**EJERCICIOS PARA EL MÓDULO ENTRENADOR UNIVERSAL**

**CLASE01 - EJERCICIO 01**

Incrementar un contador cada segundo, desde 0 hasta 9. Primero, mostrar el valor de la cuenta en formato binario, en 04 LEDs. Luego, mostrar el valor de la cuenta en formato decimal, en un display de 7 segmentos.

**CLASE 01 - EJERCICIO 02**

Conectar el microcontrolador a 08 LEDs y luego, implementar 3 funciones llamadas Secuencia\_LEDs\_01, Secuencia\_LEDs\_02 y Secuencia\_LEDs\_03. Cada función debe implementar una secuencia de luces indicada por el instructor. Emplear máscaras para el encendido/apagado de los LEDs.

**CLASE 01 - EJERCICIO 03**

Conectar el microcontrolador a 02 pulsadores (BOTON1 y BOTON2), con resistencia pull-down, y a 03 LEDs (LED1, LED2 y LED3). Luego, escribir un programa que implemente la siguiente lógica: LED1 enciende cuando BOTON1 se encuentra presionado; LED2 enciendo cuando BOTON1 y BOTON2 (ambos) se encuentran presionados; LED3 enciende cuando BOTON1 o BOTON2 (únicamente uno de ellos) se encuentra presionado.

**CLASE 01 - EJERCICIO 04**

Conectar el microcontrolador a 02 pulsadores (BOTON1 y BOTON2) y 08 LEDs. Luego, escribir un programa que incremente y decremente una variable contadora de 8 bits. Si se pulsa BOTON1, el contador incrementa; si se pulsa BOTON2, el contador decrementa. Mostrar el valor de la cuenta en formato binario los LEDs

**CLASE 01 - EJERCICIO 05**

Conectar el microcontrolador a 03 pulsadores (BOTON1, BOTON2 y BOTON03), una pantalla LCD 2x16 y un zumbador (ALARM). Luego, escribir un programa que incremente y decremente una variable contadora, la cual puede tener un valor mínimo de 0 y un valor máximo de 15. Si se pulsa BOTON1, el contador incrementa; si se pulsa BOTON2, el contador decrementa; si se pulsa BOTON3 mientras el zumbador se encuentra apagado, éste se enciende; si se pulsa BOTON3 mientras el zumbador se enceuntra enecendido, éste se apaga. Visualizar en la LCD el valor actual de la cuenta y el estado del zumbador.

**CLASE 01 - EJERCICIO 06**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CLASE 02 - EJERCICIO 01**

Conectar el microcontrolador a un pulsador (BOTON1), a un LED (LED1) y a un zumbador (ALARM). Si se pulsa BOTON1, LED1 se enciende y el zumbador emite sonido de forma intermitente, conmutando cada medio segundo. Si BOTON1 se vuelve a pulsar, LED1 se apaga y el zumbador deja de sonar. Configurar el Timer 0 en Modo Normal para escanear el pulsador cada 4ms (aproximadamente).

**CLASE 02 - EJERCICIO 02**

Conectar el microcontrolador a un teclado matricial 4x4 y una pantalla LCD 2x16 y un LED (LED1). Ingresar una clave de 4 caracteres mediante el teclado matricial; si la clave es incorrecta, la LCD muestra el mensaje “clave incorrecta” durante 1 segundo, y luego permite ingresar la clave nuevamente; si la clave es correcta, la LCD muestra el mensaje “clave correcta”, el LED se enciende y el programa se detiene. Escanear el teclado matricial cada 10ms (aproximadamente).

**CLASE 02 - EJERCICIO 03**

Conectar el microcontrolador a un LED (LED1). Luego, incrementar gradualmente el brillo del LED desde el mínimo hasta el máximo posible. Repetir este procedimiento permanentemente. Emplear el Timer0 en Modo Fast PWM para llevar a cabo esta tarea.

**CLASE 02 - EJERCICIO 04**

Conectar el microcontrolador a una pantalla LCD 2x16, un motor DC (con su respectivo driver L298), 03 pulsadores (BOTON1, BOTON2 y BOTON3) y un LED (LED1). Al pulsar BOTON1 o BOTON2, la velocidad del motor incrementa o decrementa, respectivamente. Al presionar BOTON3 el motor cambia su sentido de giro (horario, antihorario) o se detiene, dependiendo de su estado anterior. Emplear el Timer0 en Modo Fast PWM para llevar a cabo esta tarea. Mostrar en la LCD el sentido de giro del motor, y también el ciclo de trabajo de la PWM en términos de porcentaje.

NOTA: No intente cambiar el sentido de giro del motor sin antes haberlo detenido por un tiempo.

**CLASE 02 - EJERCICIO 05**

Conectar el microcontrolador a un servomotor de baja potencia (Micro Servo modelo SG90). Luego, configurar el Timer1 en Modo PWM con Periodo Ajustable, y generar una señal PWM con un periodo de 20ms. Variar el ciclo de trabajo de la PWM de modo que, al conectar la señal al servomotor, éste se desplace lentamente desde la posición 0° hasta la posición 180°, y que repita este recorido permanentemente.

**CLASE 02 - EJERCICIO 06**

Conectar el microcontrolador a 02 LEDs y configurar el Timer1 en Modo PWM con Periodo Ajustable, de tal manera que se pueda generar dos señales PWM con una misma frecuencia (100 Khz), pero con ciclos de trabajo independientes.

**CLASE 03 – EJERCICIO 01**

Conectar el microcontrolador a 04 displays de 7 segmentos multiplexados, cátodo común. Luego, escribir un código que incermente un contador desde 1000 hasta 1020, y que muestre el valor de esta cuenta en los displays multiplexados. Realizar esta tarea empleando retardos por software (\_delay\_ms ó \_delay\_us).

**CLASE 03 – EJERCICIO 02**

Conectar el microcontrolador a dos pulsadores (BOTON1, BOTON2) y 04 displays de 7 segmentos multiplexados, cátodo común. Luego, escribir un programa en donde la gestión de los habilitadores de los displays y la lectura de los pulsadores se lleve a cabo cada 2ms. Realizar la gestión de los habilitadores y el escaneo de los botones en una rutina de servicio a interrupción (ISR) que ocurra cada vez que el Timer0 se iguala a OCR.

**CLASE 03 – EJERCICIO 03**

Conectar el microcontrolador a un receptor infrarrojo TSOP1738 (receptor IR + Filtro) y a 04 dsiplays de 7 segmentos multiplexados,, cátodo común. Luego, empleando un control remoto infrarrojo, enviar al microcontrolador los números del 0 al 9. Los números transmitidos por el control remoto deben visualizarse en uno de los displays multiplexados. Emplear interrupciones internas (temporizadores) e interrupciones externas (cambio de nivel en un pin) para llevar a cabo esta tarea.

**CLASE 03 – EJERCICIO 04**

Conectar un potenciómetro al CANAL 0 del ADC del microcontrontrolador, y también conectar 04 displays de 7 segmentos multiplexados, cátodo común. Visualizar en los displays el valor ADC (binario de 10 bits) correspondiente a la medición del potenciómetro. Emplear retardos por software (\_delay\_ms) para establecer un periodo de muestreo de 100ms. Emplear interrupciones para establecer la frecuencia con que se gestionan los habilitadores de los displays.

**CLASE 03 – EJERCICIO 05**

Conectar un sensor de temperatura LM35 al CANAL 2 del ADC del microcontrontrolador, y también conectar 04 displays de 7 segmentos multiplexados, cátodo común. Visualizar en los displays el valor de la temperatura (en grados Celsius) correspondiente a la medición del sensor. Emplear interrupciones para establecer un periodo de muestreo de 100ms y también para actualizar los datos en los displays.

**CLASE 03 – EJERCICIO 06**

Conectar dos potenciómetros al ADC del Microcontrolador (CANAL 0 y CANAL 1), y también conectar una pantalla LCD 2x16. Luego, escribir un programa que permita medir el valor ADC en AMBOS potenciómetros en un periodo de 100ms. Mostrar en la pantalla LCD los valores de voltaje correspondientes a cada uno de los potenciómetros. Emplear interrupciones para establecer un periodo de muestreo de 50ms (para cada canal) y también para actualizar los datos en la LCD.

**CLASE 04 – EJERCICIO 01**

Escribir una función que permita transmitir un byte de manera serial, de acuerdo al formato RS-232 en niveles TTL. La información debe ser transmitida por un pin de salida (SERIAL\_TX) con una tasa de transferencia de2400 bps, la función debe recibir como parámetro una variable de tipo **char** y la duración de los bits debe definirse mediante el uso de la función \_delay\_us(). Transmitir el mensaje “Hola mundo.” periódicamente para verificar que la función implementada trabaja correctamente. Evaluar su funcionamiento con diferentes tasas de transferencia de bits.

**CLASE 04 – EJERCICIO 02**

Empleando la función implementada en el Ejercicio 1, escribir una función que permita transmitir una cadena de caracteres (string). La función debe recibir como parámetro un puntero a un caracter.

Luego , escribir una función que permita transmitir el valor de un número entero de 32 bits con signo (solo los dígitos significativos). Tomar en cuenta que si el número es negativo debe imprimirse el caracter ‘-’ delante del número.

NOTA: Tomar en cuenta que el último caracter de un string es siempre ‘\0’.

**CLASE 04 – EJERCICIO 03**

Conectar un potenciómetro al microcontrolador. Luego, adaptar las funciones anteriores para utilizarlas con el módulo UART, y escribir un programa que utilice el ADC para muestrear el valor entregado por el potenciómetro cada segundo. Una vez que se haya completado 20 muestras, se debe imprimir (en una terminal serial) los valores de las muestras en un formato tipo vector, que pueda ser leído y graficado en MATLAB.

**CLASE 04 – EJERCICIO 04**

Escribir un programa que reciba caracteres, de forma asíncrona, a través del USART y que transmita el mismo caracter de vuelta, también a través del USART. Verificar el funcionamiento del programa enviando datos desde una computadora con una terminal serial. Primero, emplear una tasa de transferencia de 2400 bps, y luego repetir las pruebas con una tasa de 115200 bps.

**CLASE 04 – EJERCICIO 05**

Conectar un display de 07 segmentos (o un conjunto de 08 LEDs) a uno de los puertos del microcontrolador. Luego, escribir un programa donde el microcontrolador reciba los caracteres ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’ y ‘p’ a través del puerto serial y que, dependiendo del caracter que reciba, conmute el estado del segmento (o el LED) correspondiente.

**CLASE 04 – EJERCICIO 06**

Modificar el programa del ejercicio anterior, de modo que el microcontrolador reciba el comando “toggle”, seguido de un espacio, la posición del LED que se desea manipular (de 0 a 7), y un salto la tecla ENTER (retorno de carro y/o salto de línea). Si se desea manipular todos los LEDs a la vez, entonces debe escribirse ‘A’ en lugar de la posición. Por ejemplo: “toggle 2” conmuta el LED 2 y "toggle A" conmuta todos los LEDs. Si se escribe un comando inválido, debe imprimirse un mensaje que lo indique en la consola serial.

**CLASE 05 – EJERCICIO 01**

Elaborar una librería para configurar la interfaz SPI del microcontrolador. Esta librería debe permitirnos configurar fácilmente los siguientes parámetros: pre-scaler para la frecuencia de la señal de reloj (SCK), polaridad de la señal de reloj (CPOL) y fase de la señal de reloj (CPHA). Luego haber completado la librería, revisar la hoja de datos del circuito integrado MAX6675.

**CLASE 05 – EJERCICIO 02**

Conectar una termocupla tipo K a un conversor MAX6675, y conectar este conversor a la interfaz SPI del microcontrolador. Además, conectar el microcontrolador al puerto serial de una computadora (PC o laptop). Luego, escribir un programa que permita medir el valor proporcionado por el conversor y visualizar la temperatura en una terminal serial. La medición y visualización de la temperatura debe realizarse cada 1 segundo. Una vez que el programa funcione correctamente, definir dos constantes: LIMITE\_SUPERIOR y LIMITE\_INFERIOR. Si la temperatura leída se encuentra por debajo del LIMITE\_INFERIOR o por encima del LIMITE\_SUPERIOR, entonces se activa una alarma (buzzer).

**CLASE 05 – EJERCICIO 03**

Conectar dos termocuplas (TC1 y TC2) a dos conversorer MAX6675, y conectar estos conversores a la interfaz SPI del microcontrolador. Además, conectar el microcontrolador al puerto serial de una computadora (PC o laptop). Luego, escribir un programa que permita medir el valor proporcionado por los dos conversores (empleando las líneas CS) y visualizar las temperaturas en una terminal serial. La medición y visualización de las temperaturas debe realizarse cada 1 segundo. Una vez que el programa funcione correctamente, definir dos constantes: LIMITE\_SUPERIOR\_TC1, LIMITE\_INFERIOR\_TC1, LIMITE\_SUPERIOR\_TC2 y LIMITE\_INFERIOR\_TC2. Si la temperatura leída desde TC1 sale de alguno de sus límites, se activa una alarma (buzzer) de forma continua; si la temperatura leída desde TC2 sale de alguno de sus límites, se activa una alarma (buzzer) de forma intermitente, con un periodo de 1 segundo.

**CLASE 06 – EJERCICIO 01**

Conectar un chip de memoria EEPROM 24C32 (incluido en el módulo Tiny RTC) a la interfaz I2C del microcontrolador. Luego, elaborar un programa para escribir un mensaje (por ejemplo “Hola”) dentro de las primeras posiciones de la memoria EEPROM e imprimir el mensaje en una terminal serial. Finalmente, intercambiar de módulo (Tiny RTC) con su compañero para leer lo que él ha escrito en su chip de memoria y visualizarlo en una terminal serial.

**CLASE 06 – EJERCICIO 02**

Conectar un reloj en tiempo real DS1307 (incluido en el módulo Tiny RTC) a la interfaz I2C del microcontrolador. Luego, elaborar un programa que, al iniciar, permita ajustar la fecha y hora. Después del ajuste, el programa debe imprimir la fecha y hora, cada segundo, en una terminal serial.

**CLASE 06 – EJERCICIO 03**

Conectar un reloj en tiempo real DS1307 y un chip de memoria EEPROM 24C32 a la interfaz I2C del microcontrolador. Asimismo, conectar un conversor MAX6675 a la interfaz SPI del microcontrolador. Luego, escribir un programa que mida la temperatura cada segundo y, si esta se encuentra por encima de un umbral (TEMPERATURA\_MAXIMA), la información de temperatura, fecha y hora es almacenada en la EEPROM. Además, visualizar la fecha, hora y temperatura en una terminal serial, con un periodo de 1 segundo.

**CLASE 07 – EJERCICIO 01**

Control de temperatura mediante algoritmo PID, empleando un sensor LM35 y una resistencia cerámica. La regulación de la potencia proporcionada se realiza mediante modulación por ancho de pulso (PWM).

**CLASE 07 – EJERCICIO 02**

Control de temperatura mediante algoritmo PID, empleando una termocupla tipo K y una resistencia calefactora alimentada con 220V. La regulación de la potencia proporcionada se realiza mediante el método de control de paquetes.

**CLASE 07 – EJERCICIO 03**

Control en lazo abierto de intensidad de luz de una lámpara incandescente. La regulación de la potencia proporcionada se realiza mediante el ajuste del ángulo de disparo en un TRIAC.

**CLASE 10 – EJERCICIO 01 (Solamente para AVR)**

Escribir un programa que permita controlar la posición de un servomotor (SERVO\_2) mediante una señal PWM que sea generada por hardware, específicamente con el generador de señales del módulo Timer1. La posición de este servo puede ajustarse pulsando los botones S3 y S4, de la siguiente manera: cada vez que se pulsa S3, la posición del servo se incrementa en 10° (se desplaza en sentido horario); cada vez que se pulsa S4, la posición del servo se decrementa en 10° (se desplaza en sentido horario). Tomar en cuenta que el rango de trabajo del servomotor es desde 0° hasta 180°, y los botones no deben tener ningún efecto si se alcanzan los límites.

**CLASE 10 – EJERCICIO 02 (Solamente para AVR)**

Escribir un programa que permita controlar la posición de un servomotor (SERVO\_2) mediante una señal PWM que sea generada por hardware, específicamente con el generador de señales del módulo Timer1. La posición de este servo puede ajustarse moviendo el potenciómetro POT\_2, y leyendo el potenciómetro con el módulo ADC. Cuando el valor ADC es 0, el motor se ubica en la posición 0°; cuando el valor ADC es 1023, el motor se ubica en la posición 180° (intrapolar los demás valores).

**CLASE 10 – EJERCICIO 03**

Escribir un programa que permita controlar la posición de 03 sermomotores (SERVO\_1, SERVO\_2 y SERVO\_3) mediante 03 señales PWM que sean generadas por software, de preferencia empleando una rutina de servicio a interrupción (ISR). La posición de los servos puede ajustarse pulsando las parejas de botones S1-S2, S3-S4 y S5-S6, para la posición de SERVO\_1, SERVO\_2 y SERVO\_3, respectivamente. Tomando como ejemplo SERVO\_1: cada vez que se pulsa S1, la posición del servo se incrementa en 10° (se desplaza en sentido horario); cada vez que se pulsa S2, la posición del servo se decrementa en 10° (se desplaza en sentido horario). Tomar en cuenta que el rango de trabajo del servomotor es desde 0° hasta 180°, y los botones no deben tener ningún efecto si se alcanzan los límites.

**CLASE 10 – EJERCICIO 04**

Escribir un programa que permita controlar la posición de 03 sermomotores (SERVO\_1, SERVO\_2 y SERVO\_3) mediante 03 señales PWM que sean generadas por software, de preferencia empleando una rutina de servicio a interrupción (ISR). La posición de estos servos puede ajustarse moviendo los potenciómetros POT\_1, POT\_2 y POT\_3, para SERVO\_1, SERVO\_2 y SERVO\_3, respectivamente. Los potenciómetros son leídos por el módulo ADC y se sigue la misma lógica que en el Ejercicio 02.