Facultad de Ingeniería - Universidad Nacional de Cuyo			
Asignatura:	**Arquitectura de Computadoras**		
Carrera:	**Licenciatura en Ciencias de la Computación**		
Año: 2022	**Semestre: 1**	**Trabajo Práctico N°4**	

OBJETIVOS

- Reconocer el funcionamiento del sistema de entrada/salida de una computadora.
- Comprender los mecanismos de entrada salida a través de interrupciones, mapeo en memoria e instrucciones especiales.
- Utilizar funciones adicionales que poseen los dispositivos de sistemas embebidos.
- Poner en práctica los conceptos teóricos aprendidos mediante prácticas sobre una computadora simple.
- Comprender la diferencia entre un lenguaje de alto nivel y ensamblador.

Metodología

Trabajo grupal. Número de estudiantes por grupo según computadoras disponibles.
Tiempo de realización estimado: 2 - 3 clases.

Aprobación

- Mostrar en clases los programas que se solicitan en las actividades 1, 2 y 3 funcionando.
- Realizar la actividad 4 y responder a través de la plataforma Moodle.
- Responder preguntas que los profesores realicen.

Materiales y equipamientos necesarios:

Computadoras (Notebook, PC de escritorio, etc).
Interfaz de desarrollo de plataformas Arduino (provista por la cátedra).
Placas Arduino Uno (provistas por la cátedra).
Placas de desarrollo, leds, pulsadores, elementos auxiliares (provistos por la cátedra).

Actividad 1 - Interrupciones

1) Utilizando la interfaz de desarrollo de las plataformas Arduino, escriba un programa que encienda y apague de forma intermitente (que parpadeen) los leds conectados en los pines 13, 12, 11, 10, 9, 8, 7 y 6. El tiempo en estado encendido y apagado deberá ser de un segundo.
Importante!: primero configure los pines respectivos como salidas.

Sugerencia 1: Cree primero un programa que encienda y apague un solo led, luego de verificar que funciona correctamente, extienda el código a todos los leds.
Sugerencia 2: Utilice la función *delay(tiempo)* para implementar una pausa de un segundo, donde el argumento *tiempo* es el tiempo en milisegundos que dura la pausa.

2) Escriba un bloque de código (puede ser como subrutina) que, al pulsar el pulsador conectado al pin 2, los led se enciendan consecutivamente, uno a uno, solo uno por vez, desde el 13 al 6, durante 0,8 segundos cada led.
Importante!: primero configure el pin 2 como entrada.

3) Escriba un bloque de código que, al pulsar el pulsador conectado al pin 3, los led se enciendan consecutivamente, uno a uno, solo uno por vez, desde el 6 al 13, durante 0,8 segundos cada led.

Importante!: primero configure el pin 3 como entrada.

4) Implemente los bloques de códigos escritos en los puntos 2 y 3 como rutinas de servicios de respectivas interrupciones, que deberán dispararse al pulsar los pulsadores 2 y 3 (Necesitará utilizar la primitiva “attachInterrupt”. En el anexo 1 encontrará instrucciones de uso).

Sugerencia: Implemente primero la rutina de servicio de un pulsador, verifique su correcto funcionamiento. Luego implemente la rutina de servicio para el próximo pulsador.

Nota: La función *delay(tiempo)* puede no funcionar, o funcionar de manera diferente si la utiliza dentro de una rutina de servicio. Deberá utilizar la función *delayMicroseconds(tiempo)* donde tiempo es el tiempo en microsegundos que dura la pausa. Deberá considerar que el valor máximo que puede tomar el argumento *tiempo* de la función *delayMicroseconds(tiempo)* es 16383 (0,016 seg), pero la pausa debe durar 0,8 segundos.

5) Analice experimentalmente:

Pulse los pulsadores en secuencias rápidas tales como 3 3 2, 2 3 2, 3 2 3 2, y analice el comportamiento del programa. En base a lo observado, conteste las siguientes preguntas (a través de la plataforma Moodle):

a - ¿Permite el microprocesador anidamiento de interrupciones?
b - ¿Cuál interrupción tiene prioridad?
c - ¿Qué ocurriría si dentro de una rutina de servicio se implementa un lazo (por ejemplo, un while()) del cual solo se sale al detectar que se produjo una interrupción de mayor prioridad?
d - ¿Por qué la función *delay(time)* funciona adecuadamente dentro del código

principal, pero funciona de manera diferente, o directamente no funciona, dentro de una rutina de servicio?.
e - ¿Qué ocurriría con las situaciones descriptas en los puntos c y d si el microprocesador permitiera interrupciones anidadas?

Actividad 2 - Funciones adicionales en sistemas embebidos

Los pines, además de las funciones digitalRead() y digitalWrite(), tienen muchas otras funciones asociadas. Veremos una de ellas, medición del ancho de un pulso con "unsigned long *pulseIn(pin,nivel,timeout)*", y utilizaremos esta función para medir distancia utilizando el sensor de distancia HC-SR04.
La función "*pulseIn(pin,nivel,timeout)*" mide el ancho de un pulso aplicado a un pin, donde:

- pin: pin en el cual queremos medir el ancho de un pulso.
- nivel: puede valer:
    - HIGH si el pulso es un pulso con nivel de tensión alto.
    - LOW si el pulso es un pulso con nivel de tensión bajo.

Pulso alto o HIGH

Pulso bajo o LOW

- Timeout: tiempo máximo en microsegundos que el procesador espera el pulso. Si después de este tiempo no se aplicó ningún pulso, el procesador interrumpe la ejecución de la función pulseIn(pin,nivel,timeout).

El sensor de distancia HC-SR04 funciona de la siguiente manera:

- a - Se aplica un pulso HIGH de al menos 10 microsegundos en el pin "Trig".
- b - Se espera un pulso HIGH en el pin "Echo". El ancho de dicho pulso en microsegundos es proporcional a la distancia. Para conocer la distancia en centímetros, se debe dividir por 58 (las señal de ultrasonido recorre un centímetro en 58 microsegundos).

Procedimiento:
1 - Conecte los pines de alimentación del sensor a +5V y GND, y los pines "Trig" y "Echo" a pines de entrada/salida del Arduino, por ejemplo el pin 5 para "Trig" y el pin 4 para "Echo".
2 - Configure estos pines como corresponde, el pin donde esté conectado "Trig" debe ser salida, y el pin donde está conectado "Echo" debe ser entrada.
3 - Escriba un programa que genere un pulso en el pin "Trig" como el requerido.

4 - Utilice la función “unsigned long pulseIn(pin,nivel,timeout)” para leer el ancho del pulso. Imprima la información de la distancia por pantalla (recuerde dividir el valor leído por 58).
5 - Escriba un programa que implemente una alarma de distancia que realice las siguientes tareas:

a - Mida la distancia cada un segundo (ya hecho arriba).

b - Al presionar uno de los pulsadores, la alarma debe activarse. Puede utilizar una de las rutinas de servicio creada en la actividad 1 y agregar el código necesario para activar la alarma. Se debe mostrar por pantalla la frase “Alarma activada”. Si se detecta que la distancia del intruso es menor a 1 metro, todos los leds deben parpadear rápidamente, y se debe mostrar por pantalla un mensaje de alerta de intruso.

c - Al presionar otro pulsador, la alarma debe desactivarse. Puede utilizar una de las rutinas de servicio creada en la actividad 1 y agregar el código necesario para desactivar la alarma. Se debe mostrar por pantalla “Alarma desactivada”. Si se detecta que la distancia es menor a 1 metro, no debe mostrarse nada ni realizar ninguna acción, ya que la alarma está desactivada.

Actividad 3 - Entrada salida mapeada en memoria y mediante instrucciones especiales - Lenguaje ensamblador

6) Implemente la instrucción que enciende el led 13 mediante lenguaje ensamblador, utilizando mapeo en memoria.

7) Implemente la instrucción que enciende el led 12 mediante lenguaje ensamblador, utilizando instrucciones específicas de entrada/salida.

8) Teniendo en cuenta que la memoria de datos del Atmega 328 (microprocesador utilizado por el Arduino Uno) tiene un tamaño de 2 KBytes, responda las siguientes preguntas (a través de la plataforma Moodle):
a - ¿Podría utilizar las instrucciones del punto 6 para leer cualquiera de las posiciones de memoria contenidas en los 2 KBytes de la memoria de datos y llevarla a un registro para luego realizar una operación aritmética? ¿Porqué?
b - ¿Podría utilizar las instrucciones del punto 7 para leer cualquiera de las posiciones de memoria contenidas en los 2 KBytes de la memoria de datos y llevarla a un registro para luego realizar una operación aritmética? ¿Porqué?

9) Escriba todas las instrucciones de entrada/salida del programa, incluso las instrucciones de configuración de los pines, en lenguaje ensamblador.

Actividad 4 - Análisis de la arquitectura de una computadora mediante comandos Linux

4.1 Mapas de buses y periféricos

Realice un mapa de la arquitectura de las siguientes computadoras:

- Computadora del laboratorio de informática.
- Computadora Raspberry Pi 3.

Los mapas podrá realizarlos en papel (y tomar foto) o con algún software de dibujo. Los mapas deberán indicar los buses, los puentes entre buses y los periféricos conectados a diferentes buses. Deberá subir esos mapas a través de las preguntas correspondientes en la plataforma Moodle.
Para la computadora Raspberry Pi 3, encontrará un archivo con la respuesta de una computadora Raspberry Pi 3 al comando lshw en el aula abierta (carpeta "Unidad 4 - Entrada Salida").

4.2 Analizando dispositivos de entrada/salida de una computadora

Con la información obtenida con los comandos de la actividad 4.1 responda las preguntas que encontrará en la plataforma Moodle.

Anexo 1: Escribiendo programas en Arduino

Estructura básica de un programa en el IDE de Arduino:

```
void setup() {
/*Dentro de la función setup se implementa el código que configura la plataforma. Acá deben configurarse pines como entrada/salida, periféricos a utilizar, habilitar interrupciones y declarar el nombre de las respectivas rutinas de servicio. La función setup solo se ejecuta una vez al iniciar la plataforma.*/
}
void loop() {
/*La función loop se ejecuta repetidamente sin fin. Acá debe colocar el código que se ejecutará repetidamente mientras la plataforma esté encendida*/
}

void función(){
/*Acá puede declarar funciones y rutinas de servicio de interrupciones.*/
}
```

```
void rutina_de_servicio(){
/*Una rutina de servicio se implementa como una función que será llamada automáticamente cuando ocurra la interrupción. Deberá primero habilitar la interrupción en la función setup() y declarar que función implementa su rutina de servicio, mediante la instrucción:
attachInterrupt(número de la fuente de interrupción, nombre de la rutina de servicio, evento que dispara la 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 valor.
RISING: dispara la interrupción cuando se detecta un flanco de subida.
FALLING: dispara la interrupción cuando se detecta un flanco de bajada. */
}
```

Fuente: https://www.arduino.cc/reference/en/#functions

Para ejecutar código en lenguaje ensamblador dentro de C utilize:

```
asm(
   "Intrucción en ensamblador 1\n"
   "Intrucción en ensamblador 2\n"
   ....
   ....
    );
```

Funciones útiles en C para Arduino:

```
pinMode(13, OUTPUT); //declara en pin 13 como salida. No retorna nada.
pinMode(3, INPUT); //declara en pin 3 como entrada. No retorna nada.
digitalRead(pin); // Lee el valor del pin. Retorna un entero que puede ser 1 o 0.
digitalWrite(pin, value); //Escribe val en el pin indicado, val puede valer HIGH o LOW.
Serial.begin(9600);//Inicializa el puerto serie.
Serial.println(datos a imprimir);//imprime datos en el puerto serial.
```

Algunas instrucciones útiles que puede ejecutar el procesador Atmel Mega 328.

```
ORI Rd, K (Logical OR Register and Constant Rd ← Rd v K)
ANDI Rd, K (Logical AND Register and Constant)
OR Rd, Rr (Logical OR Registers Rd ← Rd v Rr)
```

AND Rd, Rr (Logical AND Registers Rd ← Rd · Rr)
LDI Rd, K (Load Immediate Rd ← K)
NOP (No Operation)
Instrucciones que hacen uso de la memoria principal (y se emplean para entrada salida mapeada en memoria)
LDS Rd, k (Load Direct from SRAM Rd ← (k))
STS k, Rr (Store Direct to SRAM (k) ← Rr)
MOV Rd, Rr (Move Between Registers Rd ← Rr)
Instrucciones que no hacen uso de la memoria principal (y emplean entrada salida mediante instrucciones especiales)
IN Rd, A (In from I/O Location Rd ← I/O (A))
OUT A, Rr (Out to I/O Location I/O (A) ← Rr)
SBI P,b (Set Bit in I/O Register. I/O (P,b) ← 1)
CBI P,b (Clear Bit in I/O Register. I/O (P,b) ← 0)

Mapeo de pines del puerto B:
PuertoB bit 7: no pin
PuertoB bit 6: no pin
PuertoB bit 5: pin 13
PuertoB bit 4: pin 12
PuertoB bit 3: pin 11
PuertoB bit 2: pin 10
PuertoB bit 1: pin 9
PuertoB bit 0: pin 8

Mapeo de pines del puerto C:
PuertoC bit 7: no pin
PuertoC bit 6: pin RESET
PuertoC bit 5: pin A5
PuertoC bit 4: pin A4
PuertoC bit 3: pin A3
PuertoC bit 2: pin A2
PuertoC bit 1: pin A1
PuertoC bit 0: pin A0

Mapeo de pines del puerto D:
PuertoD bit 7: pin 7
PuertoD bit 6: pin 6
PuertoD bit 5: pin 5
PuertoD bit 4: pin 4
PuertoD bit 3: pin 3

PuertoD bit 2: pin 2
PuertoD bit 1: pin 1
PuertoD bit 0: pin 0

Direcciones memoria donde se encuentran mapeados los registros de los puertos B y C:
PORTB: (dirección 0x25) Para escribir el Puerto cuando es salida (o poner en pull-up (1) si es entrada)
DDRB (dirección 0x24) selecciona la dirección: 1 salida, 0 entrada
PINB: (dirección 0x23) Para leer el puerto cuando es entrada
PORTC: 0x28
DDRC: 0x27
PINC: 0x26
PORTD: 0x2B
DDRD: 0x2A
PIND: 0x29

Para utilizar los puertos anteriores con instrucciones especiales, debe utilizarse el mapa de memoria de entrada salida, siendo las direcciones:
PORTB: (dirección 0x05)
DDRB (dirección 0x04)
PINB: (dirección 0x03)
PORTC: 0x08
DDRC: 0x07
PINC: 0x06
PORTD: 0x0B
DDRD: 0x0A
PIND: 0x09

Fuente: Hoja de datos del procesador Atmel Mega 328.

Anexo 2: Obteniendo información sobre una computadora

Puede obtener información sobre una computadora siguiendo alguno de los siguientes procedimientos:

- En Linux: Ejecute los programas lshw, lspci y lsusb en su computadora (si no los tiene instalados, puede descargarlos desde los repositorios de Linux).
    - Puede generar una salida fácilmente legible con “sudo lshw -html > nombre_archivo.html”. Este comando mostrará los resultados en un archivo html que será creado en su carpeta principal.

- EN Windows:
  Programa “dxdiag”: Use la barra de búsqueda de programas y archivos para buscar y ejecutar la aplicación dxdiag (o presione Windows+R para abrir la herramienta “ejecutar”). Si se le pregunta si desea comprobar firmas digitales, seleccione no.
  - Herramienta “Administrador de dispositivos”. Podrá encontrarla en el Panel de Control (Menú Inicio de Windows -> Panel de Control).
  - Herramienta Información del Sistema. Use la barra de búsqueda de programas y archivos para buscar y ejecutar la aplicación Información del Sistema (o presione Windows+R para abrir la herramienta “ejecutar” y ejecute “msinfo32”. También puede encontrarla en Accesorios->Herramientas del sistema).