite contar hasta 2^{16} = 65.536

La librería TimerOne

- Hay varias versiones de esta librería corriendo por ahí, soportar más modelos de Arduino y porque el código es más rápido que el original.
- Las cosas importantes de la librería. Cuando la importéis tendréis esta línea:
  - #include <TimerOne.h>
  - Esto nos crea un objeto llamado Timer1 directamente, sin necesidad de instanciarlos
  - Lo siguiente es programar el intervalo de disparo en microsegundos:
    - Timer1.initialize(150000); // 150 ms
- Y ya solo falta hacer el attach de la interrupción con el servicio de gestión:
  - Timer1.attachInterrupt(ISR_Callback)//Mirar Ejemplo para mas detalle

La librería TimerOne

- Entre las ventajas tenemos:
  - Código limpio y elegante.
  - No importa que estemos en un delay, la interrupción salta impecable.
  - La medida del tiempo es muy precisa.
- Inconvenientes:
  - El primero y grave, es que si jugamos con los timers, muchas de las instrucciones que dependen de ellos dejaran de funcionar(Los pines PWM y analogWrite() y la librería Servo