

**Sistemas Embebidos**  
**Trabajo práctico N°3 - Parte B - Año 2022**  
**Interrupciones, Timers, EEPROM, modos de bajo consumo de  
energía**

**Objetivos**

- Utilizar librerías comunes para configurar y utilizar timers, modos de bajo consumo de energía e interrupciones.
- Leer y escribir memoria EEPROM.
- Interactuar aplicaciones en sistemas embebidos con aplicaciones web.

**Metodología**

Trabajo individual o grupal. 2 estudiantes por grupo máximo.

Tiempo de realización estimado: 2-3 clases.

**Aprobación**

- Mostrar en clases la aplicación funcionando correctamente.
- Enviar los programas de computación implementados a través de la plataforma Moodle.

**Materiales necesarios**

- Placas Arduino UNO (provistas por la cátedra).
- Entorno de desarrollo de Arduino UNO (puede descargarse de <https://www.arduino.cc/en/software>).
- Librerías para Arduino IDE:
  - TimerOne creada por Paul Stoffregen.
  - LowPower\_LowPowerLab creada por LowPowerLab.

**Actividades**

Escribir una aplicación que realice las siguientes tareas:

1. Implemente un reloj en tiempo real que cuente años, meses, días, horas, minutos y segundos usando un Timer del Arduino Uno. Por simplicidad, suponga que todos los meses poseen 30 días. Para el año, utilice el formato "22" (los últimos dos dígitos del año). Este reloj debe poder actualizarse desde el puerto serie mediante un comando con el formato: "New-date=aa/mm/dd-hh:mm:ss". Debe implementarse mediante un Timer que produzca una interrupción cada segundo, y en la respectiva rutina de servicio del Timer, incrementar un segundo el reloj.

2. Implementar interrupciones en los pines 2 y 3 del Arduino Uno, de modo que cuando se presionen los pulsadores conectados a dichos pines, se almacene en la memoria EEPROM la hora completa (aa/mm/dd-hh:mm:ss) a la cual ocurrió el evento, y el evento ocurrido. La función debe ser tal que, si el Arduino se desconecta de la fuente de alimentación y se vuelve a conectar, nuevas escrituras en la EEPROM no deben sobrescribir escrituras anteriores.
3. Implementar funciones que permitan recuperar la información almacenada en la EEPROM para mostrarse en la página web cuando se envíe el comando "leer-eprom" a través del puerto serial del Arduino UNO. La memoria EEPROM debe poder borrarse completamente cuando se envíe el comando "borrar-eprom" a través del puerto serie.
4. Hacer que el Arduino Uno permanezca en modo bajo consumo de energía (modo Idle) cuando no esté realizando ninguna de las tareas listadas antes. El Arduino debe ser despertado por:
  - a. El Timer, cuando interrumpa cada segundo para actualizar el reloj.
  - b. Las interrupciones de los pulsadores 2 y 3.
  - c. El puerto serie, cuando el Arduino reciba un comando a través del mismo. Notar que la recepción de datos desde el puerto serie deberá implementarse como interrupción, no en el loop principal. Analice el ejemplo "SerialEvent" de la carpeta "Communication" para comprender su funcionamiento.

Al utilizar el modo bajo consumo, es recomendable que todas las tareas que realice el Arduino se implementen como interrupciones que despierten al procesador. El loop principal debería tener solo la primitiva para dormir al microcontrolador y algunas rutinas que permitan verificar el buen funcionamiento.

5. Toda la interacción del usuario con la aplicación debe realizarse a través de una página web (puede reutilizar los mecanismos implementados en trabajos prácticos anteriores).

### **Sugerencias de implementación**

A continuación se brindan sugerencias de implementación. No es obligatorio seguirlas.

#### Implementación del reloj en tiempo real

Un reloj en tiempo real suele implementarse mediante un contador que cuenta la cantidad de segundos desde una fecha inicial, por ejemplo, el segundo cero puede ser el 1/1/2000 a las 00:00:00 horas. Además, se agregan funciones que realicen las siguientes tareas:

- Transforman los segundos totales en cadenas de texto que indiquen fechas y horas en formatos como "aa/mm/dd-hh/mm/ss".

- Transformen cadenas de texto con información de la fecha y hora en formatos como “aa/mm/dd-hh/mm/ss” a una variable entera que almacene el total de segundos desde el segundo cero.

La hora del reloj en tiempo real debe actualizarse a través de una interrupción que cada segundo sume un segundo a la hora. Existen muchas librerías que emulan timers, pero pocas que realmente trabajen sobre el hardware de un timer. Se sugiere utilizar la librería “TimerOne” creada por Paul Stoffregen, ya que la misma trabaja directamente sobre el hardware del timer<sup>1</sup>. Se sugiere instalar la librería y estudiar el ejemplo “Interrupt” para comprender su funcionamiento.

Tenga en cuenta que necesitará una variable de varios bytes para almacenar los segundos necesarios (por ejemplo: long int).

### Implementación de interrupciones en los pines 2 y 3

La siguiente primitiva configura las interrupciones de los pines 2 y 3.

*attachInterrupt(número de la fuente de interrupción, nombre de la rutina de servicio, evento que dispara la interrupción);*

- Número de la fuente de interrupción:
  - Pin 2: Interrupción número 0
  - Pin 3: Interrupción número 1
- Eventos que disparan las interrupciones:
  - LOW: para activar la interrupción cuando el pin es bajo
  - CHANGE: para activar la interrupción cuando el pin cambia de estado.
  - RISING: dispara la interrupción cuando se detecta un flanco de subida.
  - FALLING: dispara la interrupción cuando se detecta un flanco de bajada.
- Nombre de la rutina de servicio: Debe indicarse el nombre de la rutina de servicio, sin agregar (). La rutina de servicio debe ser escrita como una función que no recibe argumentos ni retorna ningún resultado.

Ejemplo: `attachInterrupt(digitalPinToInterrupt(2),pin2_ISR,FALLING);`

### Leer y escribir la memoria EEPROM

Cada posición de la memoria EEPROM almacena un byte. Se sugiere analizar los ejemplos `eeprom_clear`, `eeprom_write` y `eeprom_read` para comprender el funcionamiento de la memoria EEPROM. Estos ejemplos se encuentran en la carpeta de ejemplos EEPROM, instalada por defecto en Arduino IDE.

Considerar que la hora requiere una variable de varios bytes, mientras que cada posición de la memoria EEPROM almacena un byte.

Tener en cuenta que la escritura de la memoria flash para cargar un nuevo programa no sobrescribe la memoria EEPROM. Cuando una posición de la memoria EEPROM está vacía, tiene escrito el número binario 11111111 (0xFF).

### Modos de bajo consumo

Se sugiere instalar y analizar los ejemplos de la librería LowPower\_LowPowerLab creada por LowPowerLab.

La primitiva para poner el microcontrolador Atmel328P en modo bajo consumo es:

```
LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF ,  
SPI_OFF, USART0_OFF, TWI_OFF);
```

Donde el flag SLEEP\_8S indica que el microcontrolador debe despertarse luego de 8 segundos (vea la documentación de la librería para otros intervalos de tiempo).

Los demás flags indican que periféricos deben permanecer funcionando, y por lo tanto pueden interrumpir y despertar al microcontrolador. Tenga en cuenta que el Timer1 y el puerto USART deben permanecer funcionando.

En cuanto al puerto serie, si se utiliza la bandera USART0\_ON, permanece activo el receptor esperando datos. Por otro lado, el puerto serie trabaja de forma asíncrona con el procesador, por lo tanto, si intenta transmitir un dato desde el Arduino a través del puerto serie e inmediatamente coloca el microcontrolador en modo bajo consumo, la transmisión serial podría no completarse. Por lo tanto, antes de ingresar en modo bajo consumo, debe agregar una espera de resguardo luego de un envío de datos (se sugiere utilizar `delayMicroseconds(10000)`).