Práctica 2. Manejo de Timers, pwm y UART: Servos y motores paso a paso

1. Objetivo

En esta segunda práctica de la parte digital, perseguimos varios objetivos:

- Realizar control pwm con variación de parámetros
- Realizar un pequeño monitor por puerto serie
- Manejar un motor paso a paso

2. Material necesario

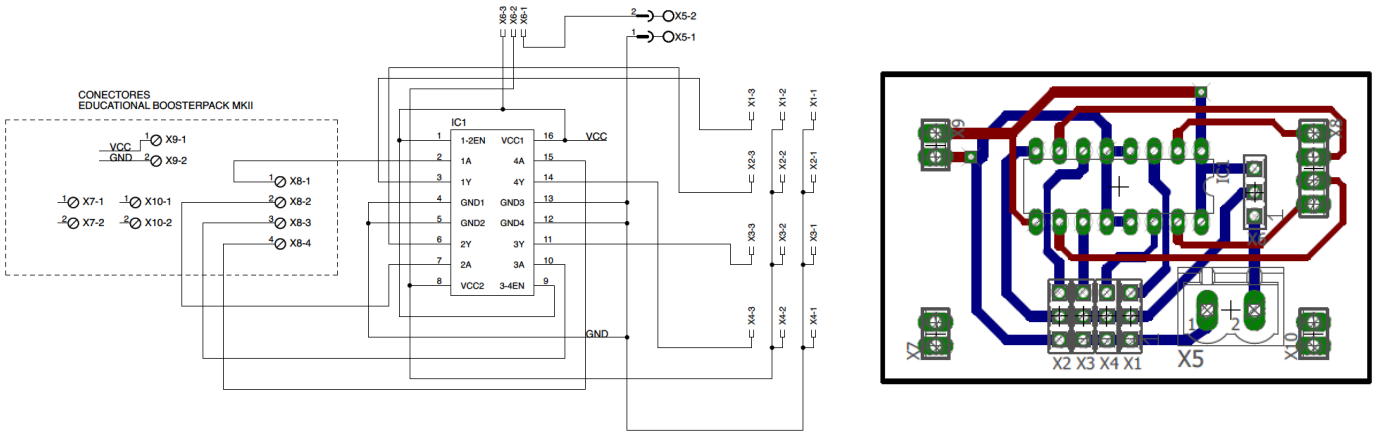
- Connected Launchpad de Texas Instruments con el microcontrolador TIVA TM4C1294NCPDT
- Manual de la librería de funciones DriverLib y datasheet del microcontrolador
- Placa de conexión Servo-Tiva
- Servomotor 5V y motor paso a paso con reductora.
- Programa TeraTerm, Putty o similar instalado en el ordenador

3. Fundamento teórico

Para la realización de la práctica hará falta conocer el funcionamiento de los diferentes periféricos explicados en clase, en particular los timer, las interrupciones de los mismos, los moduladores pwm, y el puerto serie asíncrono (UART)

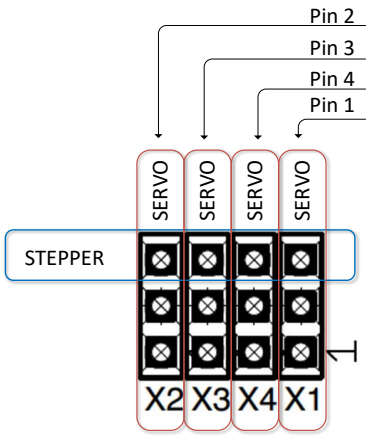
También es necesario conocer la placa de conexión diseñada para actuar sobre los motores paso a paso y servomotores, así como las conexiones de los propios dispositivos.

La placa de conexión incorpora un driver L293D conectada a los cuatro primeros pines del conector derecho interior (PF1, PF2, PF3, PG0 en el primer Boosterpack, o PG1, PK4, PK5 y PM0 si se usa la segunda posición):



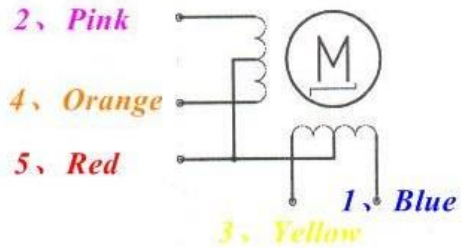
El conector de 12 pines formado por la unión de los conectores X1, X2, X3, X4 permite conectar en vertical hasta 4 servos, de manera que las líneas de alimentación y tierra estén en la parte inferior del mismo, y las señales de control en los pines superiores de estos conectores.

De igual forma, estos cuatro pines internos pueden ser usados para conectar un motor paso a paso, prescindiendo de los otros pines:



Por otra parte, es necesario conocer las características y la conexión del motor paso a paso a utilizar. En concreto, se trata de un motor paso a paso de 32 pasos con una reductora de 1/16.032 que ofrece un total de 513 pasos por vuelta del eje. El esquema de conexiones suministrado por el fabricante es:

Por lo que para que gire según las agujas del reloj el orden de activación será rosa-amarillo-naranja-azul. El terminal intermedio (rojo) se dejará sin conexión.



4. Realización de la práctica

I. Primer Ejercicio: Manejo de un Servomotor.

Partiendo del ejemplo disponible en Enseñanza virtual, se realizará un programa que maneje un servo conectado al pin PK4. El servo empezará estando en el estado central. Al pulsar el botón 1, hará el recorrido desde el punto central hasta el extremo izquierdo (giro en sentido contrario a las agujas del reloj), y volverá al punto inicial. De manera análoga, al pulsar el otro botón se realizará el giro contrario.
Continuando con el empleo de *buenas prácticas* de programación, se desea que todo el proceso se realice siguiendo un esquema óptimo de consumo de energía:

- Programar la lectura de los botones por interrupción
- En el bucle infinito, poner el sistema en modo de bajo consumo
- Cuando la rutina de interrupción salte (y saque al sistema del modo de bajo consumo), ejecutar el movimiento requerido. El movimiento NO SE DEBE realizar en la rutina de interrupción sino en el bucle principal.
- Tras ejecutar el movimiento, volverá a repetir el bucle con lo que retornará al modo de bajo consumo.

Para poder usar las características de bajo consumo, es necesario usar las siguientes funciones de la librería sysctl.h:

- `extern void SysCtlPeripheralSleepEnable(uint32_t ui32Peripheral):` especifica que el periférico indicado permanece encendido durante

el modo Sleep. Es necesario para que se produzcan las interrupciones que despierten al sistema.

- `extern void SysCtlPeripheralSleepDisable(uint32_t ui32Peripheral)`: Deshabilita un periférico específico. Puede ser útil si queremos que un periférico en concreto esté deshabilitado en alguna parte del proceso.
- `extern void SysCtlPeripheralClockGating(bool bEnable)`: Por defecto, todos los periféricos están habilitados, independientemente de lo que digan las dos funciones anteriores, hasta que se ejecute esta instrucción pasando un 'true' (1) como parámetro. Si se le pasa un 'false' (0), se deshabilita el apagado de los periféricos en modo bajo consumo.

Una vez configurado el modo de bajo consumo, para pasar a dicho modo basta invocar la función:

```
extern void SysCtlSleep(void);
```

Una vez que el periférico habilitado (los pines de entrada salida en este caso) despierte al sistema, cuando retorne de la rutina de interrupción ya volverá en modo normal de funcionamiento. Puede resultar de ayuda consultar el ejemplo2_5

II. Segundo Ejercicio: Monitorización del proceso.

En este segundo apartado, realizaremos una monitorización del proceso anterior. Suponiendo que cada uno de los botones es un sensor que detecta una pieza tipo A o tipo B, y que en función de eso está moviendo el servo para mandarla hacia un lado o hacia otro, se realizará una modificación en el programa anterior para que, cada vez que se pulse un botón, se mande un mensaje a la consola, informando de que se ha detectado una pieza (de tipo A o tipo B), y la hora (en HH:MM:SS desde que se arranca el proceso). El mensaje podría ser algo como:

```
>> [hh:mm:ss] Pieza tipo A detectada.
>>  XX piezas tipo A / YY piezas tipo B
```

III. Tercer Ejercicio: Motor paso a paso.

Para el manejo de un motor paso a paso es conveniente saber que, para que gire en un determinado sentido, se deben ir activando de manera consecutiva las bobinas del mismo. Por lo tanto, con el motor conectado convenientemente al microcontrolador, bastará con ir siguiendo la secuencia de activación para dar tensión en un sentido o en el otro a las bobinas. Para que el motor se mueva con fluidez, los pasos tienen que tener una duración mínima de 5ms (empíricamente calculado).

Partiendo del ejemplo existente en EV, realizar un programa que simule el funcionamiento del segundero de un reloj "analógico". Para ello, deberá moverse 1/60 de vuelta, y permanecer en la postura resultante hasta que acabe el tiempo (1s). Nota: teniendo en cuenta que el sistema tiene 513 pulsos por vuelta, habrá que aproximar el ángulo resultante (8.5 pasos) dando una vez 8 pasos y la siguiente vez 9. En media, deberá dar la vuelta en 1s.